Table of Contents

- Introduction
- General Information
- Getting Started
  - Loading the Library
  - Adding HTML Elements
  - Working with the JavaScript Diagram Model
  - TypeScript and Code Completion
  - Starter Projects
- What's New in this Release
- Version History
- Moving to version 4
- User Interaction
- Programming Interface Overview
  - Model-View Architecture
  - User Interface Components
  - Diagram Structure
    - Standard Item Types
    - Composite Nodes
    - Creating and Deleting Items
    - Accessing Items
    - Collapsing and Expanding Tree Branches
    - Attaching and Grouping Items
  - User Interaction
    - Diagram Events
    - Controlling User Interaction
    - Alignment Grid
    - Undo and Redo
    - Scrolling and Zooming
    - Testability
  - Placement and Layout
    - Items Placement and Z-order
    - Lane Grid
    - Anchor Points and Patterns
    - Link Routing
    - Laying out the Diagram
  - Appearance
    - Customizing Node Appearance
    - Node Effects
    - Customizing Link Appearance
    - Styles and Themes
    - CSS Styling
    - Define Shape Geometries
  - Animations
  - Serializing Diagrams
  - Miscellaneous
- Tutorials
  - Tutorial 1: Loading Graph Data
  - Tutorial 2: Loading Hierarchical Data
  - Tutorial 3: Create a Custom Node Type
  - Tutorial 4: Custom Nodes Integration
- Integration with UI Frameworks
  - React
  - Vue.js
  - Angular

- Blazor
- Reference
  - MindFusion.Animations
    - Animation Class
    - AnimationType Enumeration
    - EasingType Enumeration
    - Events Class
  - MindFusion.Collections
    - Grid Class
    - Queue Class
  - MindFusion.Controls
    - Alignment Enumeration
    - CancelEventArgs Class
    - Canvas Class
    - CanvasControl Class
    - Control Class
    - Disposable Class
    - DomUtils Class
    - EventArgs Class
    - EventDispatcher Class
    - Events Class
    - MouseButton Enumeration
    - MouseCursors Enumeration
    - TickPosition Enumeration
    - WebComponents Class
    - ZoomControl Class
  - MindFusion.Diagramming
    - Action Enumeration
    - AddItemCommand Class
    - AdjustmentHandles Enumeration
    - AeroEffect Class
    - Alignment Enumeration
    - AnchorPattern Class
    - AnchorPoint Class
    - ArrowHeads Class
    - AutoResize Enumeration
    - Behavior Enumeration
    - BehaviorBase Class
    - Border Class
    - Builder Class
    - Button Class
    - CaptionedNode Class
    - Cell Class
    - CellEventArgs Class
    - CellFrameStyle Enumeration
    - ChangeItemCommand Class
    - ColumnStyle Enumeration
    - Command Class
    - CompatConfig Class
    - ComponentEventArgs Class
    - CompositeCommand Class
    - CompositeNode Class
    - CompositeRouter Class
    - ConnectionPoint Class
    - ConnectionStyle Enumeration
    - ContainerNode Class
    - ContainerPage Class
    - ContainerPagesCommand Class
    - ControlNode Class
    - CreateLinkController Class

CreateNodeController Class
CreateSelectionController Class
DelKeyAction Enumeration
Diagram Class
DiagramBase Class
DiagramDocument Class
DiagramEventArgs Class
DiagramItem Class
DiagramLink Class
DiagramNode Class
DiagramPage Class
DiagramView Class
DirectionConstraint Enumeration
DoNothingBehavior Class
DrawAnchorPointEventArgs Class
DrawLinksBehavior Class
DrawNodesBehavior Class
DrawShapesBehavior Class
DummyNode Class
EffectPhase Enumeration
Events Class
ExpandButtonAction Enumeration
Factory Class
FitSize Enumeration
FoldContainerCommand Class
FreeFormNode Class
GlassEffect Class
GlassEffectType Enumeration
GridColumn Class
GridPanel Class
GridRouter Class
GridRow Class
GridStyle Enumeration
HandlesStyle Enumeration
HandlesVisualStyle Class
HeaderEventArgs Class
HeaderResizeEventArgs Class
InplaceEditEventArgs Class
Interaction Class
InteractionState Class
ItemEventArgs Class
ItemLabel Class
ItemListView Class
LabelPosition Enumeration
LengthType Enumeration
LinkCrossingEventArgs Class
LinkCrossings Enumeration
LinkEventArgs Class
LinkLabel Class
LinkNodesBehavior Class
LinkShape Enumeration
LinkShapesBehavior Class
LinkTextStyle Enumeration
LinkValidationEventArgs Class
ListViewBase Class
MagnifyBehavior Class
Manipulator Class
MarkStyle Enumeration
ModificationStart Enumeration
ModifierKeyAction Enumeration

ModifierKeyActions Class
ModifyBehavior Class
ModifyLinkController Class
ModifyNodeController Class
ModifySelectionController Class
MouseButtonActions Enumeration
MouseInputDispatcher Class
MouseWheelAction Enumeration
MoveNodesBehavior Class
NodeConstraints Class
NodeEffect Class
NodeEventArgs Class
NodeLabel Class
NodeListView Class
Orientation Enumeration
Overview Class
PageContainerCommand Class
PagedContainerNode Class
PanBehavior Class
PanController Class
Panel Class
Path Class
PathFinder Class
PatternRouter Class
RelativeToLink Enumeration
RelativeToNode Enumeration
RemoveItemCommand Class
ResizeTableColumnCommand Class
ResizeTableRowCommand Class
RoutePattern Class
Router Class
RouteStep Class
RouteStepKind Enumeration
Ruler Class
SaveToStringFormat Enumeration
ScaleMode Enumeration
ScrollTableCommand Class
Selection Class
SelectionChangedEventArgs Class
SelectionEventArgs Class
SelectOnlyBehavior Class
SerializeTagEventArgs Class
ShadowsStyle Enumeration
Shape Class
ShapeDecoration Class
ShapeElement Class
ShapeLibrary Class
ShapeNode Class
Shapes Class
ShowAnchors Enumeration
SimplePanel Class
SimpleShape Enumeration
SinglePointerController Class
StackPanel Class
Style Class
SvgContent Class
SvgExporter Class
SvgNode Class
TableColumn Class
TableConnectionPoint Class

- TableNode Class
- TableRow Class
- Theme Class
- TreeItemEventArgs Class
- TreeViewConnectionPoint Class
- TreeViewItem Class
- TreeViewNode Class
- UndoEventArgs Class
- VideoNode Class
- WebComponents Class
- XmlPersistContext Class

MindFusion.Diagramming.Animations

- BounceAnimation Class
- FillColorAnimation Class
- ItemAnimation Class
- OpacityAnimation Class
- PathAnimation Class
- PulseAnimation Class
- ScrollAnimation Class
- ShakeAnimation Class
- StrokeColorAnimation Class
- ViewAnimation Class
- ZoomAnimation Class

MindFusion.Diagramming.Controls

- Events Class
- PageEventArgs Class
- Palette Class
- ShapeDesigner Class
- TabAlignment Enumeration
- TabbedDiagramView Class
- WebComponents Class

MindFusion.Diagramming.Lanes

- Cell Class
- Grid Class
- Header Class
- ResizeType Enumeration
- Style Class

MindFusion.Drawing

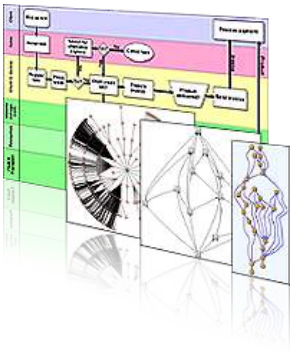
- Alignment Enumeration
- Border3D Class
- Component Class
- ComponentBase Class
- Container Class
- DashStyle Enumeration
- DrawingUtils Class
- Ellipse Class
- Font Class
- FontStyle Enumeration
- GraphicsUnit Class
- Image Class
- ImageAlign Enumeration
- LayoutAlignment Enumeration
- LineJoin Enumeration
- Path Class
- Point Class
- Rect Class
- Size Class
- StringAlignment Enumeration
- StringFormat Class
- Text Class

Thickness Class
Vector Class
Video Class
Visibility Enumeration
MindFusion.Graphs
Anchoring Enumeration
AssistantPosition Enumeration
AssistantType Enumeration
BorderedTreeLayout Class
Edge Class
FlowchartLayout Class
FractalLayout Class
Graph Class
GridLayout Class
LayeredLayout Class
Layout Class
LayoutDirection Enumeration
MultipleGraphsPlacement Enumeration
Orientation Enumeration
OrthogonalLayout Class
Path Class
PathFinder Class
RadialTreeLayout Class
SpringLayout Class
TopologicalLayout Class
TreeLayout Class
TreeLayoutLinkType Enumeration
TreeMapLayout Class
Vertex Class
Table of Predefined Shapes
Licensing
Online Resources
Technical Support
MindFusion Products

Introduction

MindFusion.Diagramming for JavaScript (JsDiagram for short) is a flow diagramming class library that can be used to display and edit graphs and diagrams on a web page. It is implemented entirely in JavaScript and renders graphics objects using the HTML 5 Canvas element. JsDiagram is written in modern JavaScript as a set of ES6 modules, and transpiled to also provide support for CommonJS, UMD and ES5. For Canvas support and heavy-duty geometric and layout calculations done in JavaScript, JsDiagram requires latest-generation browsers.

JsDiagram can give graphical representation to:



- Workflow processes
- Class relationships
- Graphs and networks
- Entity-relationship / database diagrams
- IVR systems
- Industrial automation processes
- Genealogy trees
- Algorithms and flowcharts
- Organizational charts
- Classification hierarchies
- And much more

The library provides intuitive user-interaction model for creating or editing diagrams. In addition, its programmatic interface places at your disposal many useful functions, ranging from event handling and JSON serialization to link-routing and automatic layout algorithms.

The control has been tested with the most popular modern browsers on desktop and mobile systems, and is used by companies from all around the world to create applications for various web-development platforms.

General Information

The JsDiagram Component

- Includes ES6, CommonJs and UMD modules
- Redistribution is royalty free
- Can be purchased with source code
- Is written in JavaScript
- Draws via the HTML5 Canvas element

User Interaction

- Several user interaction modes
- Scrolling and zooming
- Copy and paste support
- Panning
- Automatic scrolling while dragging items with the mouse
- Magnification
- Alignment to grid
- Multiple selection
- Tool-tips
- Docking points can be defined for nodes
- Undo / redo support
- Links can auto-snap to nearest node

Programming

- Model / view architecture
- Programmatic access to the diagram elements
- Rich event set
- Numerous properties for customizing appearance and behavior
- Numerous utility methods
- Builder classes for fluent API

Diagram Elements

- More than 100 predefined node shapes
- Rotation of shapes at arbitrary angles
- Container nodes
- Tables with unlimited number of rows and columns
- Table cells can span several columns and rows
- Display SVG drawings in nodes
- Links can comprise multiple segments
- Link segments can be straight lines or curves
- Display multiple labels per link
- Support for custom node and link types
- Video nodes
- Display hierarchical data in TreeView nodes
- Host HTML contents in nodes

Automatic Layout

- Layered graph layout
- Spring-Embedder graph layout
- Fractal tree layout
- Directional tree layout
- Radial tree layout
- Topological ordering layout
- Flowchart layout
- Orthogonal graph layout
- Automatic link routing

Appearance

- Nodes can display images and text
- Z-order
- Multiple text labels per item
- Various text alignment styles
- Glass visual effects
- Customizable fonts and colors
- Custom painting of nodes
- Shadows
- Transparent nodes
- Invisible nodes
- A background swimlane grid

Behavior

- Lock diagram elements
- Read-only mode supported
- Tables can be scrolled
- Collapse and expand hierarchy branches
- In-place text editing
- Dynamic positioning of links relative to connected nodes

Link Diagram Items

- Nodes to nodes

- Table rows to table rows
- Nodes to table rows
- Unconnected links

Auxiliary Controls

- Overview control
- NodeListView control
- Ruler control
- ZoomControl control

Import and Export

- Export Scalable Vector Graphics (SVG) files
- Printing and print preview

Front-end integration

- React component
- Vue.js component
- Angular component

Getting Started

This guide provides a step-by-step walkthrough of setting up and using the diagram library. It is divided into five main parts.

1. Loading the Library

Learn the different ways to include the diagramming library in your project. This section covers loading via simple <*script*> tags, using modern ES modules with import maps, and integrating it as an npm package for bundled applications.

2. Adding HTML Elements

Discover how to add the diagram component to your web page. This part explains the two main approaches: using a manual <*canvas*> element for direct control, or using the modern Web Components standard for better encapsulation.

3. Working with the JavaScript Diagram Model

Dive into the core objects that control your diagram's content and behavior. This section introduces the fundamental Diagram (the data model) and DiagramView (the visual control) classes and shows you how to create a diagram instance and add items to it.

4. TypeScript and Code Completion

Enable code completion in your editor and import the Diagramming API to use along with TypeScript compiler. This guide explains how to use the provided TypeScript declaration (.d.ts) files for real-time code suggestions and type-checking.

5. Starter Projects

Get a head start by cloning a pre-configured project from GitHub. This section provides links to repositories for vanilla JavaScript, npm/bundler setups, Electron, and popular frameworks like React, Angular, and Vue.

Loading the Library

There are several ways to load the diagramming library, depending on your project's needs. This guide covers the most common approaches:

- Simple & Quick: Use <*script*> tags to get started on a basic HTML page right away.

- Modern & Browser-native: Use modern ES Modules with an *importmap*.
- Professional & Scalable: Install *npm* package for projects that use a bundler like Webpack or Vite.

Choose the method that best fits your development setup.

A Note on Library Structure

The library is organized into six main modules or namespaces: Collections, Controls, Drawing, Diagramming, Animations and Graphs. In the UMD distribution, these are accessed via the global `MindFusion` object (e.g., `MindFusion.Diagramming`). When using ES modules or npm packages, you'll typically import these modules directly. You will see these concepts in action in the examples below.

For more information on JavaScript modules see https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Modules.

Loading UMD namespaced scripts

UMD is a pattern of universal module definition for JavaScript modules. These modules are capable of working irrespective of the executing environment. In web browsers, it exposes the module functionality to the global scope.

In order to load the UMD scripts, add references to files from diagram distrbution's */umd* folder using <script> tags. Note that the order of the scripts matters: controls.js must be loaded after collections and drawing, and diagramming.js must be loaded after animations and graphs. After these scripts are loaded, you can access the global MindFusion namespace object from your code.

HTML

```
<!DOCTYPE html>
<html>
<head>
    <title>My First Diagram</title>

    <!-- 1. Load the library scripts -->
    <script src="umd/collections.js" type="text/javascript"></script>
    <script src="umd/drawing.js" type="text/javascript"></script>
    <script src="umd/controls.js" type="text/javascript"></script>
    <script src="umd/animations.js" type="text/javascript"></script>
    <script src="umd/graphs.js" type="text/javascript"></script>
    <script src="umd/diagramming.js" type="text/javascript"></script>
</head>
<body>

    <!-- 2. Define the container for the diagram -->
    <div style="width: 600px; height: 400px; overflow: auto; border: 1px solid #ccc;">
        <canvas id="diagram" width="2100" height="2100">
        This page requires a browser that supports HTML 5 Canvas element.
        </canvas>
    </div>

    <script type="text/javascript">
        // 3. The diagramming API is now accessible via global MindFusion object
        document.addEventListener("DOMContentLoaded", function () {
            console.log(`Your MindFusion.Diagramming version is:
${MindFusion.Diagramming.Diagram.version}`);
            var DiagramView = MindFusion.Diagramming.DiagramView;
            var diagramView = DiagramView.create(document.getElementById("diagram"));
        });
    </script>

</body>
</html>
```

Loading ES6 modules via importmap

Diagram's ES6 modules conform to modern JavaScript module definition that supports native **import** and **export** statements. You could import these modules directly, without using any additional module/package libraries or tools, by defining your script file as a module via <script type="module"> tag:

HTML

```
<script src="app.js" type="module"></script>
```

In order to make the diagram's ES6 modules available for importing, add the following **importmap**, referencing the scripts, contained in distribution's */esm* folder. The purpose of the **importmap** is to define a map of module names which allow using **import** specifiers such as import {x} from "@mindfusion/diagramming" (instead of a relative path to the corresponding script file), both internally and in in your code.

HTML

```
<script type="importmap">
{
    "imports":
    {
        "@mindfusion/collections":"./esm/collections.js",
        "@mindfusion/controls":"./esm/controls.js",
        "@mindfusion/drawing":"./esm/drawing.js",
        "@mindfusion/diagramming":"./esm/diagramming.js",
        "@mindfusion/graphs":"./esm/graphs.js",
        "@mindfusion/animations":"./esm/animations.js"
    }
}
</script>
```

Now you can consume the modules you need by importing them into your app.js:

JavaScript

```
import * as Diagramming from '@mindfusion/diagramming';
import * as Drawing from '@mindfusion/drawing';

var diagramView = Diagramming.DiagramView.create(
    document.getElementById("diagram"), diagram);
```

As of late 2024, **importmap** is now supported by all major modern browsers, including Chrome, Firefox, Edge, and Safari. For compatibility with older browsers, you can still use a polyfill like the https://github.com/guybedford/es-module-shims library.You can always check the latest support details at https://caniuse.com/import-maps.

Loading npm packages and bundling

There are six npm packages that provide the functionality of MindFusion.Diagramming, each containing the corresponding module. To use JsDiagram in a project, you only need to install the @mindfusion/diagramming package, the rest will be automatically added as dependencies:

npm install @mindfusion/diagramming

Each of the packages contains three distributions - ES module, CommonJs module and UMD, that you can use depending on your project setup.

To consume a module as a CommonJS module, use the 'require' statement. The CommonJs version of the module will be loaded from the /dist/cjs folder, as specified in the package.json's main field.

JavaScript

```
const Diagramming = require('@mindfusion/diagramming');
const { DiagramView } = require('@mindfusion/diagramming');
```

To consume a module as an ES6 module, use the 'import' statement. The ES6 version of the module will be loaded from the /dist/esm folder, as specified in the package.json's module field.

JavaScript

```
import * as Diagramming from '@mindfusion/diagramming';
import { DiagramView, Diagram } from '@mindfusion/diagramming';
```

A sample webpack bundle configuration is shown in Samples\MinApp\TypeScript\WebPack folder of distribution:

package.json

```
...
"dependencies": {
    "@mindfusion/diagramming":"^4.8.2"
},
...
```

Adding HTML Elements

Now that the library is loaded, you need to add the diagram component to your HTML page. There are two primary ways to do this:

1. The classic *<canvas>* element: This is a straightforward approach where you manually add an HTML *<canvas>* element to your page, wrapped inside a * * that handles scrolling. It's great for simple projects or when you want explicit control over the core HTML structure.

2. The Web Components method: This is a more modern, encapsulated approach. You use custom HTML tags (e.g. <mindfusion-diagramview>) and the library handles the internal setup for you. This is ideal for complex UIs or if you are building your application using a component-based architecture.

We will cover both methods below.

HTML elements

The DiagramView control relies on the HTML5 Canvas element to render graphics on web page. If you need to display large diagrams, place the Canvas inside a Div element.
Overflow and scrolling will be handled internally by **DiagramView**.

HTML

```
<div style="width: 800px; height: 600px;">
    <canvas id="diagram" width="2100" height="2100">
        This page requires a browser that supports HTML5 Canvas element.
    </canvas>
</div>
```

To create an instance of the **DiagramView** class, call its static create method, passing the DOM canvas object as argument:

JavaScript

```
var diagramView = DiagramView.create(
    document.getElementById("diagram"));
```

The MindFusion.Diagramming toolkit includes additional user interface controls such as Ruler, Overview, NodeListView and ZoomControl. If you'd like to use any of them, you will need to add Div and/or Canvas element for each one; for an example, see the *Controls* sample project.

Web components

Call WebComponents.register to register each control class from the Diagramming package as a web component. After registration, you can use the following tags to create corresponding components:

- <mindfusion-diagramview> creates a DiagramView instance.
- <mindfusion-ruler> creates a Ruler instance.
- <mindfusion-zoomcontrol> creates a ZoomControl instance.
- <mindfusion-overview> creates an Overview instance.
- <mindfusion-nodelistview> creates a NodeListView instance.

When instantiated as a web component, each control class creates required HTML elements as internal shadow DOM. You can get the JavaScript object corresponding to a web component by calling controls' find method with id argument.

HTML

```
<mindfusion-diagramview
    id="diagramView"
    style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: auto;">
</mindfusion-diagramview>
```

JavaScript

```
var diagramView = DiagramView.find("diagramView");
var diagram = diagramView.diagram;
```

This method is demonstrated in the *WebComponents.html* example from the library's distribution, where you can find a complete, runnable example.

Working with the JavaScript Diagram Model

With the library loaded and a view established on your page, you can now start working with the core JavaScript objects that define your diagram's structure and content.

This section introduces the two most important classes:

- Diagram: The data model that contains all your nodes, links, and their relationships.
- DiagramView: The visual component that renders the **Diagram** model onto the canvas and handles user interactions.

You will learn how to create a diagram instance and populate it with items, both programmatically and through direct user interaction.

Creating and displaying a Diagram

The Diagram is a model class, while rendering and interactions are handled by the DiagramView control. The **Diagram** instance can be created first and passed to the view as a second argument of the DiagramView.create method, or if not provided, an empty **Diagram** will be created and will be accessible through the view's diagram property.

JavaScript

```
var diagram = new Diagramming.Diagram();
var diagramView = Diagramming.DiagramView.create(
    document.getElementById("diagram"), diagram);

var diagramView = Diagramming.DiagramView.create(
    document.getElementById("diagram"));
var diagram = diagramView.diagram;
```

Add items from code

You can create diagram items programmatically by calling methods of the diagram's Factory object. Alternatively, create an instance of any class derived from DiagramItem by calling its constructor, and call addItem to add it to the diagram:

JavaScript

```
var node1 = diagram.factory.createShapeNode(0, 0, 10, 10);

var node2 = new Diagramming.ShapeNode(diagram);
node2.bounds = new Drawing.Rect(20, 0, 10, 10);
diagram.addItem(node2);
```

Draw items interactively

The DiagramView can respond in various way to user input depending on the value of its behavior property. By default, it creates a ShapeNode when the user starts drawing from unoccupied location, or a DiagramLink when the mouse pointer is over a node. See the Behavior enumeration for other supported modes.

Further reading

Browse the topics under Programming Interface Overview for conceptual information. See the tutorial topics and check the sample scripts provided with JsDiagram.zip distribution for sample source code. For integrating the diagram with popular front-end libraries and frameworks, see React, Vue.js and Angular topics.

TypeScript and Code Completion

TypeScript support

A d.ts declaration file is available for every package. For more information on declaration files see https://www.typescriptlang.org/docs/handbook/2/type-declarations.html#dts-files.

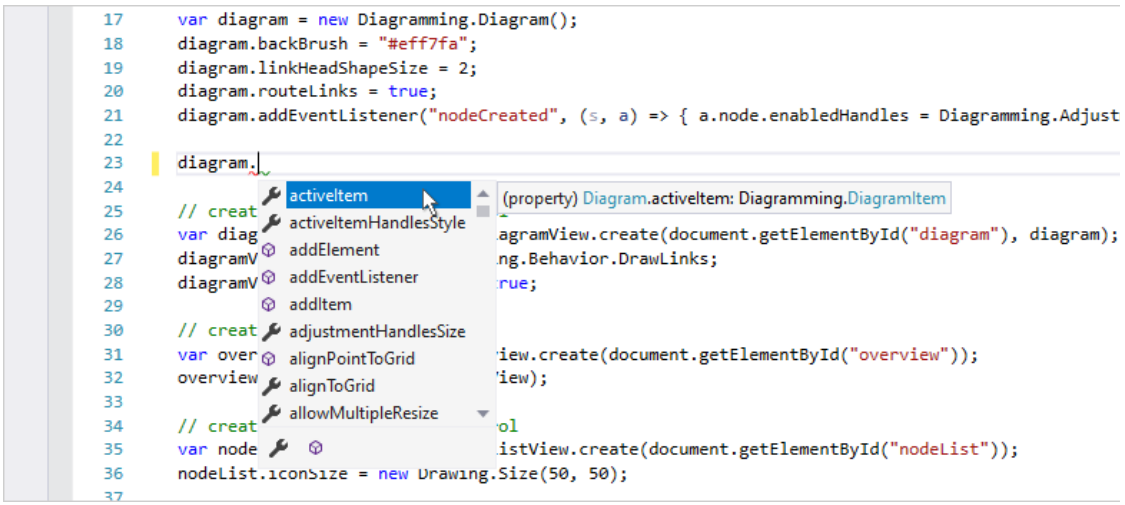
Set up code completion

The library provides code completion and hints through d.ts declaration files, supported by most popular IDEs. In order to enable the feature, you need to add a commented reference to the TypeScript *.d.ts files that are included in the library distribution. The most commonly used API members are in the diagramming file, but we advise that you add references to the auxiliary files as well:

JavaScript

```
/// <reference path="Scripts/diagramming.d.ts" />
/// <reference path="Scripts/animations.d.ts" />
/// <reference path="Scripts/collections.d.ts" />
/// <reference path="Scripts/controls.d.ts" />
/// <reference path="Scripts/drawing.d.ts" />
```

Once the references are loaded, you should be able to see code suggestions for the applicable members together with their type. All members should be visible: methods, properties, classes etc.



The distribution also includes JsDiagram-vsdoc.js file providing IntelliSense documentation in VSDoc format. If your programming environment supports it, you could add a reference to that file instead of d.ts ones:

JavaScript

```
/// <reference path="Scripts/JsDiagram-vsdoc.js" />
```

Starter Projects

Here are some starter projects you can clone from GitHub to get up and running quickly with different frameworks and environments. Each project provides a basic, pre-configured setup to demonstrate the integration of the diagramming library.

Vanilla JavaScript: A basic project for using the library with plain JavaScript, without any frameworks.
https://github.com/MindFusionComponents/diagram-starter-vanilla

NPM and a Bundler: Demonstrates how to install the library from npm and integrate it into a project that uses a module bundler (here we use Vite).
https://github.com/MindFusionComponents/diagram-starter-npm

Electron: A starter project for building cross-platform desktop applications.
https://github.com/MindFusionComponents/diagram-starter-electron

React: A starter for integrating the diagram library with the React framework.
https://github.com/MindFusionComponents/diagram-starter-react

React with Vite: A specific starter for React projects that use the Vite build tool.
https://github.com/MindFusionComponents/diagram-starter-react-vite

Angular: A starter for integrating the diagram library with the Angular framework.
https://github.com/MindFusionComponents/diagram-starter-angular

Vue.js: A starter for integrating the diagram library with the Vue.js framework.
https://github.com/MindFusionComponents/diagram-starter-vue

Socket.IO and Express server: This example instantiates Diagram model on server side.
https://github.com/MindFusionComponents/collaborative-mindmap-javascript

What's New in this Release

The list below describes recent changes and additions to JsDiagram:

New in version 4.9

CSS Styling

The styling system now supports defining item appearance through external CSS files. You can style different item types using custom element selectors such as **mfd-shapenode**, **mfd-diagramlink**, or **mfd-tablenode**. The system maps CSS Custom Properties to the internal Style properties. For example, use **--brush** for backgrounds, **--stroke** for outlines, and **--font-size** for text. The new cssClass property on DiagramItem allows for more granular styling using standard class selectors in your CSS. To ensure high performance, computed styles are cached globally, and the system automatically resolves style inheritance across class hierarchies (e.g., a ShapeNode will resolve styles from **mfd-shapenode**, **mfd-diagramnode**, and **mfd-diagramitem** selectors in that order).

CSS

```
/* Style all shape nodes */
mfd-shapenode {
    --brush: #f9f9f9;
    --stroke: #333;
    --stroke-thickness: 1px;
    --font-name: 'Verdana';
}
```

```
/* Specific styling via the new cssClass property */
mfd-shapenode.urgent {
    --brush: #ffcccc;
    --stroke: #ff0000;
    --stroke-thickness: 2px;
}

/* Style table captions using custom properties */
mfd-tablenode {
    --caption-brush: #e0e0e0;
    --text-color: #000066;
}
```

Compact Multi-graph Layout

A new MinimalArea option has been added to the **MultipleGraphsPlacement** enumeration, which uses a bin-packing algorithm to arrange connected components in a compact, square-like area. This ensures a more efficient use of space compared to horizontal or vertical arrangements when laying out multiple independent graphs.

Polygonal Text Layout

The ShapeNode class now supports a polygonalTextLayout property, enabling text to wrap and align precisely within the geometric boundaries of non-rectangular shapes like triangles, diamonds, and custom polygons. This feature supports both plain and styled text across all alignment modes.

Version History

The list below describes past changes and additions to JsDiagram:

New in version 4.8.3

- generateShortestRoute and findShortestRoute methods added to PathFinder.
- Table row anchorPattern fixes.
- Coordinates in vertical Ruler scale are now rotated in opposite direction to read from bottom to top, which is the accepted standard for rotated labels and consistent with MindFusion components for other platforms.

New in version 4.8.2

Improved BPMN shapes.

New in version 4.8.1

Spatial indexing fixes.

New in version 4.8

Paged containers

PagedContainerNode is a ContainerNode subclass that organizes its child nodes into a collection of pages, represented by ContainerPage objects. This allows for the creation of more organized diagrams where nodes can be grouped into logical views within a single container. Only the nodes belonging to the currently active page, specified by currentPage property, are rendered and participate in user interactions.

Navigation between pages is handled by scroll arrows in the caption bar, allowing users to cycle through the pages sequentially. For faster access, users can also double-click the container's caption to open an in-place combo box listing available page titles, enabling direct navigation to any page. The node's text value is automatically set to the title of newly activated **ContainerPage**.

Pattern router

New PatternRouter class routes links by applying patterns of segment and turn sequences and accepting them when resulting paths do not cross nodes. A path is accepted if its final segment ends in target point and is orthogonal to respective side of target node. If there are several possible paths, the router selects the shortest one. The minimumDistance property specifies minimum allowed distance of link segments from nodes. The preferredDistance property specifies distance of first or last link bend from associated node. **PatternRouter** includes several predefined patterns designed to create consistent-looking paths, and allows creation of custom patterns. Custom patterns can be defined as sequence of RouteStep objects specifying whether the path advances by relative or absolute distance, or changes direction:

JavaScript

```
var rightZigzag = new RoutePattern();
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.AdvanceRelative, 0.5));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.TurnRight));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.AdvanceRelative, 1));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.TurnLeft));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.AdvanceRelative, 0.5));
router.patterns.push(rightZigzag);

// this can be defined in shorter form using a pattern string
// var rightZigzag = new RoutePattern(
//     "ADR0.5 TRR ADR1 TRL ADR0.5");
```

Composite router

The CompositeRouter class lets you chain link routers (objects implementing the Router interface) so that a link passes through the sequence until it gets routed successfully. Diagram's default router is now set to a **CompositeRouter** instance containing a PatternRouter and **Router** sequence. This applies the several standard pattern paths when possible, and otherwise routes links using a simple cost-minimizing router.

API changes

Default Diagram.linkRouter changed from Router to CompositeRouter instance.

New in version 4.7.1

Grid layout

The GridLayout algorithm arranges diagram nodes in a grid, keeping connected nodes close to each other. The algorithm strives to achieve a small number of link crossings. It is based on an iterative process whose initial steps shuffle the grid nodes randomly. That can lead to very different results each time the algorithm is run.

Miscellaneous

- startNode and endNode properties added to LayeredLayout. They let you specify which nodes to place on first and last layer, instead of relying on the algorithm selecting them automatically.
- The crossingCost property of GridRouter implements penalty cost for link crossings. It's applied only when routing multiple links at once, e.g. when running routeAllLinks.
- The bringIntoView method scrolls the diagram view to make the specified item visible.
- The allowLinksRepeat property of Diagram specified whether more than one links can connect the same origin and destination diagram nodes.
- controlUnloading and controlUnloaded events let you handle the control being unloaded from page DOM, or disposed by wrapper components for supported frameworks (e.g. by **ngDestroy** hook in angular).

New in version 4.7

ItemListView component

ItemListView supersedes NodeListView and lets you show both nodes and links in the list. Items can be added by calling addItem method, and removed by calling removeItem. defaultNodeSize property from **NodeListView** is called newInstanceSize in the new class. The autoConnectLinks and autoConnectDistance properties control whether links created via drag-and-drop should connect to nearby nodes in the target diagram. **ItemListView** also provides more properties for customizing the list layout, such as columns and labelPosition. For compatibility with legacy code, **NodeListView** won't be removed from the library, but it won't be getting any new improvements.

Palette component

The Palette control represents a tool palette that displays DiagramItem objects grouped into categories. It implements Accordion user interface that displays ItemListView components as its child panels, and category icon, title and collapse/expand button in pane headers. Each child **ItemListView** displays the items added to its corresponding category. Call the addCategory method to define a category and create its respective accordion pane. Call addItem to add an item to specified category. Palette provides same layout and appearance properties as the **ItemListView** class.

ItemLabel image

New image and imageLocation properties added to ItemLabel let you display icons along a link's geometry, or add more images to a ShapeNode. If a label contains both text and **image**, their relative position is specified by imageAlign property (by default the image is placed on left side of text), and distance by contentPadding. The image is rendered using bitmap's intrinsic size, unless you override it by setting imageSize property. You can treat label images as clickable icons by handling linkClicked or nodeClicked events and checking label argument passed to their handlers.

Miscellaneous

- locked property added to ItemLabel. It lets you prevent users from editing label's text, and moving it when behavior is set to MoveLabels.
- label argument now provided with item click and double click events.
- Improved repaint speed of Overview component.

New in version 4.6

Spatial index

Set the enableSpatialIndex property of Diagram to create an index of item positions for faster hit-testing and viewport clipping queries. This should greatly improve user interaction and rendering speed for diagrams containing tens of thousands or more items.

JavaScript

```
// create 20000 links + nodes in total
for (var i = 0; i < 10000; i++)
{
    var node = diagram.factory.createShapeNode(x, y, size, size);
    node.text = i;
    if (x > diagram.bounds.x)
    {
        diagram.factory.createDiagramLink(
            diagram.nodes[i - 1], diagram.nodes[i]);
    }

    x += dist;
    if (x >= diagram.bounds.right())
    {
        x = diagram.bounds.x;
        y += dist;
```

```
    }
}

diagram.enableSpatialIndex = true;
```

Miscellaneous

- The foldedSize property of ContainerNode specifies the container's size when it is folded.
- clipText property added to ShapeNode.
- actionUndone and actionRedone events raised by undo manager for either top-level commands or children of CompositeCommand.
- Fixed multiple resize when selection contains rotated nodes.
- Fixed rotation when rotation handle is at a custom position (set from hitTestAdjustmentHandle event handler).
- strokeThickness is now scaled by zoomFactor.

New in version 4.5.1

Node constraints

The constraints property lets you apply size and direction constraints to a node. It returns a NodeConstraints object with following properties:

- keepRatio: a bool value indicating whether the initial width/height ratio of a node should be preserved.
- maxHeight: the maximum node height allowed.
- maxWidth: the maximum node width allowed.
- minHeight: the minimum node height allowed.
- minWidth: the minimum node width allowed.
- moveDirection: a member of the DirectionConstraint enumeration. Set it to **Horizontal** or **Vertical** to let users move a node only in specified direction.

These constraints are considered when a user modifies the node interactively. They are ignored if node placement is changed programmatically by using API methods and properties.

Miscellaneous

- TabbedDiagramView now keeps separate zoom level and scroll position per tab.
- shapeLibraryLoaded event raised when library XML file is loaded and its shape definitions are available for use.
- fixed undo / redo of text in-place edit operations.
- fix for item hover events of DiagramDocument not raised by **TabbedDiagramView**.

New in version 4.5

Shape design enhancements

- The anchorPattern property of Shape lets you associate anchor points with shape definitions. That pattern is shared between all nodes of same shape, but can be overridden by setting node's own anchorPattern.
- Individual segments of a shape's geometry can be styled independently by setting stroke and fill attributes of ShapeElement and ShapeDecoration objects.
- We have replaced the Windows -only shape designer tool with an online shape editor at https://mindfusion.dev/tools/shape-designer.html
- The shape editor is implemented as a reusable ShapeDesigner control, which can be integrated into your own applications.

New event system

Event emitter objects are now exposed as properties of Diagram and DiagramView, and you can register handlers by calling their addEventListener method:

JavaScript

```
diagram.nodeCreated.addEventListener(
    (sender, args) =>
    {
        if (sender.defaultShape.id == "Rectangle")
            args.node.anchorPattern = pattern1;
        else
            args.node.anchorPattern = pattern2;
    });
```

Legacy event handling syntax will remain supported for compatibility with old code.

Collapse and expand table rows

Table rows can be assigned to distinct sections of the table. Each section can be collapsed or expanded, hiding or showing all rows in the section except the header one. To define a section, set the header property of a row. A section consists of all rows after a header row and spans to the next header. Each header row displays a [±] button that allows expanding or collapsing the section interactively. Clicking that button raises the tableSectionCollapsed and tableSectionExpanded events. Sections can be collapsed or expanded programmatically by setting the expanded property of their header rows.

Miscellaneous

- dividerStroke properties added to ContainerNode, TableNode and TreeViewNode classes. It lets you customize the appearance of the divider line drawn between a node's caption bar and content area.
- removed an **eval** call used for creating custom components from CompositeNode Json templates, allowing for stricter Content Security Policy rules. Now custom components must be registered by calling registerComponent method to allow instantiating them from a template.
- linkRouted event raised when links are modified by automatic routing.

API changes

To allow for stricter Content Security Policy rules, the library no longer loads CSS scripts automatically. If you use components from the Diagramming.Controls package, you must now explicitly link the common-ui.css file from distribution's *themes* folder.

New in version 4.4.4

New mouse events

- **nodePointed** event has been replaced by a set of more specific events: nodePointerEnter and nodePointerLeave are raised when the mouse pointer respectively enters or leaves the boundaries of a node. nodeHovered fires if the pointer rests over a node for hoverDelay milliseconds.
- **linkPointed** event has been replaced by a set of more specific events: linkPointerEnter and linkPointerLeave are raised when the mouse pointer respectively enters or leaves the boundaries of a link. linkHovered fires if the pointer rests over a link for **hoverDelay** milliseconds.

Miscellaneous

- Enable the autoDeleteChildren property to automatically delete child nodes when a container is deleted.
- Fix for custom Shape definitions not setting the **id** property.
- Fix for selected property setter of DiagramItem not immediately reflecting assigned value due to async calls.

API changes

- **linkPointed** and **nodePointed** events replaced by linkHovered and nodeHovered.

New in version 4.4.3

- Due to problems with some script loaders, the library no longer registers web components automatically. Call the WebComponents.register method to explicitly register component classes.
- Set cornerRadius to change the size of corners when node's shape is set to a rounded rectangle.
- Set the autoDeleteLinks property to **false** to prevent deleting links along with a node they connect, and disconnect the links instead.
- Fixed a bug where child nodes keep reference to their deleted container.

New in version 4.4.2

TableNode improvements

- getRow and getColumn methods now return strongly-typed TableRow and TableColumn objects.
- The anchorPattern property of **TableRow** and rowAnchorPattern property of TableNode let you specify the points where links connect to table rows.
- cellTextColor property lets you set default color of cell texts, without changing caption color.

TabbedDiagramView improvements

- Enable allowRenamePages to let users edit the title of DiagramPage by clicking the active tab.
- The control raises pageAdding event when users click the plus icon to let you validate the operation, and pageAdded once the page is added.
- The control raises pageRemoving event when users close a tab to let you validate the operation, and pageRemoved once the page is removed.
- Fix for scroll range not updating when changing tabs.

Miscellaneous

- Disable allowUnanchoredLinks to prevent users from drawing links to nodes without designated anchor points.
- Set inplaceEditAcceptOnEnter and inplaceEditCancelOnEsc properties to specify whether edit operations end when user presses respectively *Enter* and *Esc* keys.
- The showAnchors property can be set to Selected | Auto combination.
- Set horizontalDropAlignment and verticalDropAlignment properties to specify how nodes are positioned relatively to the drop point after drag operations from NodeListView.
- Set dragIndicatorOpacity property to specify the opacity of drag indicator displayed during drag-and-drop operations.

New in version 4.4.1

- Angular bindings updated to Angular version 16.
- Custom-draw anchor points by setting MarkStyle.Custom and handling drawAnchorPoint event.
- SinglePointerController can now be inherited from TypeScript.
- Fix for allowUnconnectedLinks not working when allowSelfLoops is disabled.
- Set validityChecks to **false** to add items faster.
- Fix for ControlNodes drawing on top of DiagramView's scrollbars.
- Fixed XML serialization of unconnected links.
- Fixed exception when moving multiple selection in TabbedDiagramView.
- Fixed NodeListView exception for nodes with rotation handle.
- Blazor bindings moved to MindFusion.Diagramming.JavaScript nuget package.

New in version 4.4

Web components

The library registers each control class as a web component. You can use the following tags to create corresponding components:

- <mindfusion-diagramview> creates a DiagramView instance.
- <mindfusion-ruler> creates a Ruler instance.
- <mindfusion-zoomcontrol> creates a ZoomControl instance.

- <mindfusion-overview> creates an Overview instance.
- <mindfusion-nodelistview> creates a NodeListView instance.

When instantiated as a web component, each control class creates required HTML elements as internal shadow DOM. You can get the JavaScript object corresponding to a web component by calling controls' find method with id argument.

HTML

```
<mindfusion-diagramview
    id="diagramView"
    style="position: absolute; left: 0px; top: 0px; right: 0px; bottom: 0px; overflow: auto;">
</mindfusion-diagramview>
```

JavaScript

```
var diagramView = DiagramView.find("diagramView");
var diagram = diagramView.diagram;
```

Animations

The library now includes several standard animation classes and methods.

The animate method of DiagramItem can run following animations:

- FillColorAnimation animates the fill color of nodes or link arrowheads.
- StrokeColorAnimation animates the color of node borders or link segments.
- OpacityAnimation animates items' opacity for fade-in and fade-out effects.

Following animations can be applied to DiagramNodes:

- BounceAnimation animates vertical position of nodes.
- ShakeAnimation animates horizontal position of nodes.
- PulseAnimation animates node's local scale factor.
- PathAnimation moves nodes along specified path.

The animate method of DiagramView can run following animations:

- ScrollAnimation animates the scroll position of a view.
- ZoomAnimation animates the zoom factor of a view.

DiagramLink improvements

- Set the intermediateShape property to display arrowheads at the middle of each link segment. This could help users follow longer link paths. Intermediate arrowheads can be customized further by setting intermediateBrush and intermediateShapeSize.
- Set the allowSplitLinks property of Diagram to let users insert new link segments by dragging any point of a selected link's segment. In this mode users can also merge link segments by making them close to collinear, or by moving adjustment handles close to each other, as set via mergeThreshold.

New in version 4.3.2

Reversed tree layouts

The reversedLinks property of TreeLayout, RadialTreeLayout and BorderedTreeLayout lets you arrange trees where links point from child to parent nodes, instead of the default parent to child direction.

Miscellaneous

- column, row and node property getters added to CellEventArgs.
- fixed RaiseEvents mode of expandButtonAction; it no longer collapses or expands clicked node's subtree.
- ContainerNode now sorts its child nodes by zIndex instead of insertion order.

New in version 4.3.1

TabbedDiagramView wrapper components now available in React, Angular, Vue and Blazor packages.

New in version 4.3

Multiple diagram pages

The DiagramDocument class represents a collection of diagram pages or sheets, represented by DiagramPage objects. New pages can be added to the document and existing pages can be removed and reordered by using the addPage and removePage methods respectively. The toJson, saveToXml and saveToString methods of **DiagramDocument** save all pages in a single file, and respectively the fromJson, loadFromXml and loadFromString methods load all pages from the file. **DiagramDocument** can also load files created by the serialization methods of Diagram, and will show them in a single page.

DiagramPage is derived from **Diagram** and only adds a few properties, so any code that processes **Diagram** objects will work with **DiagramPage** objects too. A diagram document forwards each event raised by its pages, giving you a chance to use the same handler for events raised by different pages. If a page should implement distinct user interaction or validation logic, it is also possible to attach handlers to the **Diagram** events inherited by **DiagramPage**.

A single **DiagramPage** could be assigned to the diagram property of DiagramView, and that lets you create your own user interface for presenting a list of pages to the user and selecting a current page. The package also includes a TabbedDiagramView that presents all pages of a document in a tabbed user interface where each tab corresponds to a page. The page displayed in the active tab is exposed by the selectedPage property of **TabbedDiagramView**. The active page can be changed interactively by activating its associated tab. New pages can be added and removed to / from the document through buttons in the tab strip and the pages can be rearranged by dragging their associated tabs with the mouse.

Tabbed diagram view

The TabbedDiagramView control is a view that displays DiagramDocument objects. Individual diagram pages can be activated through the tabs located at the top or at the bottom of the control. The appearance and behavior of the tab strip can be customized using various properties. The location and visibility of the strip is specified through the tabAlignment property. Its height can be changed via the tabsSize property. showAddButton and showNavigationButtons determine the visibility of the buttons for interactive tab creation and tab navigation respectively. showTabCloseButtons specifies whether the tabs can be closed interactively, by clicking on the small 'x' button displayed in each tab. allowReorder enables interactive tab rearrangement by mouse dragging.

TabbedDiagramView comes as s a part of the diagramming-controls npm package or the MindFusion.Diagramming.Controls namespace for UMD users.

For npm users simply import the diagramming-controls package:

JavaScript

```
import * as Controls from '@mindfusion/diagramming-controls';
```

For UMD users, add references to files from distrbution's /umd folder using <script> tags. Note that the order of the scripts matters.

HTML

```
<!-- core Diagramming -->
<script src="Scripts/collections.js" type="text/javascript"></script>
<script src="Scripts/drawing.js" type="text/javascript"></script>
<script src="Scripts/controls.js" type="text/javascript"></script>
<script src="Scripts/graphs.js" type="text/javascript"></script>
<script src="Scripts/animations.js" type="text/javascript"></script>
<script src="Scripts/diagramming.js" type="text/javascript"></script>

<!-- for TabbedDiagramView -->
<script src="Scripts/common.js" type="text/javascript"></script>
<script src="Scripts/common-collections.js" type="text/javascript"></script>
<script src="Scripts/common-ui.js" type="text/javascript"></script>
<script src="Scripts/diagramming-controls.js" type="text/javascript"></script>
```

Creating the control instance and adding it to an HTML page is the same as for the DiagramView control. You can pass a **DiagramDocument** instance to the create function, or omit that parameter to let the control create a new document with a single DiagramPage.

JavaScript

```
var doc = new MindFusion.Diagramming.DiagramDocument();
var page1 = new MindFusion.Diagramming.DiagramPage("diagram1");
var node = page1.factory.createShapeNode(10, 10, 20, 20);
doc.addPage(page1);
var diagramView = MindFusion.Diagramming.Controls.TabbedDiagramView.create(
    document.getElementById("diagram"), doc);
```

or simply

JavaScript

```
var diagramView = MindFusion.Diagramming.Controls.TabbedDiagramView.create(
    document.getElementById("diagram"));
```

You can specify a title, iconUrl and titleColor for diagram pages by setting the corresponding properties.

Default images for the navigation buttons must be located in an ./icons folder. Alternatively, you can style the control with a predefined css theme, in which case the themes included in the distribution must be located in a ./themes folder and referenced in the head of the HTML document, e.g.

HTML / JavaScript

```
<link rel="stylesheet" type="text/css" href="themes/business.css" />
...
diagramView.theme = MindFusion.Diagramming.Theme.fromId("business");
```

New in version 4.2.5

Blazor bindings

MindFusion.Diagramming for Blazor allows integrating the JavaScript diagramming API into Blazor applications. It contains a set of .NET wrapper classes, that use Blazor's JSInterop API to create and manipulate the client-side objects. The package provides easy access from C# code to the most of the MindFusion.Diagramming functionality, including different node types, layouts and styling. Additional UI components such as Overview, NodeListView, ZoomControl and Ruler are included too.

Miscellaneous

- The opacity property lets you create translucent DiagramItem by applying it to the alpha channel of all colors in the item.
- Diagram's background image now scales along with zoomFactor.
- The validateAnchorPoint event validates connecting diagram links to specific anchor points.
- The validateLinkCrossing event validates visibility of individual link crossing point decorations.

Fixed bugs

- Fix for mouse wheel not scrolling when virtualScroll is disabled.
- Touch input fixes.
- Fix for not raising **nodePointed** and **linkPointed** events.
- Fixed strokeDashStyle XML serialization.

New in version 4.2.4

Lasso zoom tool

The control now supports several ways to zoom using lasso tool:

- Set the behavior property to Zoom to always draw zoom lasso.
- Set the Zoom flag for one of the keys in modifierKeyActions to let users draw zoom-lasso with respective modifier key pressed and the left mouse button.
- Assign Zoom value to rightButtonActions or middleButtonActions property to zoom using respective mouse button.

Miscellaneous

- The PanAndModify behavior lets you select and modify when the mouse pointer is over an item, or pan the view otherwise.
- removeNode method added to NodeListView.
- Fixed clearAll method of **NodeListView** keeping visual elements from old nodes.
- strokeThickness now affects dash distances.
- Fixed styled text rendering in rotated NodeLabel, TableNode and ContainerNode objects.
- **strokeThickness** no longer affects text rendering.

New in version 4.2.3

Radial tree layout

The RadialTreeLayout class arranges tree levels in concentric circles around the root. The levelDistance property specifies the distance between adjacent circles. The direction property specifies placement of first child node. Set stretchFactor to a value larger than 1 in order to stretch level circles into ellipses, providing more space for the layout when using wider nodes.

Miscellaneous

- strokeDashArray and strokeDashOffset properties let you set custom dash patterns when strokeDashStyle is set to Custom.
- Styled text now supports <size=N> element letting you set font size.
- enableStyledText property added to ItemLabel lets you format labels using HTML-like format flags.
- enableWrap and maxWidth properties of **ItemLabel** let you wrap label's text.
- The ArrowHeads class exposes arrowhead shapes as static methods returning respective predefined shape.
- Fixed linkTextStyle exception.

New in version 4.2.2

React functional components

The @mindfusion/diagramming-react package now contains functional-component wrappers for the DiagramView and auxiliary controls, and has been upgraded to React 18. Old class-component wrappers have been moved to the @mindfusion/diagramming-react-cc package.

Using the functional components looks like this:

JSX

```
const [diagram] = useState(new Diagramming.Diagram());
...
<DiagramView diagram={diagram} {...props}
    onDiagramChanged={(diagram, args) => onDiagramChanged(diagram, args)}
    onNodeCreated={(diagram, args) => onDiagramNodeCreated(diagram, args)} />
```

The **DiagramView** control exposes a **forwardRef**, that can be passed on to other controls, such as the Overview and ZoomControl. To obtain a reference to the underlying core controls, use the respective find method of the **ref**.

Miscellaneous

- Fix for NodeListView not working correctly when hosted nodes contain NodeLabel objects.

- Fixed exception in dispose method of Overview control if called before the overview is attached to the diagram.

New in version 4.2.1

Fixed LinkLabel deserialization from old file formats when using along-length positioning.

New in version 4.2

Multiple labels per node

The NodeLabel class allows multiple captions to be displayed for a single DiagramNode of any type. Node labels provide a set of properties allowing full customization of their display and positioning. Label position is defined by specifying a pin point and offset from it, set through setCornerPosition, setEdgePosition, setCenterPosition methods. In addition, the horizontalAlign and verticalAlign properties of **NodeLabel** specify on which side of pin point to draw the label's caption.

For example, following code adds extra labels to top-left and bottom-right corners of a ShapeNode:

JavaScript

```
var node = diagram.factory.createShapeNode(10, 50, 40, 30);
node.text= "text"; // centered main text

var lb1 = node.addLabel("label 1");
lb1.setCornerPosition(0, 0, 0);
lb1.horizontalAlign = Alignment.Near;
lb1.verticalAlign = Alignment.Near;
lb1.brush = "red";
lb1.textColor = "white";

var lb2 = node.addLabel("label 2");
lb2.setCornerPosition(2, 0, 0);
lb2.horizontalAlign = Alignment.Far;
lb2.verticalAlign = Alignment.Far;
lb2.brush = "yellow";
lb2.textColor = "red";
```

Multi-touch improvements

- The multiTouchDraw property of DiagramView lets you prevent drawing multiple items simultaneously, while keeping other multitouch gestures enabled.
- The handleTouchHitDistance property of **DiagramView** makes it easier to grab adjustment handles on mobile devices, without increasing the adjustmentHandlesSize value.

Miscellaneous

- The MoveLabels behavior lets user move link and node labels interactively.
- LinkEventArgs now includes origin and destination properties that report respective candidate nodes to linkCreating and linkModifying event handlers.
- The attach method of DiagramNode now automatically detaches from previous master.
- contentAlign property of SvgNode specifies alignment of SVG drawing inside the node.
- lineAlignment property of DiagramLink sets the vertical position of link's text.
- Horizontal scroll gestures should now work on Macbook trackpad when using virtual scroll mode.

Bug fixes

- DiagramLink.baseShape serialization fix.
- Fix for createDiagramLink not working with Point arguments (to create unconnected links).
- Fix for link adjustment handles not applying stroke from HandlesVisualStyle.

New in version 4.1

Multi-touch support

The control now handles DOM Pointer events to implement multi-touch interactions, such as zooming, node rotation or simultaneous drawing of multiple diagram items:

- If multiTouchZoom property is enabled (default), the view can be zoomed or panned using two-touch pinch / flick gestures.
- If multiTouchModify property is enabled (default), diagram nodes can be moved, scaled and rotated using two-touch pinch / flick gestures.
- If **multiTouchZoom** property is disabled, each touch draws diagram items corresponding to current behavior.
- If **multiTouchModify** property is disabled, each touch started from a node draws a diagram link.

Latter modes can be used for collaborative whiteboarding / classroom scenarios.

Built-in multi-touch can be disabled altogether by setting enableMultiTouch to **false**, e.g. if you need to handle touch events yourself or through a third-party gesture library. If disabled, the control will revert to its mouse event handling code from previous versions. It will also no longer be able to capture mouse input during drag operations (to detect mouse pointer leaving the view), which is handled through DOM Pointers API.

Miscellaneous

- The createDiagramLink method can now be used to connect to TableNode rows or TreeViewNode items.
- cancelDrag method added to DiagramView.
- The control now captures mouse input during drag operations and will not lose current drag state if mouse pointer leaves the view.
- Fixed exceptions when linkCrossing is set to Arcs.
- Fix for containers clipping their child nodes when children are being dragged.

New in version 4.0.1

Updated framework wrappers

- React diagramming library has been updated for version 4 API. In addition, it has been moved to scoped *@mindfusion/diagramming-react* package on NPM, and now requires React 17.0.2 as minimum version.
- Vue diagramming library has been updated for version 4 API. In addition, it has been moved to scoped *@mindfusion/diagramming-vue* package on NPM, and now requires Vue.js 3.0.0 as minimum version.
- Angular diagramming library has been updated for version 4 API. In addition, it has been moved to scoped *@mindfusion/diagramming-angular* package on NPM, and now requires Angular 12.2.0 as minimum version.

API changes

Angular component's event emitter names no longer have an 'on' prefix. E.g. use (nodeCreated) instead of (onNodeCreated) to bind to the nodeCreated event.

New in version 4

ES6 modules, classes and properties

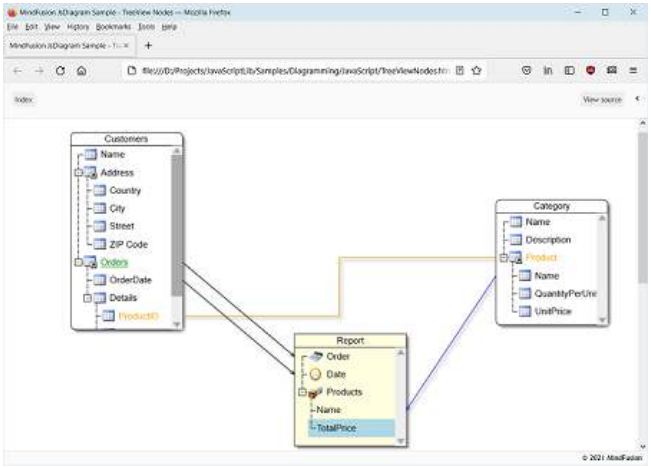
JsDiagram source code has been refactored following ES6 standards, including modules, classes, properties and native collections. **ArrayList**, **Dictionary** and **Set** classes from MindFusion.Collections namespace have been removed, and replaced by respective JavaScript native **Array**, **Map** and **Set**. Get/set functions have been replaced by getter/setter properties, which should help using the diagram API with binding expressions of various JS frameworks. You could enable propFunctions in CompatConfig in order to continue using legacy get/set functions during migration. The distribution still includes ES5-compatible scripts located in *Scripts/umd* folder, which are transpiled from current ES6 code, and whose classes are exposed as members of global MindFusion namespace object as in previous versions. The *Scripts/esm* folder contains ES6 code that lets you import JsDiagram classes from respective modules.

Diagram view

The Diagram control has been refactored into two classes. What remains of **Diagram** is now just a model class that defines diagram structure and implements serialization. The new DiagramView class deals with rendering and user interaction. You can show the same diagram in several views, each one rendering at different scroll positions and zoom levels. The separation also makes it easier to use the diagramming API in node.js server code, avoiding the need to load DOM shim/polyfill packages.

TreeView nodes

The TreeViewNode class represents nodes that can display hierarchical data. The root items displayed in the node can be accessed through the rootItems property. Items can be added and removed individually by using the addItem and removeItem methods, or in bulk by calling the fromObject method, which loads the tree view items from an array of objects.

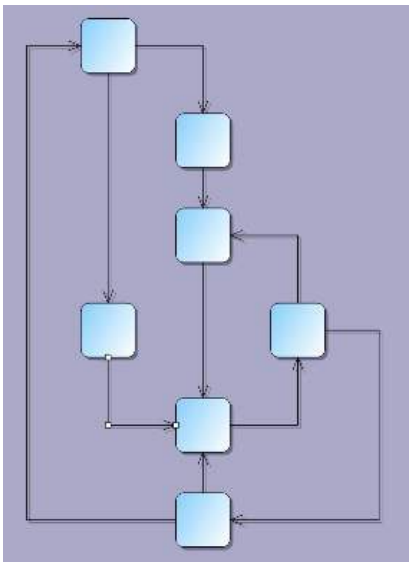


Print pagination

The printPreview and print methods of DiagramView let you export the diagram as a list of smaller images in HTML page. Supported options include **printArea** (defaults to diagram's content bounds) and pageSize (defaults to DiagramView's viewport). Note that print methods use HTMLCanvasElement.toDataURL() internally, so any limitations it has will apply (such as canvas size and CORS).

Orthogonal layout

The OrthogonalLayout class implements an orthogonal graph layout algorithm. Each link is drawn as a chain of alternating horizontal and vertical segments. Nodes are placed in a way that facilitates few links bends and crossings. This algorithm was designed for planar graphs where the nodes have at most four incident links, and produces best results with such graphs as input. It can arrange any arbitrary graph as well, by adding some dummy nodes and links to planarize the graph and reduce its vertex degree, and removing them after the layout.



New events

- nodeSelecting, linkSelecting, selectionChanged events allow handling of selection interactions.
- dataLoaded is raised when the diagram data is loaded from JSON or XML.
- mouseWheel is raised when the user rotates the mouse wheel while the cursor is positioned over the diagram surface.
- treeItemTextEditing, treeItemTextEdited allow handling of inplace-edit operations in TreeViewNode.

Miscellaneous

- Clipboard methods now use modern **Navigator.clipboard** API when their *systemClipboard* argument is enabled.
- Rotation of FreeFormNode instances.
- ImageAlign supports new **FitLeft**, **FitTop**, **FitRight** and **FitBottom** alignment styles, which resize image to fit node's boundaries and align it to respective border.
- mouseWheelAction property of DiagramView lets you choose between Scroll (default) and **Zoom** as the default behavior of the control in response of a wheel event.

API changes

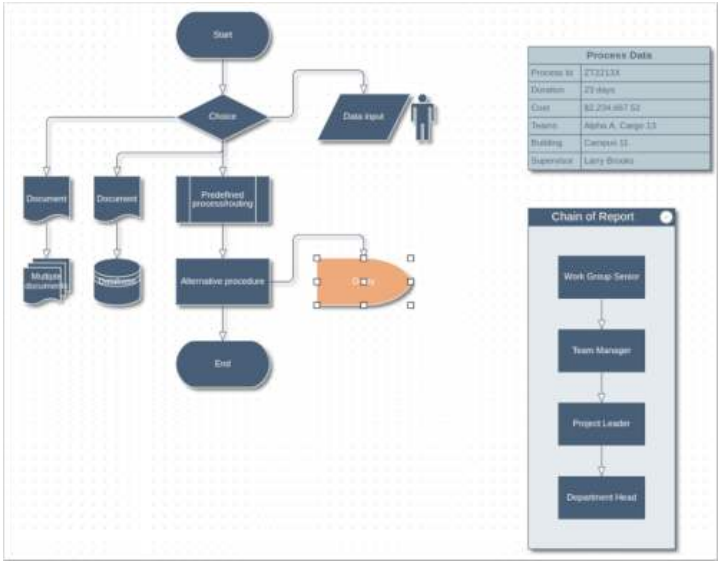
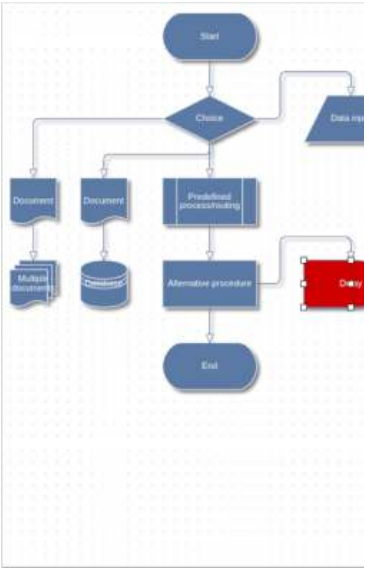
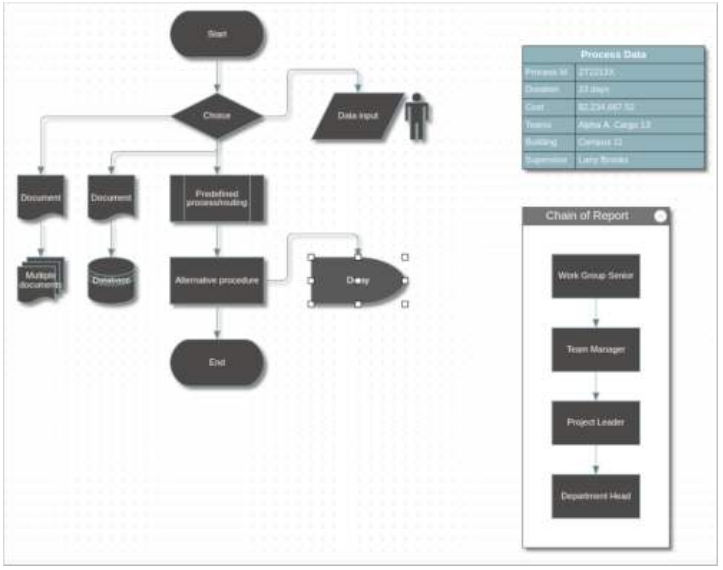
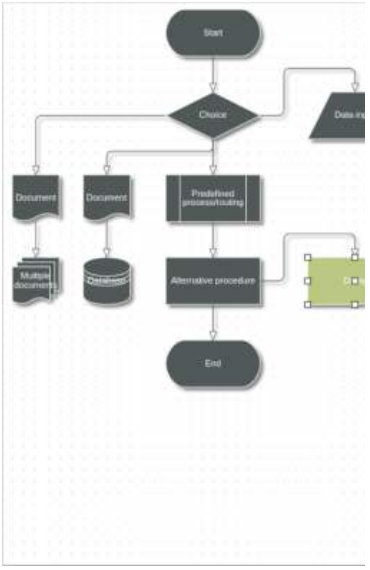
- The Diagram class is now purely a model class and no longer represents a DOM control. To display and allow interactions with a **Diagram** instance, you will need to create a DiagramView control and pass the **Diagram** instance to its diagram property.
- Auxiliary controls, such as Overview, Ruler and ZoomControl, must be attached to a **DiagramView** instance instead of **Diagram** via the diagramView / target properties.
- Most getProperty/setProperty function pairs have been replaced by ES6 getters/setters. Set CompatConfig.propFunctions to **true** in order to continue using legacy get/set functions (for properties added to the library before v.4.0) in your code.
- scroll and zoom properties moved from **Diagram** to **DiagramView**: autoScroll, scrollZoneSize, autoScrollAmount, scrollX, scrollY, viewport, virtualScroll, zoomFactor.
- input related properties moved from **Diagram** to **DiagramView**: allowInplaceEdit, delKeyAction, behavior, tooltipDelay, modificationStart, modifierKeyActions, leftButtonActions, middleButtonActions, rightButtonActions.
- magnifier properties moved from **Diagram** to **DiagramView**: magnifierEnabled, magnifierFactor, magnifierWidth, magnifierHeight, magnifierFrameThickness, magnifierShading, magnifierShape,magnifierFrameColor, magnifierSecondaryFrameColor.
- other properties moved from **Diagram** to **DiagramView**: defaultControlTemplate, **licenseKey**, licenseLocation.
- the following methods have been moved from **Diagram** class to **DiagramView** class: create, find, record, stopRecording, replay, clearTooltip, beginEdit, copyToClipboard, pasteFromClipboard, cutToClipboard, scrollTo, zoomToRect, zoomToFit, setZoomFactorPivot.
- **MindFusion.AbstractionLayer** class has been removed.

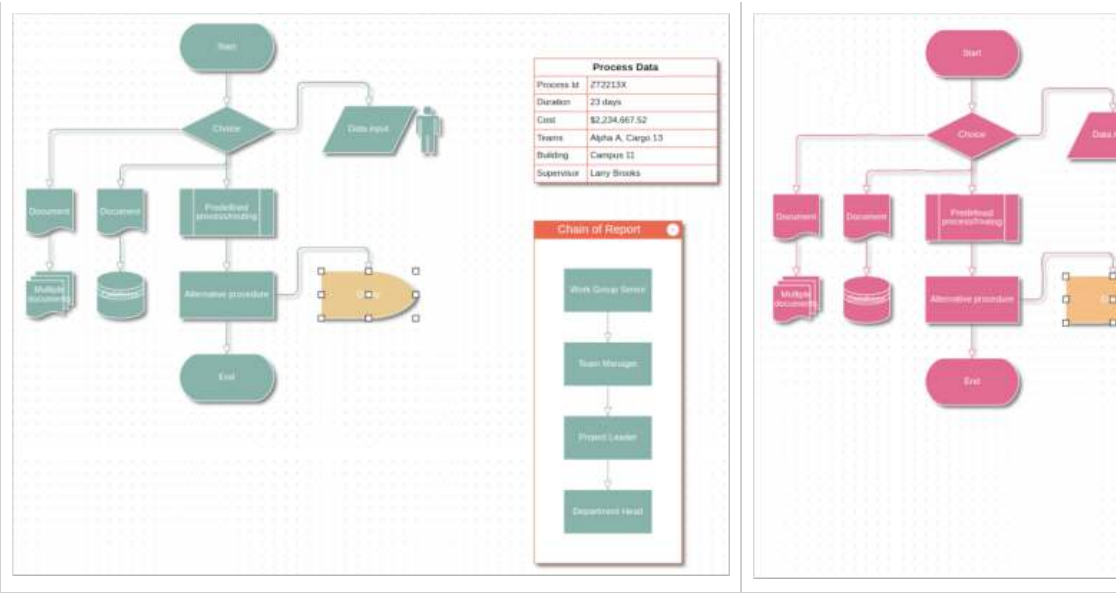
- EventArgs and CancelEventArgs classes have been moved to the MindFusion.Controls namespace.
- **ArrayList**, **Dictionary**, **ObservableCollection**, **Set** classes have been removed from MindFusion.Collections namespace.
- Due to **Navigator.clipboard** API being asynchronous, pasted items are not available immediately after calling **pasteFromClipboard** with *systemClipboard* enabled. You will be able to access them only after nodePasted and linkPasted events are raised.

New in version 3.5.4

Stock themes

The library now includes nine stock themes, accessible via respective static methods of the Theme class. Call the setTheme method of Diagram to apply a theme.

Blue	Business
	
Gray	Green
	
Pastel	Peach



Miscellaneous

- The MoveNodes behavior allows grabbing nodes to drag them without using adjustment handles.
- New SelectedBrush and SelectedStroke properties of Style used to draw selected items.
- CaptionBackBrush of ContainerNode and TableNode objects can now be set via **Style** instead of nodes' own properties.
- Fix for resizeToFitText not calculating correct size when cells have text padding applied.
- BackColor and TextColor properties added to NodeListView.
- ToolTip property of AnchorPoint shows tooltip text when mouse hovers over the point.
- New *Scripting.js* sample project demonstrates ControlNode objects and graph traversal.

New in version 3.5.3

Adjustment handles styling

Appearance of adjustment handles can be customized via ActiveItemHandlesStyle, SelectedItemHandlesStyle and DisabledHandlesStyle properties. The HandlesVisualStyle objects returned by them provide sub-properties corresponding to graphic attributes of the different handle types. Adjustment handles can now be painted not only in solid color but with arbitrary brushes such as gradients and patterns.

Miscellaneous

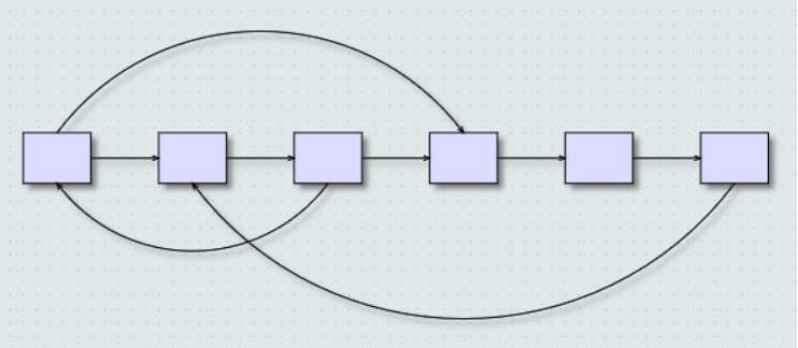
- nodeTextEditing, linkTextEditing and cellTextEditing validation events let you prevent users from editing a specific item.
- XML serialization fixes.
- Fixed rendering of joints for links with larger strokeThickness and more than one segment.
- ActiveItem reference is now serialized in JSON format.
- Fix for scroll position serialization in virtual scroll mode.

New in version 3.5.2

Topological Layout

TopologicalLayout applies graph topological ordering to the diagram (topological ordering of a graph is one where link's origin node is placed before link's destination node). The layout algorithm arranges nodes in a row or a column, depending on the value of the direction property, in such a way that there are no backward links when the graph is acyclic. If the graph contains cycles, the algorithms selects ordering with as few backward links as possible. Links that connect non-adjacent nodes are rendered as arcs, whose amplitude is proportional to the distance between respective connected nodes. Forward links are

rendered at one side of the nodes and back links are rendered at the opposite side. All that makes it easy to discern graph features such as overall flow direction, cycles and nested cycles.



Miscellaneous

- The diagram canvas now automatically expands to fill the area freed up when a scrollbar auto-hides.
- Fixed Overview flickering when the tracker rectangle is being dragged near the control's borders.

New in version 3.5.1

Virtual scroll improvements

- The control now handles mouse wheel events to scroll the diagram when virtual scrolling is enabled.
- DOM elements that implement virtual scroll mode now use CSS grid layout. You might need to adjust page layout to apply size constraints on the diagram div or its parent. For compatibility with older browsers, you can set CompatConfig.gridLayout = **false** to fall back to the absolute positioning from previous versions.
- VirtualScroll mode is now enabled by default.

Electron compatibility

Serialization and clipboard operations should now work under Electron.js. In previous versions they would throw exceptions due to restricted use of **eval**.

Miscellaneous

- Mouse pointer position is now provided as argument to nodeCreated and linkCreated events.
- Visibility property added to CompositeNode components lets you keep invisible components in the template.
- General performance improvements.
- Collapse / expand icon is now drawn correctly for **CompositeNode** when Expandable is enabled.
- Json deserialization no longer creates undo records.

API changes

For compatibility with MindFusion diagramming API for other platforms, StrokeThickness is now specified in diagram's MeasureUnit instead of pixels. You can set CompatConfig.pixelThickness = **true** in order to revert to using pixel values.

New in version 3.5

Control nodes

ControlNode objects display custom HTML content, specified via their Template property. A node's DOM structure is created inside the Content div element, which is then rendered on top of the diagram canvas. Note that this prevents **ControlNode**s drawing in diagram' usual Z order, and they will always appear on top of other diagram items.

By default the diagram intercepts mouse events of all hosted html elements to enable moving nodes or drawing links. You can specify that interactive elements, such as buttons or text boxes, should receive input

instead by setting their **data-interactive** attribute to **true**.

You can register event handlers for elements in the template by assigning function names to **data-event** attributes in the form data-event-'eventName'='handlerName'. Alternatively, you can attach event listeners using DOM API once nodeDomCreated is raised to signify that **content** is available. E.g. access child elements by calling args.getNode().getContent().querySelector(...) and attach handlers via elements' **addEventListener** method.

ControlNodes create DOM elements from their template only in the main diagram, and render images instead when shown in Overview or NodeListView. Note that in this case image elements in the template will render only if specified in base64 format. The Utils.toDataUrl method helps you convert an image URL to base64 data URL.

New ControlNodes.html example included in distribution demonstrates the ControlNode API.

Button components

Composite nodes can now contain buttons. Button components respond to user clicks by calling the JavaScript function whose name is assigned to their **clickHandler** attribute. A button can show a text label assigned to the **text** attribute, or an image whose URL is assigned to **imageLocation**. The following example adds a button to CompositeNode template:

JavaScript

```
{
    component: "Button",
    brush: "#ccc",
    text: "Delete",
    width: 30,
    cornerRadius: 3.5,
    clickHandler: "onDeleteClick"
}
```

Miscellaneous

- Content of CompositeNodes is now clipped by default if Shape in the template is marked with **isOutline** attribute. To disable that, set ClipToOutline to **false**.
- Set ExpandButtonAction to RaiseEvents and +/- buttons of Expandable nodes will only raise an expandButtonClicked event instead of expand or collapse the tree under that node. You can use this to implement custom logic for the +/- button.

New in version 3.4

React support

The DiagramView React component allows integrating the MindFusion.Diagramming API into React applications. It renders a Diagram instance assigned to "diagram" prop as its model. Most diagram properties can be set from JSX, and all diagram events can be handled through JSX syntax as well. For more information, see the Integration with React topic.

Vue.js support

The diagram-view Vue.js component allows integrating the MindFusion.Diagramming API into Vue.js applications. It renders a Diagram instance assigned to "diagram" prop as its model. Most diagram properties can be set from the vue template, and all diagram events can be handled through the v-on directive as well. For more information, see the Integration with Vue.js topic.

Angular support

The diagram-view Angular component allows integrating the MindFusion.Diagramming API into Angular applications. It renders a Diagram instance assigned to "diagram" property as its model. Most diagram properties can be set from the html template, and all diagram events can be handled through event binding syntax as well. For more information, see the Integration with Angular topic.

Video nodes

VideoNode objects display video streams, along with UI elements to play, pause, seek and change volume. Call node's setVideoLocation method to specify its video URL. Auto-playing is not supported at this time, and the video can be started only by users clicking the play button. **VideoNode** is built around Video component, which can also be used in CompositeNode templates.

Miscellaneous

- Improved automatic routing of links.
- More precise scrolling with TableNode scrollbar thumb.

New in version 3.3.4

Shape components

The Shape class used to specify ShapeNode geometry can also be used as a component. When its **isOutline** attribute is set, the shape will control CompositeNode's geometry too, defining hit test and clip area, and link alignment points along node's border. If **isOutline** is disabled, the shape will serve mostly as a decorative element. The code below shows a sample fragment specifying shape component in template:

JavaScript

```
{
    component: "Shape",
    id: "Cloud",
    autoProperty: true,
    name: "OutlineShape",
    pen: "gray",
    brush: "Transparent",
    isOutline: true
},
```

Miscellaneous

- containerChildAdding event handlers can stop propagation of the event up in the container hierarchy by calling setHandled.
- TableNode.resizeToFitText improvements and fixes.
- LayeredLayout now uses stable sorting and should always create same results when applied to same graph.

New in version 3.3.3

- Fixed handling of transparent BackBrush in new drawing system.
- Fix for initializeNode and initializeLink events not firing.
- Fixed several drawing glitches.

New in version 3.3.2

Improved performance

Diagram rendering and user interactions are now faster. The control redraws the smallest possible area after diagram changes, and compresses refresh operations into a single batch when possible. Custom node classes that need to draw outside of their Bounds should override the getRepaintBounds method to specify the repaint area.

Miscellaneous

- The Border component draws frame lines around and background behind other components in CompositeNode templates.
- NodeListView now displays nodes' Tooltip texts when the mouse hovers over an item in the list view.

- Fix for FlowchartLayout creating node overlaps in some situations.

New in version 3.3.1

Default StandAlone mode

The component now loads in stand-alone mode by default and does not depend on any third party libraries. With this change, it won't work out of the box in Internet Explorer versions older than 11. You can still explicitly enable JQuery or MsAjax modes for older browsers, or alternatively use polyfill libraries for missing APIs (e.g. JSON serialization).

Miscellaneous

- FlowchartLayout fixes and improvements.
- The linkTextEdited event is now raised for labels too. Access to the edited LinkLabel is done via the getLabel method of LinkEventArgs.
- **LinkLabel** property setters now redraw the label automatically
- Fixed focus border appearing at click position in Firefox.

New in version 3.3

Flowchart graph layout

FlowchartLayout recognizes program code-like patterns in the graph, such as loops, sequences and if/switch branchings, and arranges them recursively. **FlowchartLayout** could be used to arrange other types of graphs as well, though with some restrictions. For example it treats all back links as loops in the code, and expects that they are nested - loop links starting closer to the stop node should end closer to the start node. Another similar restriction is that there shouldn't be any cross-links that connect different branches of a decision sub-graph.

Container improvements

- resizeToFitText method and EnableStyledText property added to the ContainerNode class.
- containerChildRemoved event now also raised when dragging multiple selected child nodes out of a container.
- Improved handling of folded containers by automatic layout classes and link routing methods.

Miscellaneous

- Typescript definitions now available for the Diagramming.Lanes namespace.
- Auto-arranged link labels now can optionally be placed over link segments and containers by setting diagram's AutoArrangeAvoidSegments and AutoArrangeAvoidContainers properties to **false**.
- The dispose method of Diagram class now removes a focus-proxy DIV element from the page.
- Improved text alignment for DiagramLink texts.

New in version 3.2.1

Path finding

The PathFinder class provides methods that help you find paths and cycles in a graph:

- findShortestPath finds the shortest path between two DiagramNode objects.
- findLongestPath finds the longest path between the specified **DiagramNode** objects.
- findCycle detects whether the specified **DiagramNode** participates in a cycle.
- findAllPaths finds all paths that exist between two **DiagramNode** objects.
- findAllCycles finds all cycles in the underlying diagram.

Path objects returned by these methods contain nodes, links and items arrays containing sequences of elements in the path. The new PathFinder sample page included in distribution demonstrates path finding and animations over the found paths.

Embedded hyperlinks

Nodes and Text components with style text enabled can now contain <a> tags to create hyperlinks. When a link is clicked, the control raises hyperlinkClicked event to let you implement navigation:

JavaScript

```
// node is a ShapeNode instance
node.setText("test <a='https://mindfusion.dev'>link</a> test");
node.setEnableStyledText(true);

// attach an event listener to the hyperlinkClicked event
diagram.addEventListener(Events.hyperlinkClicked, onHyperlinkClicked);

function onHyperlinkClicked(sender, args)
{
    window.open(args.getHyperlink());
}
```

Miscellaneous

- The serializeTag event lets you save complex Tag and Id objects in XML format.
- Various arrowhead rendering fixes.

New in version 3.2

Export to SVG

The SvgExporter class creates a Scalable Vector Graphics (SVG) drawing from the content of a Diagram. The exportElement method returns a DOM <svg> element that can be directly added to the page. The exportString method returns an SVG string that can be saved in local storage or submitted to server.

C#

```
function onExportSvg()
{
    var diagram = $find("diagram");
    var exporter = new MindFusion.Diagramming.SvgExporter();
    var svgElement = exporter.exportElement(diagram);
    document.body.appendChild(svgElement);
    svgElement.style.position = 'fixed';
    svgElement.style.zIndex = 100;
}
```

AMD module

The diagramming.js script can be loaded as an AMD module now. If it detects an AMD loader during startup, it will list common.js as its dependency, so it is enough to require only diagramming.js as application's direct dependency:

C#

```
MindFusionImpl = "StandAlone";

requirejs(["MindFusion.Diagramming"],
    function(MindFusion)
    {
        // create a Diagram component that wraps the "diagram" canvas
        var Diagram = MindFusion.Diagramming.Diagram;
        var element = document.getElementById("diagram");
        var diagram = Diagram.create(element);
        var node = diagram.factory.createShapeNode(10, 10, 40, 30);
        node.setText("Hello, world!");
    }
);
```

Miscellaneous

- Arc definitions in custom shapes now follow correct SVG syntax (A rx ry x-axis-rotation large-arc-flag sweep-flag x y) instead of specifying bounding box and angles.
- ShapeNode's Stroke attribute is now also applied to shape decoration elements.
- Underlined fonts can be applied to the text of links.
- RowCount and ColumnCount properties now exposed by TableNode.

New in version 3.1

Composite nodes

The CompositeNode class implements nodes whose appearance can be defined via composition of components and layout containers. The content of a composite node can be created by building a tree of components programmatically, or by loading a JSON template. This initial release includes layout containers such as StackPanel and GridPanel. Objects from the MindFusion.Drawing namespace now double as **CompositeNode** components (Image, Text, Path, Rect). Future versions will add interactive components like buttons and text editors. Extended overview is available here.

Animated layout

arrangeAnimated methods added to Diagram and ContainerNode classes display movement of items from their original locations to new locations assigned by the layout object. This can be used to create an explode effect by starting new layout from overlapping nodes on same position, or an insertion effect by adding a new node to an existing layout.

Overview improvements

- ScaleMode enumeration and properties added to Overview control implement several scaling modes. The **FitAll** element corresponds to the original behavior where the **Overview** always fits diagram's contents. **FixedScale** mode applies the scale specified via overview's ScaleFactor property. **CombinedScales** mode multiples overview's ScaleFactor by diagram's current ZoomFactor.
- If AllowZoom is enabled, users can zoom the diagram by resizing the overview's viewport tracking rectangle.
- MinVisibleFontSize specifies a threshold value that hides text if scaled font sizes become smaller.

Miscellaneous

- ImagePadding property of ShapeNode and Cell lets you set padding space between elements' borders and their Image.
- resizetoFitText method of TableNode resizes the columns and rows of tables to fit cells' texts.
- The new GridRouter class implements link routing using A* algorithm with customizable criteria costs exposed by its LengthCost, NodeVicinityCost and TurnCost properties.
- All sample projects now available in TypeScript.

New in version 3

User-interaction controllers

All kinds of user interaction are now carried out by controller objects. The **startDraw** method of BehaviorBase-derived classes that returned an opaque **InteractionState** instance has been replaced by createController method returning a controller of class specific to the modified item and interaction types. Built-in controllers include CreateNodeController, CreateLinkController, ModifyNodeController, ModifyLinkController, PanController. You can also create custom controllers by deriving and overriding the methods of SinglePointerController:

JavaScript

```
SinglePointerController.prototype =
{
    start: function (position) {},
```

```
    move: function (position) {},
    validate: function (position) {},
    commit: function (position) {},
    cancel: function (position) {},
    drawInteraction: function (context) {}
}
```

Record and replay

The Diagram class provides record, stopRecording and replay methods that can be used to save and replay all user actions with keyboard and mouse. The replay method takes *recording*, *timer* and *diagramJson* parameters. The *recording* argument is an array of records returned by stopRecording. If the *timer* argument is set to **true**, events are replayed from a timer preserving their original timing, otherwise they are replayed immediately. Timer mode can be used to show tutorials played directly in the diagram. Immediate mode could be used to automate user interaction testing where an assertion is checked after replaying series of events. Recorded events could also be replayed as macros if ran on a diagram with items placed on same initial coordinates.

Improved testability

User interaction can now be tested automatically in several ways outlined below.

- Create a MouseInputDispatcher instance and call its methods

  JavaScript

  ```
  onMouseDown: function (position, button)
  onMouseMove: function (position)
  onMouseUp: function (position, button)
  ```

  where position is a Point instance specifying logical coordinates in the diagram (in current measure unit, not scaled by zoom factor). The results of called mouse methods will depend on diagram's current Behavior, e.g. you could assert that a sequence of down, move and up events creates a ShapeNode when Behavior is set to DrawShapes.
- Create an instance of controller class and call its start, move and commit methods. This would not depend on the value of diagram's Behavior, but could be still affected by other behavioral properties such as AllowSelfLoops and AllowUnconnectedLinks.
- Check assertions after replaying a recording on a saved diagram.

Ruler control

The Ruler control provides horizontal and vertical scales that help users measure and align diagram items. In order to set up a **Ruler** instance, create a DIV element parenting the diagram's scroller div -

HTML

```
<div id="ruler">
  <div id="view">
     <canvas id="diagram" width="2100" height="2100">
     ...
```

JavaScript

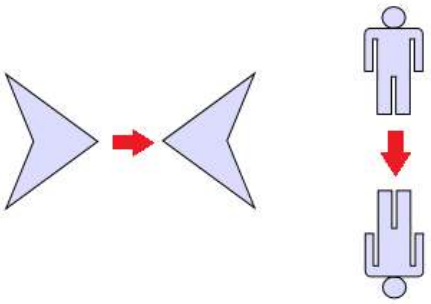
```
var ruler = MindFusion.Diagramming.Ruler.create(document.getElementById("ruler"));
ruler.setDiagram(diagram);
```

3rd party dependencies now optional

The diagram can now be used without JQuery and Microsoft Ajax libraries. To enable that mode, set MindFusionImpl = "StandAloneImpl" before loading diagramming.js. Stand-alone mode requires relatively modern browsers, the base feature-set against we test it is the one of IE11.

Flip shapes

The shape of a ShapeNode can be flipped horizontally or vertically. Enable the AllowFlip property and the shape flips when an edge is dragged over its opposite edge. You can get or set current flip state using FlipX and FlipY properties.



Revamped item dependency system

Dependencies between diagram items are now expressed by rule objects and applied in sequence by the DiagramMediator class. Built-in dependencies are implemented by ItemFollowsContainerRule, AttachedFollowsMasterRule, LinkFollowsNodesRule, NodeFollowsSelectionRule rules. You can create custom rule objects with following API and add them to DiagramMediator.AllRules array to make items follow other items during user interaction -

JavaScript

```
var MyRule =
{
    dependencies: function (master) {}, // return array of items that should follow master
    dependentUpon: function (dependency) {},  // return array of items that should be followed
by dependency
    recognize: function (dependency) {},  // return true or false if this rule should be applied
on an item
    updateFromMasters: function (dependency, originalStates) // set new position or size of
dependency
}
```

License keys

There are no separate trial / licensed builds of MindFusion scripts provided anymore, instead you disable trial mode by specifying a license key string. License keys are now listed on the Keys & Downloads page at MindFusion customer portal. Copy the key to a diagram_lic.txt file in same folder where the diagram html page is located. The file name and location can be changed by calling the diagram's setLicenseLocation method. Alternatively, the key string can be set directly by calling setLicenseKey method.

Miscellaneous

- CaptionBackBrush and FoldIconSize properties added to ContainerNode.
- Multiple rotation of nodes improved, now node angles are modified by same rotation delta calculated from mouse position instead of pointing all rotation handles in mouse direction.
- Xml serialization improved.
- LinkSegments property of Diagram specifies initial number of link segments.
- JQuery mode now supports version 3 of JQuery.

New in version 2.8

Fluent API

Builder objects with property setters and shortcut methods for font and brush creation add support for fluent programming style. Static With and instance init methods in DiagramItem, Style and Layout -derived classes return a builder instance that can be used to set up respective new or existing objects:

JavaScript

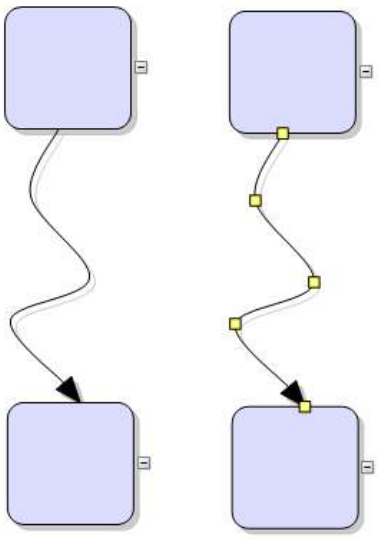
```
diagram.addItem(
    ShapeNode.With()
        .brush("lightGray")
        .font("Arial", 4)
        .enableStyledText(true)
        .text("Task <i>1</i>")
        .tooltip("This is the task")
        .create());

diagram.getFactory()
    .createShapeNode(20, 20, 20, 20).init()
        .brush("lightGray", "white", 20)
        .font("Arial", 4)
        .enableStyledText(true)
        .text("Task <i>2</i>")
        .tooltip("This is the second task");

diagram.arrange(
    TreeLayout.With()
        .levelDistance(20)
        .nodeDistance(20)
        .linkType(TreeLayoutLinkType.Cascading3)
        .create());
```

DiagramLink improvements

- HeadStroke, HeadStrokeThickness and HeadStrokeDashStyle properties let you customize arrowhead strokes independently of line segments strokes.
- The AllowSelfLoops property of Diagram class controls whether users are allowed to draw self-loop links.
- The new Spline element of **LinkShape** enumeration draws links as interpolating splines that pass through all of their control points:



Miscellaneous

- TextAlignment and LineAlignment properties moved from ShapeNode to DiagramItem; you can now use them to set alignment for table and container captions as well.
- Enable the AllowResizeHeaders property of swimlanes Grid to let users resize grid rows and columns interactively. The control raises headerStartResizing validation event when the user tries to resize, and headerResized event when the operation completes.

- IconSize and DefaultNodeSize properties of NodeListView can be set to **null**, in which case the control will use the sizes stored in nodes' Bounds.
- Implemented backward compatibility for XML files as old as v13 format (created by MindFusion diagram versions released in 2008).
- The control now supports cyclic attachment created using DiagramNode.attachTo method, where moving any of the attached nodes will move all other nodes in the cycle.
- RoundRect shape's corner arcs now keep constant radius and no longer deform when the node size changes to non-rectangular.
- itemAdded and itemRemoved events raised when an item is added or removed, either from code or interactively.

New in version 2.7

Treemap layout

Tree-maps represent hierarchies by nesting child nodes within their parents, where the areas of leaf nodes are proportional to their Weight values. Unlike other layout algorithms, TreeMapLayout expects hierarchies to be defined via grouping or containment (attachTo method and ContainerNode class), and will ignore any links in the diagram. The diagram area covered by the topmost nodes in a hierarchy is specified via the layoutArea property. By default, the layout tries to keep the ratio of node sides as close as possible to one. However this could make it hard to distinguish separate levels of the hierarchy. To alleviate that, set squarify to **false**, and child nodes will be arranged either as a row or a column inside their parent node, alternating directions for each level. The drawback is that when Weight ratios differ greatly or nodes contain many children, some nodes could end up with very narrow rectangles.

XML serialization

In addition to the loadFromXml method added in a previous version, the Diagram class now provides saveToXml method that serializes its contents in XML format supported by all MindFusion diagram libraries for mobile, desktop and web platforms. **saveToXml** posts the XML as text/xml to specified URL. The new saveToString and loadFromString methods generate or load XML strings instead, when their argument is set to SaveToStringFormat.Xml (a Json enumeration member creates JSON strings by calling older toJson method).

Miscellaneous

- Set TableNode.prototype.useScrollBars to show full scrollbars for scrollable tables instead of up/down buttons in captions.
- The ShowDeleteButton property of DiagramNode displays a close/delete X button in upper-right that lets users delete the node by clicking.
- Styled text mode (EnableStyledText) now supports <sub> and <sup> tags for respectively subscript and superscript.
- DiagramLink now calls **onUpdateVisuals** method to let you implement custom-drawing, working in same manner as its counterpart in node classes.
- The new *Flowcharter* sample project shows how to create an online flowchart editor, providing UI for zooming, scrolling, creating new nodes and formatting nodes' contents.
- The new *TreeMap* sample project demonstrates the TreeMapLayout tree mapping algorithm, where node weights are set to either population size or area of world countries.
- The new *Inheritance Diagram* sample project demonstrates how to visualize classes using TableNode objects.

New in version 2.6

Free-form nodes

A FreeFormNode collects all points from users' mouse or touch input and displays them as node's outline. To let users draw free-form nodes interactively, set Behavior to DrawFreeForms or **LinkFreeForms**. Use the Points property of **FreeFormNode** to get or set outline points programmatically. If the Closed property is set, the node is drawn as a closed shape and its interior filled, or otherwise the node is drawn as a poly-line. If the distance between first and last points drawn by user is shorter than AutoCloseDistance, the node's

Closed property is automatically set to **true**. AutoCloseDistance default value is Number.MAX_VALUE, so free-form nodes are always closed.

Convert free-form drawings to ShapeNodes

Additional drawing modes, convenient for touch input, convert FreeFormNode objects drawn by user to ShapeNode objects with matching shapes. To enable them, set Behavior to DrawFreeShapes or **LinkFreeShapes**. The shapes against which the user's input is matched are set via diagram's FreeFormTargets property. By default it contains Rectangle, Decision and Ellipse shapes.

Miscellaneous

- TypeScript definitions for the Diagramming API are now provided in *jsdiag.d.ts* file.
- nodePasted and linkPasted events are now raised when new DiagramNode and DiagramLink instances are created from clipboard contents.
- VirtualScroll mode fixed for parent div size specified in percent units.
- Fixed bug where the ExpandOnIncoming property specifying link direction for collapse/expand actions did not work correctly.
- Removed several global variables created inadvertently by the control.

New in version 2.5.1

- **System3D** member added to CellFrameStyle enumeration. If it's assigned to the CellFrameStyle property of a TableNode, cell frames are rendered using 3D effect.
- Resize and move interaction cursors now remain visible until the user releases mouse button.
- Fixed bug where auto-handles mouse hint cursor would not match the item type.
- Fixed bug where multiple lines in link's Text were drawn overlapped.
- Fixed bug where BaseShape arrowhead would point in wrong direction for strictly vertical links.
- Fixed fromJson bug where ShapeNodes were not loading correct shape.

New in version 2.5

Resize table columns and rows

Columns and rows of a TableNode can now be resized interactively if its AllowResizeColumns or AllowResizeRows properties are enabled. In order to resize, move the mouse pointer to the border line on column's right side or row's bottom side until it shows resize cursor and start dragging. The control raises tableColumnResizing and tableRowResizing events to let you validate new size or prevent resizing some elements. The tableColumnResized and tableRowResized events are raised after the operation completes.

Shape libraries

The ShapeLibrary class lets you use custom geometric shapes created using MindFusion ShapeDesigner tool. Call its loadFromXml method to load a library XML file. getShapes returns an array of the loaded Shape objects. The ShapeLibraryLocation property of NodeListView creates a prototype ShapeNode object for each shape from the specified library.

Miscellaneous

- TextStroke and TextStrokeThickness properties of DiagramItem let you set color and thickness of text outlines.
- Items can now be deleted using Backspace key when running on Mac.
- Caption divider line in TableNode and ContainerNode is now rendered clipped when caption height is smaller than corner radius.
- The TooltipDelay property specifies the delay in milliseconds before showing tooltips.
- The Orientation property of NodeListView lets you set the view's orientation to **Horizontal** or **Vertical**.
- MindFusion.Common.js contains code shared with other JavaScript libraries by MindFusion. It must be loaded before the MindFusion.Diagramming.js script.

Fixed bugs

- Fixed overlaps in TreeLayout when arranging nodes of different sizes.
- Anchor points indices were not serialized and could be reset to different values when loading from JSON.
- Deserialization of custom item classes worked correctly only if their names contained Node or Link suffix.

New in version 2.4

Appearance improvements

- Set the Shape property of tables and containers to RoundedRectangle to render them with rounded corners.
- Hide the frames of table cells by setting the CellFrameStyle property to None.
- Set the EnableStyledText property of TableNode class to render styled text in tables.

New events

- The control raises cellTextEdited event when users edit the text of table cells.
- The createEditControl event lets you create custom DOM element or fragment to use as in-place text editor.
- NodeListView raises nodeSelected event when the user selects a node.

Miscellaneous

- jQuery mode is now the default and all sample projects have been updated to use jQuery. To continue using the Microsoft Ajax library, set MindFusionImpl = "MsAjax" before loading Diagramming.js
- the loadFromXml method of Diagram class lets you load XML files created by MindFusion diagram controls for other platforms.
- fixed setZoomFactorPivot bug in virtual scroll mode.

New in version 2.3

Styled text in nodes

The EnableStyledText property of ShapeNode allows using HTML-like formatting tags to apply various attributes to the node's text. At this time the component supports the following formatting tags:

- <b> specifies bold text
- <i> specifies italic text
- <u> specifies underlined text
- <color=value> specifies text color
-
 specifies line break

Text rendering improvements

- General text rendering quality improved. The component removes scale transformations applied for MeasureUnit and ZoomFactor before drawing text on the canvas, and instead specifies a scaled font size, which helps improve text appearance in Firefox and older versions of Chrome.
- Now items' text can be drawn underlined. To enable this, set the underline attribute of the Font class.
- Font styles can be specified via Style instance by calling its setFontStyle method with a combination of FontStyle enumeration members.

Miscellaneous

- Items can be drawn using dashed lines in browsers that support the **setLineDash** function. To enable this, set the StrokeDashStyle property of DiagramItem or Style to a member of the DashStyle enumeration.
- TreeLayout supports organizational charts with assistant nodes as in Microsoft Office diagrams. To mark nodes as assistants, set the **treeLayoutAssistant** field of node's LayoutTraits to a member of the

AssistantType enumeration. Set the TreeLayout.enableAssistants flag to arrange assistant nodes in a separate branch between the main node levels.
- Specify if separate connected components of a graph should be arranged horizontally or vertically relatively to each other by setting the multipleGraphsPlacement attribute of layout classes.
- The type of LinkLabel.Margin property has been changed from number to Thickness, letting you specify different margin sizes at each side of the label.

New in version 2.2

Zoom control

The ZoomControl class lets users change interactively the current zoom level and scroll position of a Diagram. To set it up, add a <canvas> to the page, place it anywhere over the target diagram canvas, call the create method, and set the control's Target property to that diagram. Set the ZoomStep and ScrollStep properties to specify the amount added to diagram's zoom level or scroll position by **ZoomControl**'s buttons. You can customize the control's appearance by setting properties such as Fill, BorderColor, CornerRadius and TickPosition.

Shadow styles

The Diagram.ShadowsStyle property and ShadowsStyle enumeration let you switch shadow rendering order between rendering all shadows in a single background layer (**OneLevel**) and drawing each shadow close to its node, possibly overlapping other nodes (**ZOrder**). **ShadowsStyle.None** lets you disable shadows altogether.

Miscellaneous

- Set the LinkLabel.Brush property to fill the background of link labels.
- Number of link segments can now be set by calling the DiagramLink.setSegmentCount() method.
- The BackgroundImageAlign property specifies the alignment of diagram's BackgroundImage.
- The TextPadding property specifies the padding distance between the borders of a node and its text.
- Nodes of all classes can be rotated.

New in version 2.1

SVG Nodes

The SvgNode class represents nodes that can display SVG drawings. SVG graphics are rendered instead of Image, and on top of the geometry rendered by the base ShapeNode class. Enable the Transparent property to hide the node's geometry and leave only the drawing visible. The SVG file is loaded and associated with the node through an SvgContent object, which should be initialized and then assigned to the Content property.

jQuery support

Now the control can use jQuery for browser abstraction instead of Microsoft Ajax. To enable jQuery support, set MindFusionImpl="JQuery" before loading the diagram script. Instead of Microsoft Ajax $create function, use the static Diagram.create(element) method to create a Diagram instance for a given Canvas element.

Miscellaneous

- Set the BackgroundImageUrl property to draw an image as Diagram's background.
- The TextStyle property of links provides support for several different text position and orientation styles, defined by the LinkTextStyle enumeration.
- Set the AllowUnconnectedLinks property to true to let users draw links without connecting them to nodes.

New in version 2.0

Undo / redo support

If the UndoEnabled property is set to **true**, the Diagram control tracks changes done to its items and allows undoing and redoing them later by calling the undo and redo methods respectively. Multiple changes could be recorded as a single undoable operation by enclosing them between startCompositeOperation and commitCompositeOperation calls. It is also possible to create custom undoable operations by deriving from the Command class and calling executeCommand with the custom command as argument.

Animations

Diagram items can now be animated through the Animation class in the MindFusion.Animations namespace. The class implements several built-in animation and easing types, and allows definition of custom animation functions, to create fluid and pleasing animation effects. The new Animations example demonstrates how to apply several animation types to both nodes and links.

Clipboard support

A single item or a selection of items can be copied or cut to a clipboard maintained by the diagram control. This is done programmatically using the copyToClipboard and cutToClipboard methods. To paste the clipboard contents into the current diagram, call pasteFromClipboard. If running under Internet Explorer, it is also possible to copy to the system clipboard by passing a **true** argument to clipboard methods.

Swimlanes

The lane grid lets you emphasize the relationship between a group of diagram items by displaying them in a distinct lane or cell within the grid. To display the lane grid, set the ShowLaneGrid property of the Diagram class to **true**. In order to customize the grid, set the various attributes exposed by the LaneGrid property, which lets you specify the number of rows and columns, add headers, customize the cell appearance, etc. The new Lanes sample page uses the lane grid to let users draw Gantt charts.

Resize multiple nodes

Now it is possible to resize multiple selected nodes simultaneously. To enable that, set the AllowMultipleResize property to **true**. When enabled, dragging a corner or side adjustment handle of any node resizes all nodes in the selection. The operation can be cancelled for all nodes by handling the selectionModifying validation event.

Magnifier

The new magnifier tool allows users to interactively zoom in (or out) portions of the diagram by holding down a modifier key or pressing a mouse button. The magnifier's appearance can be customized via properties such as MagnifierShape and MagnifierFrameColor. Its size and magnification can be set via MagnifierWidth, MagnifierHeight and MagnifierFactor properties. The magnifier appears when users press the mouse button mapped to MouseButtonActions.**Magnify**. It can also be shown programmatically by setting the MagnifierEnabled property.

Intellisense support

The package now includes a MindFusion.Diagramming-vsdoc.js file providing code completion information. To load it in Visual Studio, add e /// <reference path="MindFusion.Diagramming-vsdoc.js" /> tag to the top of your script files. Use the static Diagram.create and Diagram.find methods instead of $create and $find to let Visual Studio infer the type of returned Diagram objects correctly.

Miscellaneous

- Set the ModificationStart property to AutoHandles to let users start moving or resizing an item without selecting it first.
- Customize the function of keyboard modifier keys via the ModifierKeyActions property.
- The diagram area can be resized automatically to fit the current diagram items as set through the AutoResize property.

- The AutoScroll property enables automatic scrolling when the mouse is dragged near the diagram edges.
- enterInplaceEditMode and leaveInplaceEditMode events raised when the control shows or hides the in-place edit box.
- **linkPointed** and **nodePointed** events raised when the mouse hovers over an item.
- The control displays a scaled-down copy of the node being dragged from the NodeListView control.
- Set Behavior to SelectOnly to let users select existing items but not to modify them or draw new ones.

New in version 1.7.1

Rounded links

Links whose Shape is set to Polyline or **Cascading** can be rendered with rounded joints between their segments. To enable that, set the diagram's RoundedLinks property to true. The RoundedLinksRadius property lets you specify the radius of joint arcs.

Link crossings

The component can represent the crossing point of two links either as an arc (bridge) drawn for the link with higher Z index, or as a cut in the link with lower Z index. To enable that, set the diagram's LinkCrossings property to either Arcs or **Cut** member of **LinkCrossings** enumeration. Set CrossingRadius to specify the radius of crossing arcs.

Miscellaneous

- Node rotation was ignored by link routing algorithm; now links are routed around the rotated node's boundaries.
- The routeAllLinks function pulls apart link segments when they would overlap.
- Several new predefined shapes available in the Shapes class - RightTriangle, Decagon, Trapezoid, Star4Pointed, Star5Pointed, Star6Pointed, Star7Pointed, Star16Pointed, Star24Pointed, Star32Pointed, Donut and Plaque.
- Set Behavior to Pan to only let users pan the view when dragging the mouse.
- The component now raises clicked event for the diagram when there isn't any item at the click location (which would cause nodeClicked or linkClicked event).

New in version 1.7

Adjustment handle styles

Appearance of nodes' adjustment handles can be customized by calling the setHandlesStyle method. Supported styles include round and square handles, dashed and hatched frames, and some combinations. Apart from appearance, the handles style also controls what part of a node can be dragged to move or resize it. It is also possible to render custom handles by handling the drawAdjustmentHandles event, and implement custom hit-testing by handling the hitTestAdjustmentHandles event.

Dynamic links

A dynamic link automatically orients its end segments so that they point to the target nodes' centers while the link's nodes are dragged around. If a node's AnchorPattern is set, link ends will align to the closest anchor point instead. Positions of the link's end points are also updated when the next-to-last control points are being modified. To enable this, set the Dynamic property to **true**.

ContainerNode enhancements

- Apply layout algorithms to content of a ContainerNode by calling its arrange method.
- Selectively prevent adding or removing child nodes to/from a container by handling the containerChildAdding and containerChildRemoving events.

Miscellaneous

- The Dirty flag is automatically set when a serializable property of an item or the diagram has been modified. It lets you determine where there are any changes that need to be saved.
- The AdjustmentHandlesSize property specifies the size of adjustment handles.
- The ShowDisabledHandles property controls whether disabled adjustment handles are draw on screen.

New in version 1.6.1

ContainerNode improvements

- Child nodes are now drawn by their containers; when containers overlap, the children of a lower container cannot appear in front of the upper container anymore.
- The ClipChildren property specifies whether to clip child items to container's boundaries.
- The ZIndex property no longer changes automatically when dropping nodes into a container.
- The Visible property of child items no longer changes when folding or unfolding containers; this lets you keep invisible items inside containers.

Miscellaneous

- The LinkHitDistance property lets you specify the furthest distance from links that a click would be considered a hit.
- The HyperLink property lets you associate an URL with diagram items.
- EnabledHandles lets you enable or disable individual adjustment handles of nodes.
- The arrangeLinkLabels method rearranges link labels whose AutoArrange flag is set.

New in version 1.6

Node effects

Two visual effects, represented by GlassEffect and AeroEffect classes, can be applied to nodes. To apply an effect, create an instance of the respective class, set up its properties, then add the instance to the Effects collection of a node (or to Diagram.NodeEffects collection to apply it to all nodes). Effects can be added, removed or modified at any time and this will immediately reflect on the diagram. Effects of different types can be applied simultaneously. It is also possible to apply more than one effect of the same type.

Multiple labels per link

The LinkLabel class allows multiple captions to be displayed for a single DiagramLink object. Link labels provide a set of properties allowing full customization of their display and positioning. Labels can also be arranged automatically to avoid overlapping nodes and other labels by setting LinkLabel.AutoArrange to true.

New events

- nodeSelected and linkSelected events are now raised when an item is selected, either programmatically or by the user.
- nodeDeselected and linkDeselected events are now raised when an item is deselected, either programmatically or by the user.
- nodeDoubleClicked and linkDoubleClicked events raised when the user double clicks an item.
- nodeCreating is now also raised while the user drags a node from the NodeListView control.

Miscellaneous

- Call setVirtualScroll to enable virtual scrolling mode. In this mode, the canvas stays as big as its containing div, and its content is drawn with a translation bound to the div's scrollbar positions. This allow displaying large diagrams in browsers and on mobile devices where the size of HTML canvas elements is limited.
- resizeToFitText method added to ShapeNode.
- Improved support for nested containers.

New in version 1.5

Styles and Themes

An instance of the Style class can be used to set items' appearance properties as a single unit via the setStyle method. In order for style attributes to be used, the respective local appearance properties of DiagramItem objects must have undefined or null values. If an item's Style does not define a value for some attribute, it is looked up in the styling hierarchy, first in the diagram's Style, next in the current Theme, and finally in the set of default styles defined for each item class.

Shadows

Diagram items can now cast shadows. The shadow color is specified via the setShadowColor method and the offset of a shadow from its item is set via setShadowOffsetX and setShadowOffsetY methods. Shadows aren't rendered if getEffectiveShadowColor returns null or undefined value.

Radial gradients

Nodes can be filled using radial gradients by passing objects in following format to setBrush. If the gradient should blend only two colors, they can be specified using **color1** and **color2** fields instead of **colorStops**.

JavaScript

```
var brush =
{
 type: "RadialGradientBrush",
 colorStops:
 [
  { position: 0, color: "white" },
  { position: 0.5, color: "lightGray" },
  { position: 1, color: "blue" }
 ],
 x1: 20, y1: 20, radius1: 30,
 x2: 40, y2: 20, radius2: 80
};
node.setBrush(brush);
```

Miscellaneous

- Call the setTooltip method to associate tooltip text with an item. The tooltip appears when the mouse pointer hovers over an item for a second.
- Specify the line width of item frames via the setStrokeThickness method.
- initializeNode and initializeLink events are raised when users start drawing a node or a link interactively.
- The NodeListView control can now host tables and containers. The **addBox** and **selectBox** methods have been replaced by addNode and selectItem methods.
- The setStartPoint and setEndPoint methods of links provide shortcuts for setting the first and last points in ControlPoints.

New in version 1.4

Table Nodes

A TableNode displays tabular data, and can be used to represent table schemas in database diagrams or classes in class-hierarchies. A table contains cells arranged in a grid, and every cell can display text and/or image. Cells, columns and rows can be customized in various ways - there are properties available for setting their extents, style, image and text alignment. The number of cells in a table's grid can be programmatically changed by calling methods such as redimTable, insertRow and insertColumn. If a table contains many rows, you can let users scroll them up or down by calling the setScrollable method.

Container Nodes

A ContainerNode can contain other diagram nodes. When a container is moved, its child nodes move too. Containers can be nested one within another to unlimited depth. They can be folded to a smaller size, hiding their child items, and unfolded to show content again. Nodes inside a container can be linked to

nodes both inside and outside of it. If a link connects items in the same container, the container is not considered an obstacle by the auto-routing algorithm.

Miscellaneous

- The Locked property can be used to disable interaction with a diagram item.
- Improved rendering speed when changing items' appearance properties.

New in version 1.3

Attaching and grouping items

JsDiagram allows attaching a node to another node, establishing a subordinate - master relationship between them. The easiest way to attach nodes is to call the attachTo method. When a master node is moved, all of its subordinates follow it, so that the initial distance between them stays constant. This happens both when a user moves the master node around, and when the node's position is changed using a method or a property.

Collapse and expand tree branches

If a diagram represents a tree structure, the tree branch starting from a node can be collapsed or expanded again in order to hide or display hierarchy details. To enable this, call the setExpandable method. There is a [±] button displayed beside an expandable node to allow users to collapse or expand the tree branch starting there. By default, pressing that button automatically toggles the Expanded property of the node and raises the TreeExpanded or TreeCollapsed events.

User interaction modes

Call setBehavior with one of the Behavior members as argument to specify how the diagram should handle mouse input.

- DoNothing: the diagram is read-only, but nodeClicked and linkClicked events are still raised.
- Modify: existing diagram items can be selected and modified, but no new items can be created interactively.
- DrawLinks: existing nodes can be linked interactively, but no new nodes can be drawn.
- DrawShapes: drawing with the mouse creates new ShapeNode objects; links cannot be created interactively.
- LinkShapes: this is the default behavior, where drawing over an empty spot of the diagram creates a **ShapeNode**, and drawing over a node creates a link.

Touch input

When running under the iOS Safari web browser, the control handles one-finger touch events to allow drawing and selecting items interactively. In addition, the hit-test area around adjustment handles is larger to make touch modifications easier. Two-finger gestures are delegated to the browser - they either scroll the canvas when dragging in the same direction or zoom the page when dragging in opposite directions. The preventDefaultTouch flag can be set to **false** to revert to the default one-finger touch behavior (scroll the whole page).

Miscellaneous

- The Visible property specifies whether an item should be shown on screen.
- Validation events added: nodeCreating, linkCreating, nodeDeleting, linkDeleting, nodeModifying and linkModifying.
- The ScrollX and ScrollY properties and scrollTo method lets you set the diagram's scroll position.
- getViewport returns the currently visible part of the diagram.
- setZoomFactorPivot sets the zoom level in such way that the pivot point's screen position does not change.
- zoomToRect zooms and scrolls the view to fit the specified document rectangle in the Diagram's visible area.
- zoomToFit zooms the view to fit the document contents into the current view size.

- getContentBounds returns the smallest rectangle that bounds all diagram objects.
- keepGroupLayout indicates whether to treat each group of attached nodes as a single vertex in the arranged graph.

New in version 1.2

Alignment grid

The alignment grid help users place the diagram items more precisely. To activate the grid, set the AlignToGrid property to **true**. If the grid is active while an item is being created or modified, the item's control-points are aligned to the nearest grid points. The distance between adjacent grid points is set via the GridSizeX and GridSizeY properties. The grid can be either visible or invisible, depending on whether the ShowGrid value is **true** or **false**. If visible, the grid is painted as a matrix of points or as series of crossing lines, as specified by the GridStyle property. GridColor defines the color with which the alignment points or lines are painted.

Bordered tree layout

In contrast to TreeLayout, which centers the parent node above its child nodes, BorderedTreeLayout arranges children in columns at a horizontal offset form their parent (see Tutorial 2). **BorderedTreeLayout** members control the tree direction, how much space to leave between tree levels, and between nodes on the same level.

Anchor points

The AnchorPattern class represents a set of AnchorPoint objects, which specify the exact locations where links are allowed to connect to nodes. A pattern can be assigned to nodes by calling their setAnchorPattern method. The **AnchorPattern** class exposes several predefined patterns as static fields: decision2In2Out, decision1In3Out, leftInRightOut, topInBottomOut.

New in version 1.1

Automatic link routing

A link whose AutoRoute property is enabled automatically finds a route that won't cross nodes marked as obstacles. Call setRouteLinks to specify the default AutoRoute value of new links. Links can be explicitly routed by calling their route method or the routeAllLinks method of Diagram. The minimal distance between routed links and nodes can be set by calling setRouteMargin.

Fractal layout

The FractalLayout tree layout algorithm places child nodes symmetrically around their parent node.Nodes at the lowest level are arranged directly in a circle around their parent. At the upper level, the already arranged nodes form branches that are arranged in a circle around the new parent node. The algorithm is recursively repeated till the highest level is reached. To apply **FractalLayout**, pass its instance as argument to the arrange method.

Miscellaneous

- The Events class exposes available events as static members that can be used as arguments of addEventListener.
- nodeTextEdited and linkTextEdited events raised when the user edits the text of nodes or links.
- The mouse cursor changes when pointing an adjustment handles to indicate what kind of modification will occur.

Moving to version 4

To port a project that uses an earlier version of JsDiagram to v.4.0, follow these steps:

- For projects that use module loaders, rather than accessing the MindFusion namespace from the global namespace, check the various "Loading ... modules" sections in Getting Started guide.

- For TypeScript projects that access MindFusion namespace from the global namespace, add a reference to the global.d.ts file.

- diagramming.compat.d.ts contains the module augmentation for the diagramming module and enables using legacy get/set functions in your TypeScript code. Use it with a reference path as below, or otherwise moving it to the project's main folder may be necessary.

TypeScript

```
/// <reference path="./node_modules/@mindfusion/diagramming/diagramming.compat.d.ts" />
```

- The DOM control is now represented by the DiagramView class, rather than the Diagram class, and you will need to replace any **Diagram.create**, **Diagram.find**, **MindFusion.AbstractionLayer.createControl** calls with respective DiagramView.create and DiagramView.find calls. The **Diagram** model object can be created first and passed to the view as a second argument of the **DiagramView.create** method, or if not provided, an empty **Diagram** will be created and will be accessible through the view's diagram property. For TypeScript projects, a cast to <HTMLCanvasElement> will be necessary for the first argument.

- Most getProperty/setProperty functions have been replaced by getter/setter properties. In addition, some properties and methods of the **Diagram** class related to clipboard operations, zoom and scroll, inplace-edit and recording have been moved to the **DiagramView** class. Check the complete list of the moved properties and methods in version 4 API changes section and replace the calling instance, e.g.

JavaScript

```
// remove
diagram.setBehavior(MindFusion.Diagramming.LinkTables);

// add
diagramView.behavior = MindFusion.Diagramming.LinkTables;
```

You can continue using legacy get/set functions for properties defined in versions up to v.4.0 in your code by setting the propFunctions field of CompatConfig class to **true**.

JavaScript

```
MindFusion.Diagramming.CompatConfig.propFunctions = true;
```

If using that with TypeScript projects, also add a reference to @mindfusion/diagramming/diagramming.compat.d.ts.

- Auxiliary controls, such as Overview, ZoomControl and Ruler, must be attached to a **DiagramView** instance instead of **Diagram**.

JavaScript

```
// remove
overview.setDiagram(diagram);
ruler.setDiagram(diagram);
zoomControl.setTarget(diagram);

// add
overview.diagramView = diagramView;
ruler.diagramView = diagramView;
zoomControl.target = diagramView;
```

- The **MindFusion.AbstractionLayer** class has been removed. Custom node classes in ES6 projects should be created following the ES6 class pattern, e.g.

JavaScript

```
var TemplatedBase = Diagramming.CompositeNode.classFromTemplate("TemplatedBase", ...

export class OrgChartNode extends TemplatedBase
{
```

```
    _cost: string;

    constructor(parent) {
        super(parent);
    };

    getCost() {
        return this._cost;
    }

    setCost(value) {
        this._cost = value;
    }

    toJson() {
       var json = super.toJson();
       json.cost = this._cost;
       return json;
    }
}
```

- Custom node classes in ES5 projects should be created following the prototypal inheritance pattern, with the account for the removal of **AbstractionLayer** class, e.g.

JavaScript

```
var TemplatedBase = CompositeNode.classFromTemplate("TemplatedBase", ...

function OrgChartNode(parent) {
    var _this = TemplatedBase.call(this, parent);
    _this.cost = "low";
    return _this;
};

OrgChartNode.prototype.getCost = function () {
    return this.cost;
};

OrgChartNode.prototype.setCost = function (value) {
    this.cost = value;
};

OrgChartNode.prototype.toJson = function ()
{
    var json = TemplatedBase.prototype.toJson.call(this);
    json.cost = this.cost;
    return json;
};
```

- Registering a custom node type with the **Diagram** now also includes serialization identifiers and version:

JavaScript

```
// remove
AbstractionLayer.registerClass(OrgChartNode, "OrgChartNode", TemplatedBase);

// add
Diagram.registerClass(OrgChartNode, "OrgChartNode", "OrgChartNode", 1, TemplatedBase);
```

User Interaction

Most likely an application would use JsDiagram to let users interactively create diagrams, flowcharts and other kinds of presentation documents. A user composes or edits a diagram by drawing its elements on the JsDiagram canvas area. With the help of a mouse or another pointing device users draw shapes, tables

and links, possibly connecting them in complex hierarchies, graphs, trees or other structures they might need.

Creating New Items

New items can be created by drawing them with the mouse. While a user drags the mouse, the view of the control is updated to show what the new item would look like at the pointer location. If the operation cannot be completed at the current mouse position, the cursor changes to a crossed out circle. When the user releases the button, the item is created. By default links can be drawn only between nodes, unless the allowUnconnectedLinks property is enabled. Users can cancel item creation by pressing the right mouse button while drawing.

An alternative method for users to create new nodes is drag-and-drop from the NodeListView control. Nodes in the **NodeListView** are used as prototypes and are cloned when dropped on the diagram canvas.

Modifying Items

A user modifies an item by dragging its adjustment handles. The modificationStart property specifies when these handles appear. The first option, AutoHandles, makes the handles appear when the mouse hovers over an item; modification of the item can start immediately by pointing it and dragging a handle. The second option is **SelectedOnly**: in order to be modified, an item should first be selected. Adjustment handles appear around the item to indicate that now it can be moved or resized. Number and functionality of those handles depend on the item type and the value of the corresponding handlesStyle property.

Modifying a Node

By default nodes display nine adjustment handles. The one at the center can be used to move the node. When dragged, each corner handle resizes the node by moving the two edges adjacent to the corresponding corner. The edge handles placed at the center of each side can be used to resize a node either horizontally or vertically. Shape nodes can be rotated using an additional round handle drawn above them. Each handle can be enabled or disabled by setting or clearing the respective bit in enabledHandles.

Modifying a Link

Variety and behavior of link adjustment handles depend on the link's shape and segmentCount. Control-points stay at both ends of each segment that constitutes a link. MindFusion.Diagramming for JavaScript offers several link styles. Following sections discuss how to modify links of each style.

- The first style called Polyline defines links, which consist of a series of straight segments, drawn between control-points. There are no restrictions as to the number of segments. Dragging a control-point modifies each segment adjacent to the point. If the allowSplitLinks property is enabled, users are allowed to insert new control points or to remove existing ones. Inserting a control point is done by pressing the left mouse button over a segment and dragging. The new point appears at the mouse location and splits the pointed segment in two.
- If link's shape is set to **Bezier**, each segment has 4 manipulation handles that define a curve. Each curve goes through the end-points, which are shaped as circles. Bend of the curve is defined by the middle two points shaped as rectangles. If a user drags an end-point modification handle, the adjacent control points will be moved with same translation vector as that of the end-point. If a bend control point is dragged, the symmetric control point of the neighboring curve is also moved to ensure the two stay on the same line with the end-point and on the same distance from it.
- **Cascading** links consist of straight horizontal and vertical segments, each one orthogonal to its neighbors. If a control-point is dragged, the segments adjacent to it are moved and resized preserving their orthogonal orientation to each other. If the **allowSplitLinks** property is enabled, users are allowed to insert new control points to **Cascading** links or remove existing ones. In that mode, users can also change orientation of end segments by dragging next-to-last control point diagonally.

Selecting Items

Multiple items are selected by dragging while CTRL key is down. Selection can also be toggled with CTRL + clicking an item. A rectangle appears around selected items, allowing a user to move all of them at once.

Interpreting Users Actions

Mouse operations can be interpreted in different ways depending on the value of the behavior property. It defines whether for example dragging the mouse creates a node or selects existing items. Check the property description for more information.

Programming Interface Overview

The *Scripts/esm* folder from distribution contains ES6 code that lets you import JsDiagram classes from respective modules, which are also available as scoped *@mindfusion/diagramming* package on npm. There are also ES5-compatible scripts located in *Scripts/umd* folder, which are transpiled from ES6 code, and whose classes are exposed as members of global MindFusion namespace object.

The Reference section of this help file lists the classes under namespaces (i.e. ES5 compatible hierarchy) whose names you can easily translate to scoped module names when coding for ES6. E.g. if a class is listed under MindFusion.Graphs namespace, you can import it from respective @mindfusion/graphs package.

Model-View Architecture

JsDiagram implements the Model-View-Controller (MVC) architectural pattern starting from version 4. The MVC design pattern as applied to MindFusion.Diagramming separates the diagram data (model) from its representation to the user (view) and controlling it by the user (controller).

Single Diagram Canvas

The Diagram class represents the diagram data model and provides operations to process the data. It exposes properties and methods for adding new diagram elements, finding and accessing existing elements, or removing them.

The DiagramView class presents the relevant data from the **Diagram** class by drawing on the screen using the **CanvasRenderingContext2D** API. When the diagram changes, the view automatically redraws the affected part of the image to reflect the changes. Multiple views can display the same diagram, set by their diagram property. Each view can have a different zoom level and scroll position.

The **DiagramView** class creates Controller objects when user starts interacting with the diagram, depending on the value of view's behavior property. That property specifies how GUI actions to manipulate the data should be interpreted.

Multiple Diagram Pages

The DiagramDocument class represents a collection of diagram pages or sheets, represented by DiagramPage objects. New pages can be added to the document and existing pages can be removed and reordered by using the addPage and removePage methods respectively. The toJson, saveToXml and saveToString methods of **DiagramDocument** save all pages in a single file, and respectively the fromJson, loadFromXml and loadFromString methods load all pages from the file. **DiagramDocument** can also load files created by the serialization methods of Diagram, and will show them in a single page.

DiagramPage is derived from **Diagram** and only adds a few properties, so any code that processes **Diagram** objects will work with **DiagramPage** objects too. A diagram document forwards each event raised by its pages, giving you a chance to use the same handler for events raised by different pages. If a page should implement distinct user interaction or validation logic, it is also possible to attach handlers to the **Diagram** events inherited by **DiagramPage**.

A single **DiagramPage** could be assigned to the diagram property of DiagramView, and that lets you create your own user interface for presenting a list of pages to the user and selecting a current page. The package also includes a TabbedDiagramView that presents all pages of a document in a tabbed user interface where each tab corresponds to a page. The page displayed in the active tab is exposed by the selectedPage property of **TabbedDiagramView**. The active page can be changed interactively by activating its associated

tab. New pages can be added and removed to / from the document through buttons in the tab strip and the pages can be rearranged by dragging their associated tabs with the mouse.

User Interface Components

The following UI components from the library can be added to a page to let users create and view diagrams:

DiagramView

DiagramView renders the contents of a Diagram, and lets users draw new items by drawing with mouse or touch on the canvas. The behavior property indicates how the control should interpret users' actions. Additional properties allow scrolling and zooming the view.

TabbedDiagramView

TabbedDiagramView displays multi-page DiagramDocuments. Each DiagramPage is displayed in its own tab. Apart from base DiagramView members, the control includes properties for customizing the tab strip, and to specify whether users are allowed add or remove pages.

Palette

The Palette control represents a tool palette that displays DiagramItem objects grouped into categories, and is implemented as accordion UI. Call the addCategory method to define a category and create its respective accordion pane (an ItemListView). Call addItem to add an item to specified category. **Palette** provides same layout and appearance properties as the **ItemListView** class.

ItemListView

ItemListView represents a tool palette that contains uncategorized list of diagram items. Items can be added to the list by calling addItem method, and removed by calling removeItem. Drag-and-drop operations from **ItemListView** to DiagramView create a copy of the dragged item, and set its size to newInstanceSize.

Overview

Overview provides a scaled-down view of a target DiagramView, helping users work with larger diagrams. A tracker rectangle shows current viewport of the main **DiagramView**, and by dragging it users can change the scroll position. It can also be resized to zoom the target view.

ZoomControl

ZoomControl lets users change the zoom level of a target DiagramView by dragging a slider. It also includes arrow buttons for scrolling the view.

Ruler

The Ruler control displays horizontal and vertical scales that let users measure and align diagram nodes.

ShapeDesigner

ShapeDesigner control lets users define shape geometries. Shape definitions can be saved as a shape library XML file, loaded by application using the ShapeLibrary class, and accessed by calling fromId method by the Shape class.

Diagram Structure

Standard Item Types

Diagram elements in MindFusion.Diagramming for JavaScript are represented by DiagramNode and DiagramLink instances. Combined and linked together, they can form complex data structures such as process or entity-relationship diagrams, class hierarchies, networks and graphs. Each of their rich features can be customized through numerous methods and properties.

Shape Nodes

ShapeNode objects can represent geometric shapes, flowchart elements, icons in network diagrams, and so on. They can display both graphics and text in different fonts, sizes and colors. A node's shape is specified by setting its shape property. There are more than 100 predefined node shapes and apart from them, MindFusion.Diagramming for JavaScript offers an easy way to define custom shapes. Shape nodes can be rotated, either interactively using the round selection handle drawn above them, or programmatically by setting their rotationAngle property.

SVG Nodes

The SvgNode class represents nodes that can display SVG drawings. SVG graphics are rendered instead of image, and on top of the geometry rendered by the base ShapeNode class. Enable the transparent property to hide the node's geometry and leave only the drawing visible. The SVG file is loaded and associated with the node through an SvgContent object, which should be initialized and then assigned to the content property.

Composite Nodes

Nodes in MindFusion.Diagramming can be composed from components. Components are graphical widgets similar to controls, which can be arranged in hierarchies and added to a Diagram through instances of the CompositeNode class. The component system contains various built-in components, such as Text and Image. Component hierarchies can be defined and loaded through Json templates using loadTemplate and classFromTemplate methods.

Table Nodes

A TableNode displays tabular data, and can be used to represent table schemas in database diagrams or classes in class-hierarchies. A table contains cells arranged in a grid, and every cell can display text and/or image. Cells, columns and rows can be customized in various ways - there are properties available for setting their extents, style, image and text alignment. Tables' dimensions can be programmatically changed; inserting, appending or deleting rows and columns can be done in a very comprehensive, intuitive way whenever you might need it. Tables can also display background images and graphics.

Free-form Nodes

A FreeFormNode collects all points from users' mouse or touch input and displays them as node's outline. To let users draw free-form nodes interactively, set behavior to DrawFreeForms. Use the points property of **FreeFormNode** to get or set outline points programmatically. If the closed property is enabled, the node is drawn as a closed shape and its interior filled, or otherwise the node is drawn as a poly-line. If the distance between first and last points drawn by user is shorter than autoCloseDistance, the node gets closed automatically.

Container Nodes

A ContainerNode can contain other diagram nodes. When a container is moved, its child nodes move too. Containers can be nested within each other to unlimited depth. They can be folded to a smaller size, hiding their child items, and unfolded to show content again. Nodes inside a container can be linked to nodes

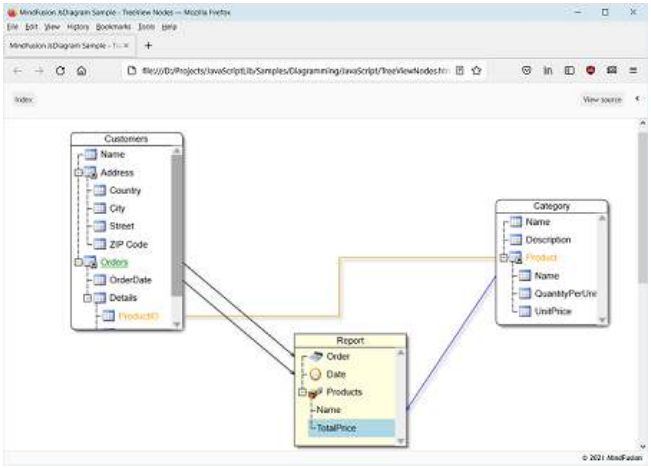
both inside and outside of it. If a link connects items in the same container, the container is not considered an obstacle by the auto-routing algorithm.

Paged Containers

PagedContainerNode is a ContainerNode subclass that organizes its child nodes into a collection of pages, represented by ContainerPage objects. This allows for the creation of more orgainzed diagrams where nodes can be grouped into logical views within a single container. Only the nodes belonging to the currently active page, specified by currentPage property, are rendered and participate in user interactions.

TreeView nodes

The TreeViewNode class represents nodes that can display hierarchical data. The root items displayed in the node can be accessed through the rootItems property. Items can be added and removed individually by using the addItem and removeItem methods, or in bulk by calling the fromObject method, which loads the tree view items from an array of objects.



Video Nodes

VideoNode objects display video streams, along with UI elements to play, pause, seek and change volume. Set node's videoLocation property to specify its video URL. Auto-playing is not supported at this time, and the video can be started only by users clicking the play button. **VideoNode** is built around Video component, which can also be used in CompositeNode templates.

Control Nodes

ControlNode objects display custom HTML content, specified via their template property. A node's DOM structure is created inside the Content div element, which is then rendered on top of the diagram canvas. Note that this prevents **ControlNode**s drawing in diagram' usual Z order, and they will always appear on top of other diagram items. By default the diagram intercepts mouse events of all hosted html elements to enable moving nodes or drawing links. You can specify that interactive elements, such as buttons or text boxes, should receive input instead by setting their **data-interactive** attribute to **true**.

Links

DiagramLink objects are used to represent various types of connections and relations in diagrams. MindFusion.Diagramming for JavaScript offers three basic link shapes, which can be assigned to links' shape property:

- polyline links composed of straight-line segments;
- curved links composed of Bézier-spline segments;
- cascading links composed of horizontal and vertical segments, alternating their orientation so that every segment is orthogonal to its neighbor.

A link can be connected to a node at each of the link's ends. It is also possible to have unconnected links - having one or both of their ends not connected to nodes. Some customizable link's properties are: arrowheads, style, text, pen width and style, modification behavior and number of segments.

See Also
Controlling User Interaction

Composite Nodes

The CompositeNode class makes it easier to add custom graphics primitives to nodes. Appearance of composite nodes is defined by a tree of layout panels and child components. Unlike MindFusion's desktop diagram libraries, node templates are specified using JSON notation instead of XML. A sample node class derived from **CompositeNode** can be defined declaratively as below.

JavaScript

```
var OrgChartNode = CompositeNode.classFromTemplate("OrgChartNode",
{
 component: "GridPanel",
 rowDefinitions: ["*"],
 columnDefinitions: ["22", "*"],
 children:
 [
  {
   component: "Rect",
   name: "Background",
   pen: "black",
   brush: "white",
   columnSpan: 2
  },
  {
   component: "Image",
   name: "Image",
   autoProperty: true,
   location: "icon4.png",
   margin: "1",
   imageAlign: "Fit"
  },
  {
   component: "StackPanel",
   orientation: "Vertical",
   gridColumn: 1,
   margin: "1",
   verticalAlignment: "Near",
   children:
   [
    {
     component: "Text",
     name: "Title",
     autoProperty: true,
     text: "title",
     font: "Arial bold"
    },
    {
     component: "Text",
     name: "FullName",
     autoProperty: true,
     text: "full name",
     pen: "blue",
     padding: "1,0,1,0"
    },
    {
     component: "Text",
     name: "Details",
     autoProperty: true,
     text: "details",
     font: "Arial 3"
```

```
    }
   ]
  }
 ]
});
```

Objects from the MindFusion.Drawing namespace, such as Rect, Image, Path, Video, double as **CompositeNode** components. You can create custom components by deriving from ComponentBase and assigning a name to the custom class by calling registerComponent. Apart from JSON templates, the visual tree can also be created by adding panels and drawing objects programmatically starting from CompositeNode.root.

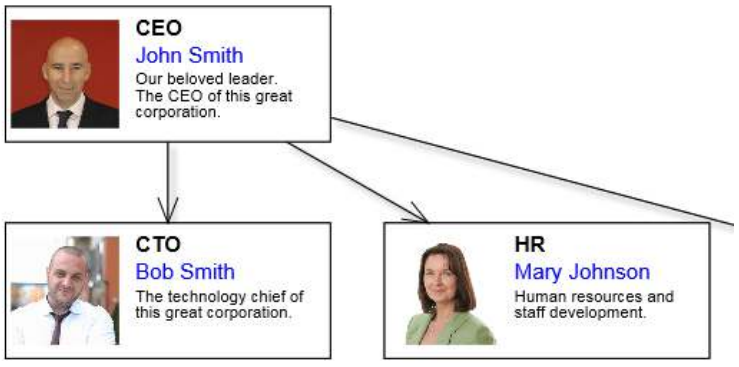
Panel classes include:

- SimplePanel, laying out its children to overlap each other, taking up all available space (useful for background graphics or images).
- StackPanel, laying out its children side by side, either horizontally or vertically.
- GridPanel, laying out its children in rows and columns, where children in same cell overlap. Rows and columns can be either set to a fixed size, auto-size to their child preferred size, or take up all space remaining unoccupied by other rows and columns.

Named components (with specified name value) can be accessed by calling the getComponent method. Components whose autoProperty attribute is enabled will have a pair of get/set property methods generated automatically by classFromTemplate method. The auto-properties from above template can be used as below.

JavaScript

```
var node1 = new OrgChartNode(diagram);
node1.setBounds(new Rect(25, 15, 60, 25));
node1.setTitle("CEO");
node1.setFullName("John Smith");
node1.setDetails(
    "Our beloved leader. \r\n" +
    "The CEO of this great corporation.");
node1.setImage("ceo.png");
diagram.addItem(node1);
```



Auto properties are also automatically serialized in JSON format, cloned by node.clone() method and saved/restored by undo commands.

The Shape class used to specify ShapeNode geometry can also be used as a component. When its **isOutline** attribute is set, the shape will control **CompositeNode**'s geometry too, defining hit test and clip area, and link alignment points along node's border. If **isOutline** is disabled, the shape will serve mostly as a decorative element. The code below shows a sample fragment specifying shape component in template:

JavaScript

```
{
    component: "Shape",
    id: "Cloud",
    autoProperty: true,
```

```
    name: "OutlineShape",
    pen: "gray",
    brush: "Transparent",
    isOutline: true
},
```

A Button component that raises event when clicked can be added to template like this:

JavaScript

```
{
    component: "Button",
    brush: "#ccc",
    text: "Delete",
    width: 30,
    cornerRadius: 3.5,
    clickHandler: "onDeleteClick"
}
```

Creating and Deleting Items

New items can be added programmatically to the diagram by calling the constructors of the item's class, such as ShapeNode or DiagramLink, and adding the new instance via the addItem method. A shortcut to creating new items is provided by the methods of the Factory class. An instance of that class is returned by the factory property of Diagram. Factory methods such as createShapeNode and createDiagramLink accept respectively node position or link's origin and destination arguments, create an instance of the class and add it to the diagram. The **Factory** class can create instances of JsDiagram predefined item types, while **addItem** can be used to add instances of custom types as well.

JavaScript

```
var ShapeNode = MindFusion.Diagramming.ShapeNode;
var Rect = MindFusion.Drawing.Rect;

function initDiagram(diagram)
{
    // create the start node
    var startNode = diagram.factory.createShapeNode(10, 10, 30, 20);
    startNode.text = "start";

    // create the end node
    var endNode = new ShapeNode(diagram);
    endNode.bounds = new Rect(10, 60, 30, 20);
    endNode.text = "end";
    diagram.addItem(endNode);

    // connect the nodes
    diagram.factory.createDiagramLink(startNode, endNode);
}
```

If an existing item should be removed from the diagram, call the removeItem method. The clearAll method clears diagram contents by deleting all items.

By default, deleting a node also removes its links. Set the autoDeleteLinks property to **false** to prevent that, and disconnect the links instead.

Accessing Items

Items in the diagram are exposed via its nodes and links members. **nodes** contains all DiagramNode - derived objects that have been added to the diagram, and **links** contains all DiagramLink objects. By iterating these arrays, you can access all diagram elements and apply common attributes to them:

JavaScript

```
// paint all nodes with light blue color
for (var node of diagram.nodes)
{
    node.brush = "lightBlue";
}

// set the links' arrowheads to triangles
for (var link of diagram.links)
{
    link.headShape = "Triangle";
}
```

Individual items provide methods that let you access their related items in the diagram. **DiagramNode** exposes the links coming into or going out of nodes via the incomingLinks and outgoingLinks properties. The nodes connected by a link can be accessed by means of the origin and destination properties of **DiagramLink**.

The component lets users select multiple diagram items of interest. The selection method of Diagram returns a Selection instance representing currently selected items. The **Selection** class provides nodes and links arrays containing respectively **DiagramNode** and **DiagramLink** -derived objects.

Collapsing and Expanding Tree Branches

If a diagram represents a tree structure, the tree branch starting from a node can be collapsed or expanded again in order to hide or display hierarchy details. To enable this, set the expandable property. To get or set whether a tree branch is expanded or collapsed, use the expanded property of its root node.

Displaying Expand/Collapse Button

If the expandable property of a node is set to true, there is a [±] button displayed beside the node to allow users to collapse or expand the tree branch starting there. By default, pressing that button automatically toggles the expanded property of the node and raises the treeExpanded or treeCollapsed events.

Defining Tree Direction

A node can be connected to other nodes through outgoing and incoming links. Setting the expandOnIncoming property specifies which items connected to a node will be regarded as its descendants. If **expandOnIncoming** is **false** (default value) then each node will be regarded as a parent of its outgoing links and the items attached to them. If **expandOnIncoming** is set to **true**, the node's incoming links define its children. So, this property specifies the direction of parent-children relationship between diagram items and respectively the overall direction in which collapsing / expanding is done.

Attaching and Grouping Items

MindFusion.Diagramming for JavaScript allows attaching a node to another node, establishing a subordinate - master relationship between them. The easiest way to attach nodes is to call the attachTo method. When a master node is moved, its subordinated nodes follow it, so that the initial distance between the nodes stays constant. This happens both when a user moves the master node around, and when the node position is changed using a method or a property.

Note

Recent versions of the API include NodeLabel and LinkLabel classes, which address several of the scenarios described below. We recommend using these classes when appropriate, as they offer a more lightweight and easily configurable alternative to attached nodes.

Possible Uses

- Attach a transparent shape node to another node and set its text, so that the shape node acts as an additional label assigned to the master node.
- Attach a shape node to another node and set its image, so it acts as an additional icon displayed inside or near the master node.
- Attach a node to another node and handle nodeClicked or similar events to perform some action on the master node. This makes the subordinated node act as a button or another type of UI element attached to the master node.
- Attach child elements in a tree branch to the branch root, so that when a user moves a tree node, the node's children moves simultaneously with it. Modifying an item in such hierarchical group makes the modification pervade through all dependant subordinated items.
- In all previous scenarios, the locked property can be set for attached nodes which are used only as decorations embellishing the main item. That prevents users from selecting, modifying or drawing links to the subordinate nodes.

Hierarchical Grouping

Every node in a group can be the master node of another group, that is, a node can be subordinate in one group and meanwhile be another group's master. MindFusion.Diagramming for JavaScript doesn't impose any limits to the depth and width of group hierarchies built that way. However, it is not possible to put a node as a subordinate in more than one groups because the hierarchy relations would become ambiguous.

Unions

It is possible to create such groups, that if any item in the hierarchy is modified, all other items are updated too. This can be done by putting the items in circular chain of dependencies - that is, item1 is attached to item2, item2 to item3 ... itemN to item1. If any of these items is modified, the change will propagate through the chain until it reaches its originator.

Accessing Group Items

The master node in a group can be accessed using the masterNode property of attached nodes. The array returned by attachedNodes getter on a master node contains references to all subordinate nodes attached to it.

User Interaction

Diagram Events

The diagram raises various events to notify you of the effects of user's actions, such as nodeCreated, nodeModified, nodeDeleted, etc. Events are represented by the EventDispatcher class and are accessible as properties of the corresponding objects (Diagram or DiagramView). Event listeners are attached through the addEventListener method and removed through the removeEventListener method.

The following sample code attaches an event listener to the **nodeCreated** event of a Diagram control. The variable diagram identifies a **Diagram** instance and **onNodeCreated** is the name of the event handler function:

JavaScript

```
diagram.nodeCreated.addEventListener(onNodeCreated);
```

All event handlers receive two arguments. The first '*sender*' argument specifies the object that emits the event, and the second '*args*' argument is an instance of an EventArgs -derived class:

JavaScript

```
function onNodeCreated(sender, args)
{
    // ...
}
```

While some events are raised to notify you that the user has performed an action, like the **nodeCreated** event, validation events let you accept or reject user actions before actually changing the diagram. Validation event names usually end in '-ing' suffix, such as nodeCreating. The *args* parameter passed to their handler functions is an instance of a CancelEventArgs -derived class. Validation event handlers are expected to set the value of args' cancel property either to **true** if it should be rejected or **false** (the default) if it should be accepted.

The following code handles the nodeDeleting event and, if the node to be deleted is a TableNode instance, cancels the action:

JavaScript

```
diagram.nodeDeleting.addEventListener(
    (sender, args) =>
    {
        args.cancel = args.node instanceof TableNode;
    });
```

Controlling User Interaction

There are several ways to control user actions on the diagram as a whole, or on individual items in the diagram.

Drawing Modes

The behavior property of DiagramView provides several predefined sets of rules specifying how to interpret user input through pointing device. The default **behavior** is LinkShapes, where drawing with mouse or touch from an unoccupied point of the document is processed as a ShapeNode creation request, and drawing from a node creates a DiagramLink. Additional members of the **Behavior** enumeration provide modes for drawing different node types, and ones that allow only limited interactions, such as **SelectOnly** or **Modify**.

Item Constraints

Constraints upon how users are allowed to modify a node are set through the constraints and enabledHandles properties. The former can be used to constrain the move direction, node size and aspect ratio. The latter enables or disables specific manipulation handles of a node. That allows making the node non-moveable, non-resizable, or resizable only horizontally/vertically.

Items can be locked by setting their locked property. Locked items are excluded from selection and users cannot select, move or modify them.

In-place Text Editing

The text of nodes and links can be edited in-place by double-clicking the respective items. To enable this feature, set the allowInplaceEdit property to **true**. Depending on the type of item, the control raises either the nodeTextEdited, cellTextEdited or linkTextEdited event when its text has been edited. In-place editing mode is exited when users click outside the text entry control. Pressing ENTER or ESC exits inplace-edit mode if respectively inplaceEditAcceptOnEnter and inplaceEditCancelOnEsc are enabled.

By default, the control creates a **textarea** DOM element over an item to allow editing the item's text. If you prefer using a different type of element as in-place editor, handle the createEditControl event and set the control property of the provided InplaceEditEventArgs object.

Validation Events

While the user draws new nodes or links, the control raises the nodeCreating or linkCreating events after each change of the mouse position. By setting the cancel property of the event object, you can prevent the operation from being completed at the current position but let the user continue drawing. Calling the cancelDrag method immediately stops the operation. The nodeModifying and linkModifying events provide the same functionality during modification of existing objects. Finally, if the user tries to delete an item, you can handle the nodeDeleting or linkDeleting event to prevent this.

Interaction Properties

- allowSelfLoops property of Diagram specifies whether users can draw reflexive / self-loop links.
- allowUnconnectedLinks property specifies whether users can create links without connecting them to nodes (e.g. for simple line or curve graphics in a drawing).
- allowLinksRepeat specifies whether user can draw a second link between same pair of nodes.
- delKeyAction specifies how the DiagramView handles the *Delete* key.
- modifierKeyActions allows changing the function of modifier keys.
- leftButtonActions, middleButtonActions and rightButtonActions allow mapping interaction types to mouse buttons.
- mouseWheelAction allows changing the function of the mouse wheel.
- allowIncomingLinks and allowOutgoingLinks properties of DiagramNode specify if a node accepts respective types of links.
- setting the locked property of a DiagramItem prevents users from interacting with it altogether.

Multi-touch Support

The control handles DOM Pointer events to implement multi-touch interactions, such as zooming, node rotation or simultaneous drawing of multiple diagram items:

- If multiTouchZoom property is enabled (default), the view can be zoomed or panned using two-touch pinch / flick gestures.
- If multiTouchModify property is enabled (default), diagram nodes can be moved, scaled and rotated using two-touch pinch / flick gestures.
- If **multiTouchZoom** property is disabled, each touch draws diagram items corresponding to current behavior.
- If **multiTouchModify** property is disabled, each touch started from a node draws a diagram link.

Latter modes can be used for collaborative whiteboarding / classroom scenarios.

Built-in multi-touch can be disabled altogether by setting enableMultiTouch to **false**, e.g. if you need to handle touch events yourself or through a third-party gesture library. If disabled, the control will revert to its mouse event handling code from previous versions. It will also no longer be able to capture mouse input during drag operations (to detect mouse pointer leaving the view), which is handled through DOM Pointers API.

User-interaction controllers

All kinds of user interaction are carried out by controller objects. Built-in controllers include CreateNodeController, CreateLinkController, ModifyNodeController, ModifyLinkController, PanController. You can also create custom controllers by deriving and overriding the methods of SinglePointerController:

JavaScript

```
SinglePointerController.prototype =
{
    start: function (position) {},
    move: function (position) {},
    validate: function (position) {},
```

```
    commit: function (position) {},
    cancel: function (position) {},
    drawInteraction: function (context) {}
}
```

The controller instance used by the diagram when interaction starts is returned by createController method of BehaviorBase -derived classes. You can replace the method of standard behavior classes with a custom function to return a different controller, or inherit a new class from **BehaviorBase** that implements it and assign it to the **currentBehavior** property of DiagramView.

Alignment Grid

MindFusion.Diagramming provides an alignment grid to help users place the diagram items more precisely. To activate the grid, set the alignToGrid property to **true**. If the grid is active while an item is being created or modified, the item's control-points are aligned to the nearest grid points. The distance between adjacent grid points is set by the gridSizeX and gridSizeY properties. The gridOffsetX and gridOffsetY properties allow specifying the horizontal and vertical offset of the first point in the alignment grid.

Items created or modified by a user are automatically aligned to the grid. However, if items are placed in the diagram programmatically, their coordinates are set to the specified positions without aligning. In such cases use the alignPointToGrid method if items must be placed exactly at grid points.

The grid can be either visible or invisible, depending on whether the showGrid property is enabled. If visible, the grid is painted as a matrix of points or as series of crossing lines, as specified by the gridStyle property. gridColor defines the color with which the alignment points or lines are painted.

Undo and Redo

If the undoEnabled property is set to **true**, the control records users actions and allows undoing them later. The records are stored in a list, and can be undone consecutively by calling the undo method. An action that has been undone can be redone by calling the redo method. You can enable or disable UI elements that undo or redo commands by binding them to the **canUndo** and **canRedo** properties.

The control also runs and saves in the undo history AddItemCommand and RemoveItemCommand commands when items are programmatically added or removed from the Items collection of the Diagram. No other changes are recorded automatically when done from code. If support for undo/redo of such changes is required, they must be done through the executeCommand method.

To allow undoing changes of DiagramItem properties, create a ChangeItemCommand instance before setting the property values, and call **executeCommand** afterwards. Any other types of undoable changes can be implemented by deriving from the Command class and overriding its execute, undo and redo methods.

Scrolling and Zooming

The DiagramView control implements scroll and zoom support. If diagram's size is larger than current viewport, scrollbars appear and let the user scroll to clipped parts of the diagram.

Diagram Size

The diagram size is set through the bounds property. Its value is a Rect instance that specifies the origin of the diagram coordinate system and the diagram size. Scrollbar ranges also depend on the current zoomFactor.

Automatic Scrolling

The autoScroll property controls automatic scrolling. If it is enabled and the user drags an adjustment handle near the DiagramView edges while manipulating an item, the control will scroll the viewport towards the mouse position. The scrollZoneSize property defines the size of the area near the view borders that triggers automatic scrolling.

Programmatic Scrolling

In order to scroll the document content programmatically, you can call the scrollTo method or set scrollX and scrollY properties. The latter can also be used to access the current scroll position in case you need it. The bringIntoView method scrolls the view to ensure that a specific diagram item can be seen inside the viewport.

Zooming

Zooming is controlled via the zoomFactor property. The zoom level is expressed as a percentage, 100% meaning no zoom. Higher than 100% makes the diagram items look closer and bigger on the screen - zooming in. Less than 100% makes items look further and smaller on screen - zooming out. To zoom and scroll the view to fit the diagram contents to the control's client area, call zoomToFit. To zoom and scroll the view so it fits a specified rectangular document area to the control's client area, call zoomToRect.

Lasso Zoom Tool

The control supports several ways to zoom using lasso tool:

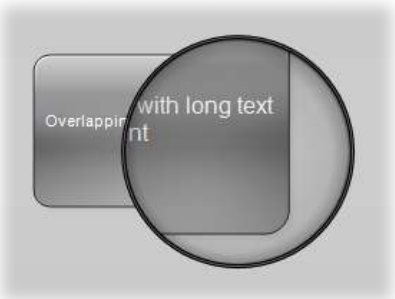
- Set the behavior property to Zoom to always draw zoom lasso.
- Set the Zoom flag for one of the keys in modifierKeyActions to let users draw zoom-lasso with respective modifier key pressed and the left mouse button.
- Assign Zoom value to rightButtonActions or middleButtonActions property to zoom using respective mouse button.

Zoom control

The ZoomControl class lets user change interactively the current zoom level and scroll position of a DiagramView. To set it up, add a <canvas> element to the page (possibly placing it on top of **DiagramView**'s one), call ZoomControl.create to instantiate, and set the control's target property to that view. Set the zoomStep and scrollStep properties to specify the amount added to view's zoom level or scroll position by **ZoomControl**'s buttons. You can customize the control's appearance by setting properties such as fill, borderColor, cornerRadius and tickPosition. For an example, see the *Controls.html* and *Controls.js* sample files from distribution.

Magnifying

The DiagramView provides an option for zooming in (or out) portions of the diagram through the Magnifier tool. The magnifier tool can be permanently activated by setting magnifierEnabled to **true**. Alternatively, on-demand magnification through the right mouse button or by pressing a modifier key can be enabled be setting the appropriate value to the rightButtonActions or modifierKeyActions properties respectively.



The magnification factor can be specified through the magnifierFactor property. The size of the tool and its border thickness can be set through the magnifierWidth, magnifierHeight and magnifierFrameThickness

properties.

Panning

The diagram view can be panned by holding ALT while dragging the mouse with the left button down. Another modifier key can be used for panning by setting modifierKeyActions. The middleButtonActions and rightButtonActions properties let you use the middle or right mouse button to start panning. To let users pan the view with the left mouse button without using a modifier key, set the behavior property of DiagramView to Pan or **PanAndModify**.

Testability

User interaction can be tested automatically in several ways outlined below.

- Create a MouseInputDispatcher instance and call its methods

  JavaScript

  ```
  onMouseDown: function (position, button)
  onMouseMove: function (position)
  onMouseUp: function (position, button)
  ```

  where position is a Point instance specifying logical coordinates in the diagram (in current measure unit, not scaled by zoom factor). The results of called mouse methods will depend on diagram's current behavior, e.g. you could assert that a sequence of down, move andup events creates a ShapeNode when **behavior** is set to DrawShapes.
- Create an instance of controller class and call its start, move and commit methods. This would not depend on the value of diagram's **behavior**, but could be still affected by other behavioral properties such as allowSelfLoops and allowUnconnectedLinks.
- Check assertions after replaying a recording on a saved diagram, using record, stopRecording and replay methods.

Placement and Layout

Items Placement and Z-order

Measure Unit

The measureUnit property specifies the unit of measure for all location and size-related values assigned to JsDiagram properties or passed as arguments to its methods.

Location and Size of Nodes

Set the bounds method to change the location and size of a node. Its values are Rect objects that contain coordinates and dimension properties. The **bounds** getter lets you find the current location and size of a node as set by users or layout algorithms.

Links Control Points

Links must have two or more control points, depending on their shape property. You can get access to the control points via link's controlPoints collection. You must call updateFromPoints after programmatically moving control points of a link; that's needed in order to repaint the link and update its internal state variables, such as arrowhead angles.

Link Dynamics

The dynamic property of a link can be enabled to make the link automatically recalculate its end point positions if the nodes connected to the link are moved or resized; the end points are moved in such a way that the link points to the centers of the connected nodes. Links also expose an autoRoute property that enables or disables automatic routing for a link. Routed links avoid nodes by finding the shortest path between their origin and destination that does not cross any other nodes.

Interactive Moving or Resizing

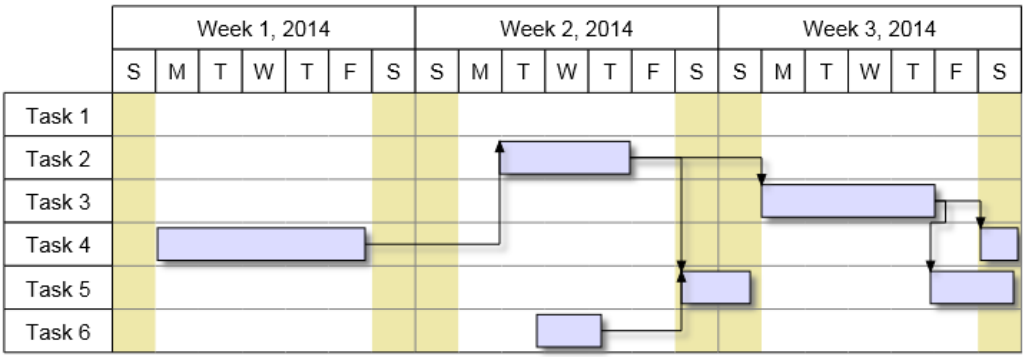
End-users of your application can move or resize the diagram elements. They do this by selecting an item and dragging any of its adjustment handles, a point inside its interior, any of its sides or a selection frame. The handlesStyle property of nodes controls how interaction can be performed. Additionally it defines the representation of selected items on screen. enabledHandles lets you disable specific manipulations users are allowed to perform on a node, for example resizing the item horizontally or moving it. If an item is modified interactively, the nodeModified or linkModified event is raised, depending on the item's type.

Controlling Z-order

The Z-order defines how items are ordered along the Z-axis, so that one item appears in front of or behind another one. Diagram items are sorted by the value of their zIndex property, and rendered on screen in ascending order. You can change the Z index of a DiagramItem by setting **zIndex** (values don't have to be consecutive).

Lane Grid

The classes from the MindFusion.Diagramming.Lanes namespace let you develop process maps and lane diagrams, useful in business process planning and reporting. Processes consist of consecutive operations accomplished by people involved, technical installations, means of contact, and flow of information. The MindFusion.Diagramming for JavaScript lane grid features allow indicating and arranging related diagram items in complicated processes and systems, that is, Swim lane diagrams (known as Rummler-Brache diagram), Gantt Charts (planning and scheduling of projects). A lane in the grid can visually contain nodes, links, shapes or images, that represent tasks fulfilled in specified periods of time by people, companies or departments.



The **MindFusion.Diagramming.Lanes** namespace provides the Grid, Header, Cell and Style classes, and the ResizeType enumeration. These types expose additional properties and methods that let you define your own grids as required by your application logic. Grids can include varying number of lanes, divided into child rows and columns whose intersections form cells. The cell color and style can be customized too.

Use the showLaneGrid property of the Diagram class to display lanes in your application. The laneGrid lets you setup the rows and columns in the grid by defining horizontal and vertical lanes, where each lane can be titled by its own header and/or subheadings. The grid can be resized by setting the columnCount and rowCount properties or by adding headers via the addColumnHeader and addRowHeader methods. A column or row can be divided into child elements by adding subheaders, which lets you form a complex hierarchy of unlimited depth. The intersection of rows and columns can be either a simple cell for bottommost elements in the hierarchy, or a group of cells for parent rows and columns. Cells or cell groups

can be accessed via the **Grid**'s get method, that lets you index cells either by their integer positions, or by their **Header** objects.

The position of the grid and grid headers within the diagram is defined via topMargin and leftMargin, which specify the grid offset relative to the diagram bounds. hookHeaders enables lane headers to be anchored to the left or right side of the control's visible area. The sizes of headers at different levels in the lane hierarchy can be set in a single operation by setting rowHeadersWidths and columnHeadersHeights properties.

The appearance of the lanes grid can be customized further via the headersOnTop poperty, which specifies whether headers should appear on top of the Z-order, and the topLeftAreaStyle specifying the appearance of the grid's top-left corner.

The **Grid** class provides many miscellaneous properties and methods that can be used to find grid elements, for determining their bounds and for hit-testing: findColumn, findRow, getCellBounds, getCellColumn, getCellFromPoint, getCellRow, getColumn, getRow, getRowIndex, columnHeaderBounds, getColumnIndex, getHeaderBounds, getRowHeaderBounds, getHeaderFromPoint.

Headers in the lanes grid can be customized by using the following members of the **Header** class:

Member name	Description
height	Gets or sets the height of the header and the related row in the grid.
resizeType	Gets or sets the resize type of the header.
rotateTitle	Gets or sets a value indicating whether rotating of the header caption at 90 degree is enabled.
style	Gets or sets the style of the header.
subHeaders	Gets a collection with the subheaders of the header.
title	Gets or sets the header caption.
width	Gets or sets the width of the header and the related column in the grid.

Anchor Points and Patterns

Anchor Points

For each node you can define one or more points to which links should dock when connecting to that node. Such points are represented by instances of the AnchorPoint class. The position of an anchor point relative to node's bounding rectangle is specified via the x and y properties. Their values are expressed as percentage of the node's width and height to accommodate for moving and resizing.

You can control whether an anchor point accepts incoming or outgoing links or both. That is done via the allowIncoming and allowOutgoing properties. When a link is drawn, its ends are aligned to the nearest anchors allowing docking of the respective end type. Another way to control whether links can be connected to an anchor point is to handle the validateAnchorPoint event.

Anchor Patterns

An AnchorPattern instance defines a set of anchor points to be used together. When creating a pattern, pass to its constructor an array of AnchorPoint objects. The **AnchorPattern** class exposes several predefined patterns as static properties. A pattern can be assigned as an integral entity to the anchorPattern property of nodes. Setting the anchorPattern property of a Shape object lets you define the

connection points for all ShapeNodes of that shape. The following example creates a pattern of four anchor points:

JavaScript

```
var AnchorPattern = MindFusion.Diagramming.AnchorPattern;
var AnchorPoint = MindFusion.Diagramming.AnchorPoint;

var apat1 = new AnchorPattern([
    new AnchorPoint(50, 0, true, true),
    new AnchorPoint(100, 50, true, true),
    new AnchorPoint(50, 100, true, true),
    new AnchorPoint(0, 50, true, true)
    ]);
```

The points are located at the middles of bounding rectangle sides of nodes to which the pattern is assigned. All four points accept both incoming and outgoing links.

Link Routing

Routed links avoid nodes marked as obstacles by finding the shortest path between their origin and destination that does not cross any other nodes.

Setting Up a Link Router

The control finds paths for links affected by modifications of nearby nodes by calling methods of an object that implements the Router interface. This object can be get or set through the linkRouter property of the Diagram class. JsDiagram provides several classes that implement **Router** - PatternRouter, GridRouter and CompositeRouter. By default **linkRouter** is set to a **CompositeRouter** instance.

Automatic or Explicit Routing

The routeAllLinks method can be invoked to explicitly rearrange links in a way they do not cross nodes. In addition, DiagramLink defines an autoRoute property that enables or disables automatic routing of a link, applied when user moves link's adjacent nodes or draws a new node in its vicinity. **autoRoute** is initialized with the value of diagram's routeLinks property.

Tuning the Routing Algorithm

Router classes define properties that let you fine-tune their specific algorithms. For a reference, see specific CompositeRouter, PatternRouter or GridRouter.

Performance Consideration

The default router is designed to produce visually optimal link paths that avoid overlapping with nodes and minimize bends. However, it is computationally intensive and can impact performance in diagrams with a large number of links or frequent layout changes. For improved performance, you could set linkRouter to a simple Router instance. This will reduce CPU usage and improve responsiveness, though the resulting link paths may be less visually refined.

Laying out the Diagram

The diagram can be arranged automatically by calling its arrange method and specifying a layout algorithm. Different layout algorithms are implemented by the various Layout classes from the MindFusion.Graphs namespace. To apply one of them, create an instance of the respective class, customize its properties and pass it as an argument to **arrange**. It is also possible to lay out the contents of an individual ContainerNode by calling the container's arrange method.

Layered Graph Layout

The LayeredLayout algorithm arranges diagram nodes in layers according to several criteria, most important of which are: connected nodes must be placed close together; links must flow in one direction if possible; links must cross as few layers as possible; links must not cross other links. To apply the layout to a diagram, create a **LayeredLayout** instance, set its members and invoke the arrange method.

Spring-Embedder Graph Layout

Spring-Embedder is force-directed layout algorithm. It simulates a physical system in which nodes repulse each other, and the links between them act as confining springs. Graphs processed by Spring-Embedder have their nodes distributed uniformly across the diagram area. The algorithm can be applied on a graph by creating a SpringLayout object and calling the arrange method. Members of **SpringLayout** allow you to set the desired distance between graph nodes and the number of iteration steps to run while processing the simulation.

Tree Layout

If application's data is structured hierarchically, it is appropriate to apply tree layout on it. To do this, create a TreeLayout or BorderedTreeLayout instance and call the arrange method. The difference between two classes is where they place child nodes in relation to their parent; the former centers the parent above its children, the latter arranges children in columns at a horizontal offset form their parent (see Tutorial 2). Members of both classes control the tree direction, how much space to leave between tree levels and between nodes on the same level.

Fractal Layout

The FractalLayout tree layout algorithm places child nodes symmetrically around their parent node. Nodes at the lowest level are arranged directly in a circle around their parent. At the upper level, the already arranged nodes form branches that are arranged in a circle around the new parent node. The algorithm is recursively repeated till the highest level is reached.

Treemap Layout

Tree maps represent hierarchies by nesting child nodes within their parents, where the areas of leaf nodes are proportional to their weight values. Unlike other layout algorithms, TreeMapLayout expects hierarchies to be defined via grouping or containment (see attachTo method and ContainerNode class), and will ignore any links in the diagram.

Flowchart Layout

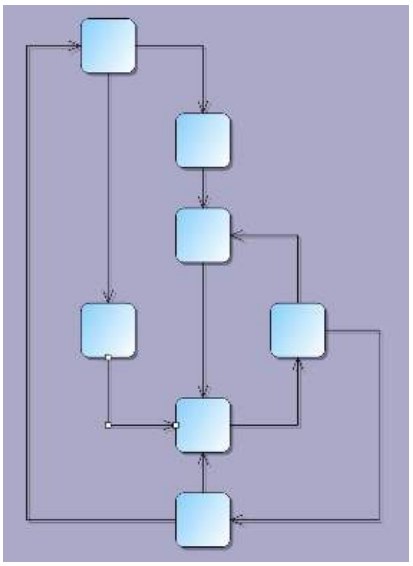
FlowchartLayout recognizes program code -like patterns in the graph, such as loops, sequences and if/switch branchings, and arranges them recursively. **FlowchartLayout** could be used to arrange other types of graphs as well, though with some restrictions. For example it treats all back links as loops in the code, and expects that they are nested - loop links starting closer to the stop node should end closer to the start node. Another similar restriction is that there shouldn't be any cross-links that connect different branches of a decision sub-graph.

Topological Layout

TopologicalLayout applies graph topological ordering to the diagram (topological ordering of a graph is one where link's origin node is placed before link's destination node). The layout algorithm arranges nodes in a row or a column, depending on the value of the direction property, in such a way that there are no backward links when the graph is acyclic. If the graph contains cycles, the algorithms selects ordering with as few backward links as possible. Links that connect non-adjacent nodes are rendered as arcs, whose amplitude is proportional to the distance between respective connected nodes. Forward links are rendered at one side of the nodes and back links are rendered at the opposite side. All that makes it easy to discern graph features such as overall flow direction, cycles and nested cycles.

Orthogonal Layout

The OrthogonalLayout class implements an orthogonal graph layout algorithm. Each link is drawn as a chain of alternating horizontal and vertical segments. Nodes are placed in a way that facilitates few links bends and crossings. This algorithm was designed for planar graphs where the nodes have at most four incident links, and produces best results with such graphs as input. It can arrange any arbitrary graph as well, by adding some dummy nodes and links to planarize the graph and reduce its vertex degree, and removing them after the layout.



Radial Tree Layout

The RadialTreeLayout class arranges tree levels in concentric circles around the root. The levelDistance property specifies the distance between adjacent circles. The direction property specifies placement of first child node. Set stretchFactor to a value larger than 1 in order to stretch level circles into ellipses, providing more space for the layout when using wider nodes.

Grid Layout

The GridLayout algorithm arranges diagram nodes in a grid, keeping connected nodes close to each other. The algorithm strives to achieve a small number of link crossings. It is based on an iterative process whose initial steps shuffle the grid nodes randomly. That can lead to very different results each time the algorithm is run.

Appearance

Customizing Node Appearance

Nodes can have different shape, background image or color. They can also have differently aligned text.

Geometric Shapes

ShapeNode exposes a shape property that specifies the geometric shape of the node. Its value can be set to either a Shape object (returned by one of the static methods of the Shapes class), or a string id of the shape, which coincides with the name of the method that gets it. Here is an example:

JavaScript

```
myNode.shape = "Arrow1";
```

TableNode and ContainerNode provide shape properties too, but their shapes can only be set to either rectangle or rounded rectangle, as specified via SimpleShape enumeration.

Bitmap Images

Each node can display a bitmap image as its content. The image can be set either via the image or imageContent properties, where the former accepts DOM Image instances, and the latter accepts base64-encoded bitmaps. By default the bitmap is drawn centered inside the node. You can change the alignment by setting imageAlign. Images can also be loaded by URL via the imageLocation property.

Fill Color

Nodes can be painted in any color that you choose. Set the brush property to specify how a node is painted.

JavaScript

```
myNode.brush = "Red";
```

Linear Gradients

Nodes can be filled using linear gradients by assigning objects in following format to brush. If the gradient should blend only two colors, they can be specified using **color1** and **color2** fields instead of **colorStops**.

JavaScript

```
var brush1 =
{
 type: "LinearGradientBrush",
 colorStops:
 [
  { position: 0, color: "red" },
  { position: 0.5, color: "green" },
  { position: 1, color: "blue" }
 ],
 angle: 30
};
myNode.brush = brush1;
```

Radial Gradients

Nodes can be filled using radial gradients by assigning objects in following format to brush. If the gradient should blend only two colors, they can be specified using **color1** and **color2** fields instead of **colorStops**.

JavaScript

```
var brush2 =
{
 type: "RadialGradientBrush",
 colorStops:
 [
  { position: 0, color: "white" },
  { position: 0.5, color: "lightGray" },
  { position: 1, color: "blue" }
 ],
 x1: 20, y1: 20, radius1: 30,
 x2: 40, y2: 20, radius2: 80
};
myNode.brush = brush2;
```

Text and Font

Nodes can display text with customizable font and horizontal/vertical alignment. Set the text property to specify the text label for a node, and textColor to change its color. textAlignment and lineAlignment properties let you set respectively horizontal and vertical alignment. The font property lets you specify a distinct font for each node:

JavaScript

```
var Font = MindFusion.Drawing.Font;

//the bool arguments specify whether the font is bold and italic
startNode.font = new Font("sans-serif", 5, true, false);
```

Apart from main text content, the NodeLabel class lets you display additional labels and pin them to node's edges or corners. Call the addLabel method to create such supplementary labels, and use **NodeLabel** (and base ItemLabel) properties and methods to customize their appearance and position.

The text of ShapeNodes can be laid out and rendered inside a polygonal outline; that depends on the value of polygonalTextLayout.

Styled Text

The enableStyledText property of shape nodes and tables enables or disables styled text rendering. Styled text can have various attributes applied to the characters it contains. The attributes are specified using HTML-like formatting tags embedded in the raw text. The following table lists the currently supported style tags:

Tag	Description
text	Indicates italic text.
text	Indicates bold text.
<u>text</u>	Indicates underlined text.
_{text}	Indicates subscript.
^{text}	Indicates superscript.
<color=#RRGGBB>text</color>	Changes the color of the text. The red, green and blue color channels are specified as two-digit hexadecimal numbers.
 	Indicates line break.
<a=url>text	Renders specified text as a hyperlink and raises hyperlinkClicked event when link has been clicked.
<size=N>text</size>	Changes the font size.

Node Effects

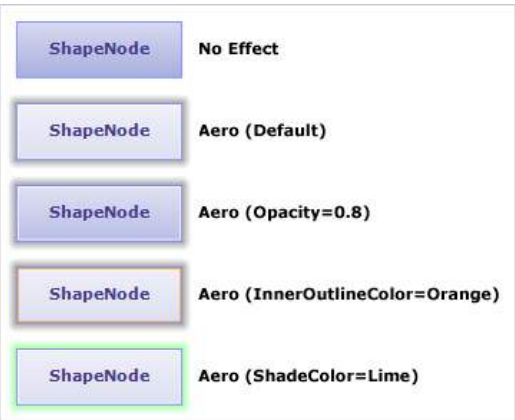
To apply an effect to the nodes in a diagram, create an instance of one of the effect classes and add this instance to the Diagram.nodeEffects array. To apply the effect to individual nodes, add the instance to a node's own effects array. To ignore the global diagram effects for a particular node, add a **null** reference at position 0 in the node's **effects**. Effects can also be applied through styles by adding them to the Style.nodeEffects array of the diagram or a node.

It is possible to combine different effects by adding instances of their appropriate types to the effect collections. If is also admissible to apply more than one effect of the same type.

The currently supplied effect types are Aero and Glass. They are examined in more details in the sections below.

Aero Effect

The Aero effect is represented by the AeroEffect class. The effect is designed to roughly resemble the Aero theme of Windows. More specifically, the effect adds semi transparency to the nodes, a smooth shade at the outside and a thin inset stroke. The semi transparency is specified through the opacity property. Opacity varies from 0 to 1 inclusive, where 0 indicates full transparency and 1 indicates the original node color. The semi transparency is applied only on the node's brush. The colors of the shade and the inner stroke are specified through the shadeColor and innerOutlineColor properties respectively. The following image illustrates how these properties affect the output:



The following example applies the Aero effect on the nodes of an existing diagram:

JavaScript

```
var AeroEffect = MindFusion.Diagramming.AeroEffect;

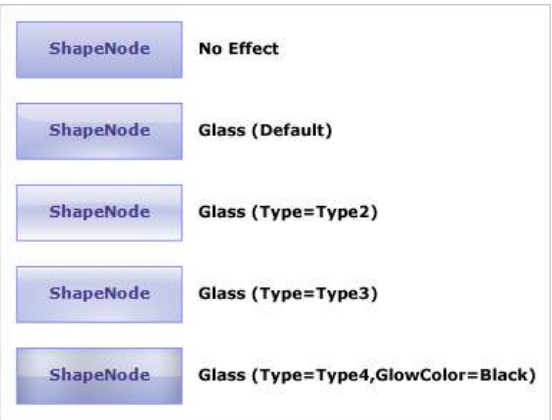
var effect = new AeroEffect();
effect.opacity = 0.5;
effect.innerOutlineColor = "rgba(255,255,255,0.5)";
diagram.nodeEffects.push(effect);
```

Tips

- The Aero effect works very well in scenarios with overlapping nodes or in cases where the background is more complex than solid color, due to the effect of semi transparency.
- Aero combines well with the Glass effect. There are some examples at the end of this topic.
- The shadeColor is usually black or another dark color. However, some nice glow effects can be achieved by setting bright **shadeColor** on a diagram with dark background.
- The Aero effect can impact performance if there are many nodes visible in the diagram.

Glass Effect

The Glass effect is represented by the GlassEffect class. The Glass effect adds visuals on top of the nodes in order to give the impression of a reflective surface. There are several variations of this effect, which can be selected through the type property. The color of the reflection and the ambient color can be specified through the reflectionColor and glowColor properties respectively. When usePenAsGlow is set to **true**, the color of the node's pen is used as the ambient color. The following image illustrates the differences between the distinct effect types:



The following example applies a Glass effect with the default settings to the nodes of a diagram:

JavaScript

```
var GlassEffect = MindFusion.Diagramming.GlassEffect;
diagram.nodeEffects.push(new GlassEffect());
```

Tips

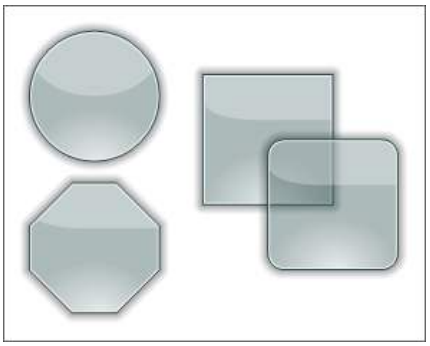
- The glowColor is usually white. However, for Type4 glass effects it is often more suitable to use black glow.
- The Glass effect combines well with the Aero effect. There are some examples at the end of this topic.
- Several glass effects of different types can be applied simultaneously for emphasized effect.

Examples

The values of the effect properties are usually selected in accordance with the color scheme of the target diagram and the desired end result. In addition different types of effects can be applied simultaneously to achieve more interesting results. The examples below demonstrate possible effect combinations in various color contexts.

Aero & Glass (Dark on Light)

In this example the Aero and Glass effects are combined. The Aero effect is useful in scenarios when dark nodes are placed on light background or vice versa, in order to emphasize the contrast.



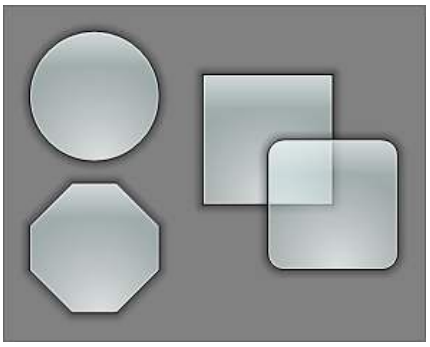
Here is the code that applies the above effects:

JavaScript

```
diagram.nodeEffects.push(new GlassEffect());
diagram.nodeEffects.push(new AeroEffect());
```

Aero & Glass (Light on Dark)

This example is similar to the first one, only this time the nodes are painted in light colors and the background is dark.



Here is the code that applies the above effect:

JavaScript

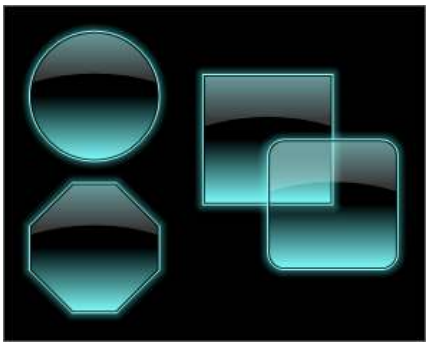
```
var AeroEffect = MindFusion.Diagramming.AeroEffect;
var GlassEffect = MindFusion.Diagramming.GlassEffect;
var GlassEffectType = MindFusion.Diagramming.GlassEffectType;

var glassEffect = new GlassEffect();
glassEffect.type = GlassEffectType.Type3;

diagram.nodeEffects.push(glassEffect);
diagram.nodeEffects.push(new AeroEffect());
```

Glow Effect

This example uses bright colors on black background to create a glow effect. The example utilizes an Aero effect with shadeColor set to the color of the nodes' pens.



Here is the code that applies the above effects:

JavaScript

```
var AeroEffect = MindFusion.Diagramming.AeroEffect;
var GlassEffect = MindFusion.Diagramming.GlassEffect;
var GlassEffectType = MindFusion.Diagramming.GlassEffectType;

var glassEffect = new GlassEffect();
glassEffect.type = GlassEffectType.Type2;
glassEffect.usePenAsGlow = true;
diagram.nodeEffects.push(glassEffect);

var aeroEffect = new AeroEffect();
aeroEffect.opacity = 0;
aeroEffect.innerOutlineColor = "black";
aeroEffect.shadeColor = "rgb(128, 255, 255)";
diagram.nodeEffects.push(aeroEffect);
```

Customizing Link Appearance

DiagramLinks-s consist of series of straight or Bezier segments and arrowhead shapes. The link and arrowhead shapes, and also color attributes and text can be customized as detailed below:

Link Geometry

Links can be straight or curved and can contain one or more segments. Set DiagramLink.shape to change the shape. It takes as an argument one of the LinkShape enumeration values:

JavaScript

```
myLink.shape = LinkShape.Cascading;
```

The link can be customized further by changing its controlPoints array, whose elements specify the position of the link segments. The number of control points per segment depends on the link's **shape**: **Bezier** links have four control points per segment. Each straight segment has two control points. Adjacent segments share a control point.

The following code sets three control points of a polyline link, which results in two segments:

JavaScript

```
var Point = MindFusion.Drawing.Point;
myLink.controlPoints = [new Point(10, 10), new Point(30, 30), new Point(30, 50)];
```

Note

Always call DiagramLink.updateFromPoints() after you change coordinates of individual control points in the array.

Rounded Links

Links comprising straight segments (ones whose shape is set to Polyline or **Cascading**) can draw arcs as joints between their adjacent segments. To enable that feature, set roundedLinks to **true**. Use roundedLinksRadius to specify the radius of the joining arcs.

Link Crossings

Intersection points where links cross their paths can be rendered as arcs or cuts, as specified via linkCrossings. It depends on the Z-order, which links jump over the others, or which links appear cut by the others. If **linkCrossings** is set to Arcs, links with higher ZIndex jump over those with lower ZIndex. If set to **Cut**, links at lower Z position are cut by links at higher Z. Radius of arcs and breaks can be set through the crossingRadius property.

Customizing Arrowheads

Each link can display arrowhead shapes at its ends. The shape at the start is specified via baseShape property, the shape at the end is specified via headShape. Arrowheads can also be displayed at middles of segments, helping users follow the path of long links. They are defined by the intermediateShape property. Arrowhead values can be set to either a Shape instance or a string that identifies it:

JavaScript

```
myLink.headShape = "BowArrow";
```

Shape objects designed specifically as arrowheads are exposed as static members of the ArrowHeads class.

Specifying Colors

Link segments are rendered according to stroke, strokeThickness and strokeDashStyle attributes. The arrowheads at the head and base are filled using headBrush and baseBrush properties:

JavaScript

```
// paints the link head with red brush
myLink.headBrush = "Red";
```

Setting Text

Set the text property to specify text label of a link, and font to customize its font. Here is an example how to do this:

JavaScript

```
myLink.text = "myLabel";

var Font = MindFusion.Drawing.Font;

// the bool arguments specify whether the font is bold and italic
myLink.font = new Font("sans-serif", 5, true, false);
```

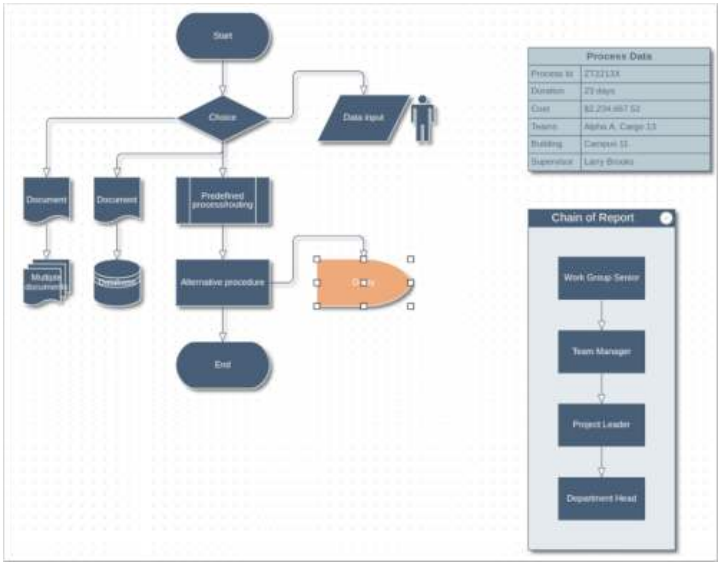
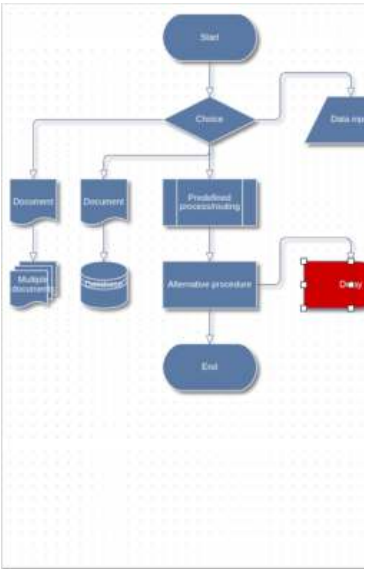
The font size is specified in the diagram's current measure unit, which is millimeter by default.

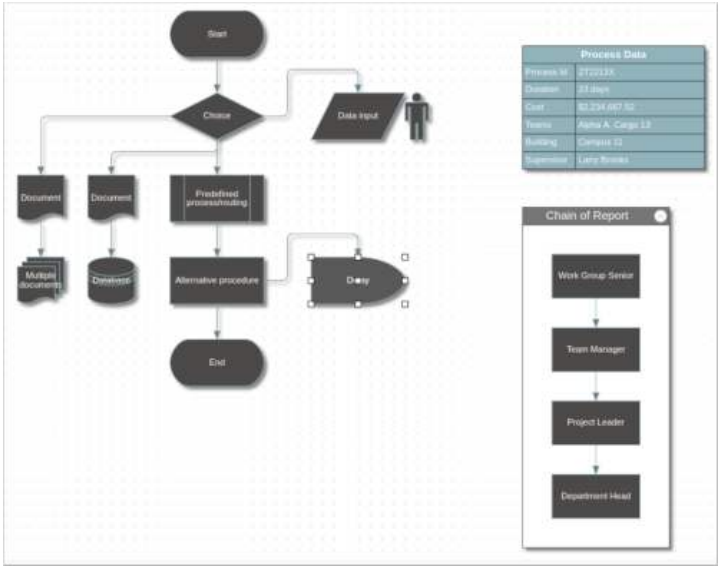
Apart from main text content, the LinkLabel class lets you display additional labels and pin them to link's points or segments. Call the addLabel method to create such supplementary labels, and use **LinkLabel** (and base ItemLabel) properties and methods to customize their appearance and position.

Styles and Themes

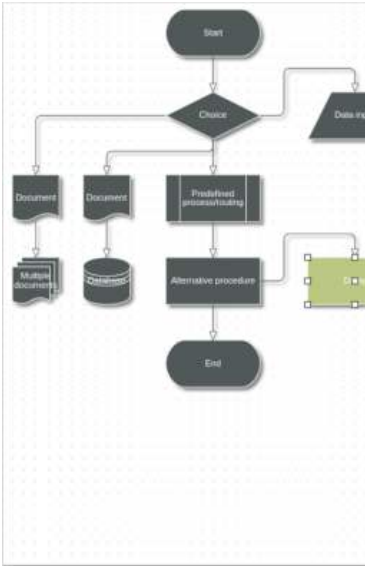
Appearance of items can be customized using Style objects assigned to respective style property of DiagramItem. The attributes from a style are applied only when item's own corresponding properties do not contain values.

Style objects can also be collected into a Theme and applied as a set to the whole diagram by assigning it to the theme property. The library includes nine stock themes listed below, accessible via respective static methods of the **Theme** class.

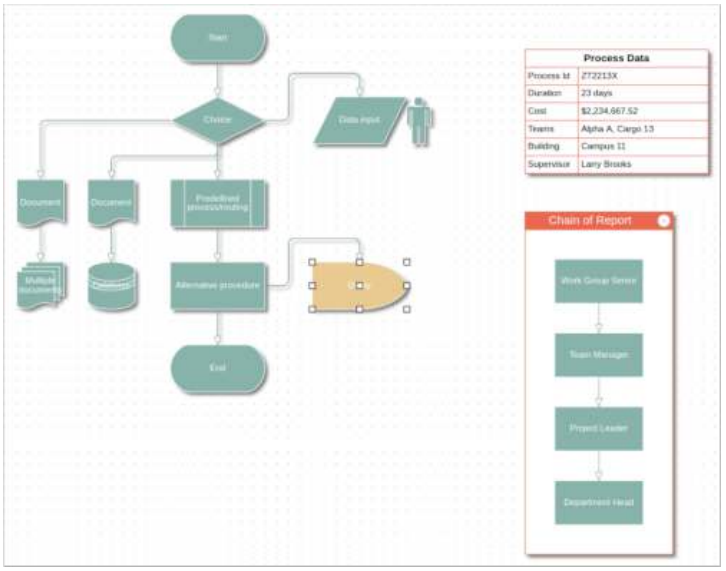
Blue	Business
	
Gray	Green

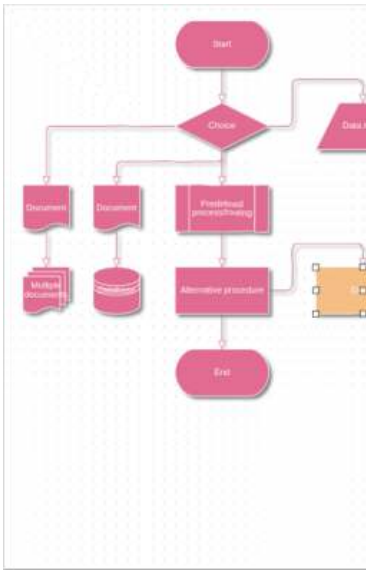


[Pastel](#)



[Peach](#)





Creating and applying a custom theme looks like this:

JavaScript

```
var theme = new Theme();

var shapeNodeStyle = new Style();
shapeNodeStyle.brush = { type: 'LinearGradientBrush', color1: '#e0e9e9', color2: '#9caac6',
angle: 90 };
shapeNodeStyle.stroke = "#7F7F7F";
shapeNodeStyle.textColor = "#585A5C";
shapeNodeStyle.fontName = "Verdana";
shapeNodeStyle.fontSize = 3;
theme.styles.set("std:ShapeNode", shapeNodeStyle);

var linkStyle = new Style();
linkStyle.stroke = "#7F7F7F";
linkStyle.textColor = "#585A5C";
linkStyle.fontName = "Verdana";
linkStyle.fontSize = 3;
theme.styles.set("std:DiagramLink", linkStyle);

var tableStyle = new Style();
tableStyle.brush = { type: 'LinearGradientBrush', color1: '#FCFCFC', color2: '#9DABB4', angle:
90 };
```

```
tableStyle.stroke = "#7F7F7F";
tableStyle.textColor = "#585A5C";
tableStyle.fontName = "Verdana";
tableStyle.fontSize = 3;
theme.styles.set("std:TableNode", tableStyle);

diagram.theme = theme;
```

CSS Styling

The styling system also lets you define item appearance through external CSS files. You can style different item types using custom element selectors such as **mfd-shapenode**, **mfd-diagramlink**, or **mfd-tablenode**. The system maps CSS Custom Properties to the internal Style properties. For example, use **--brush** for backgrounds, **--stroke** for outlines, and **--font-size** for text. The new cssClass property on DiagramItem allows for more granular styling using standard class selectors in your CSS. To ensure high performance, computed styles are cached globally, and the system automatically resolves style inheritance across class hierarchies (e.g., a ShapeNode will resolve styles from **mfd-shapenode**, **mfd-diagramnode**, and **mfd-diagramitem** selectors in that order).

CSS

```
/* Style all shape nodes */
mfd-shapenode {
    --brush: #f9f9f9;
    --stroke: #333;
    --stroke-thickness: 1px;
    --font-name: 'Verdana';
}

/* Specific styling via the new cssClass property */
mfd-shapenode.urgent {
    --brush: #ffcccc;
    --stroke: #ff0000;
    --stroke-thickness: 2px;
}

/* Style table captions using custom properties */
mfd-tablenode {
    --caption-brush: #e0e0e0;
    --text-color: #000066;
}
```

Here is the complete list of **mfd-** prefixed CSS element selectors available for styling items:

Base Elements

- **mfd-diagramitem**: The base selector for all diagram items. Styles applied here will be inherited by all other items unless overridden.
- **mfd-diagramnode**: The base selector for all node types. Inherits from **mfd-diagramitem**.

Specific Node Types

(all inherit from **mfd-diagramnode**)

- **mfd-shapenode**: Styles ShapeNode instances.
- **mfd-svgnode**: Styles SvgNode instances.
- **mfd-compositenode**: Styles CompositeNode instances.
- **mfd-controlnode**: Styles ControlNode instances.
- **mfd-videonode**: Styles VideoNode instances.
- **mfd-freeformnode**: Styles FreeFormNode instances.

Container & Complex Nodes

- **mfd-containernode**: Styles ContainerNode instances (inherits from **mfd-captionednode**).
- **mfd-captionednode**: The base selector for nodes with captions like Tables and TreeViews. Inherits from **mfd-diagramnode**.
- **mfd-tablenode**: Styles TableNode instances (inherits from **mfd-captionednode**).
- **mfd-treeviewnode**: Styles TreeViewNode instances (inherits from **mfd-captionednode**).

Links

- **mfd-diagramlink**: Styles DiagramLink instances. Inherits from **mfd-diagramitem**.

Sample Inheritance Chains

- **mfd-shapenode** ? **mfd-diagramnode** ? **mfd-diagramitem**
- **mfd-tablenode** ? **mfd-captionednode** ? **mfd-diagramnode** ? **mfd-diagramitem**
- **mfd-diagramlink** ? **mfd-diagramitem**

Properties

Here is the list of custom CSS properties (variables) available for diagram items. These properties should be defined within the **mfd-** element selectors, and possibly assigned to specific items via the cssClass property.

Primary Appearance

- **--brush**: Specifies the background / fill color. Supports CSS color strings (e.g., `#FF0000`, `rgba(...)`, `lightblue`).
- **--stroke**: Specifies the outline / border color.
- **--stroke-thickness**: Specifies the width of the outline. Supports units like px, mm, pt, or in (values are converted to diagram's measureUnit).
- **--shadow-color**: Specifies the color of the item's shadow.

Typography & Text

- **--text-color**: Specifies the color of the text label.
- **--font-name**: Specifies the font family (e.g., 'Verdana', 'Arial').
- **--font-size**: Specifies the size of the font. Supports CSS units (values are converted to diagram's measureUnit).
- **--font-weight**: Use 'bold' to apply bold styling.
- **--font-style**: Use 'italic' to apply italic styling.
- **--text-decoration**: Use 'underline' to apply underline styling.

Selection States

- **--selected-brush**: Specifies the fill brush used when the item is selected.
- **--selected-stroke**: Specifies the outline color used when the item is selected.

Specialized Properties

- **--caption-brush**: Specifies the background color of captions (used by **mfd-tablenode**, **mfd-containernode**, and **mfd-treeviewnode**).
- **--vertical-border-stroke**: Specifies the color of vertical grid lines/borders (primarily for TableNode).
- **--horizontal-border-stroke**: Specifies the color of horizontal grid lines/borders (primarily for **TableNode**).

Define Shape Geometries

Diagrams can display geometric shapes by setting the shape property of ShapeNode instances.

Shape Templates

The Shape class provides the means for defining complex node shapes, made of straight lines, arcs and Bézier curves. A shape template must always contain an outline to be used for hit testing, clipping and finding intersections with other items. Optionally shapes can contain decoration elements, which are visual elements that do not take part in hit testing and clipping. JsDiagram provides a set of over a 100 predefined shape templates, which can be accessed by the Shapes collection or fromId method of the **Shape** class. A shape is assigned to a node by means of the shape property.

JavaScript

```
shapeNode.shape = MindFusion.Diagramming.Shapes.Actor();
shapeNode.shape = MindFusion.Diagramming.Shape.fromId("Actor");
```

Custom Shapes

You can define your own shapes by using the static createShape method of the Diagramming.Controls.ShapeBuilder class. The shape geometry is specified as an array of ShapeElements. The ShapeElement class constructor expects a command and an array of command parameters, and optionally styling. The list of supported commands matches the HTML Canvas drawing API (e.g moveTo, lineTo, bezierCurveTo, arc, roundRect). Coordinates are defined as percentage of a node's bounding rectangle.

JavaScript

```
var outline = [];
outline.push(new ShapeElement("moveTo", [0, 33]));
outline.push(new ShapeElement("lineTo", [33, 0]));
outline.push(new ShapeElement("lineTo", [66, 0]));
outline.push(new ShapeElement("lineTo", [100, 33]));
outline.push(new ShapeElement("lineTo", [100, 100]));
outline.push(new ShapeElement("bezierCurveTo", [66, 112, 33, 87, 0, 100]));
outline.push(new ShapeElement("lineTo", [0, 33]));
var shape = MindFusion.Diagramming.Controls.ShapeBuilder.createShape("MyShape", outline);
shapeNode.shape = shape;
```

The Shape definition can also contain decorations and shape decorations. A decoration is a geometry that consists of one or more lines or curves. A shape decoration is a closed path of lines and/or curves, that can also have a fill. A shape can have one decoration and a number of shape decorations, that are specified through the decorationElements and shapeDecorationElements parameters as follows:

JavaScript

```
var decoration = [];
decoration.push(new ShapeElement("moveTo", [0, 33]));
decoration.push(new ShapeElement("lineTo", [33, 0], "red"));
decoration.push(new ShapeElement("lineTo", [66, 33], "blue", 4));

var shapeDecorationElements = [];
shapeDecorationElements.push(new ShapeElement("roundRect", [50, 50, 80, 80, 2]));
var shapeDecoration = new ShapeDecoration(shapeDecorationsElements, "red");

var shape = MindFusion.Diagramming.Controls.ShapeBuilder.createShape(
    "MyShape", outline, decoration, [shapeDecoration]);
```

If you don't need the per-element styling options, another way to create a shape is by creating an instance of the **Shape** class providing the outline definition using SVG Path syntax.

JavaScript

```
var shape = new Shape({
    id: "MyShape"
    outline: "M100,50 A50,50,0,1,1,0,50 A50,50,0,1,1,100,50",
    decoration: "M98,50 A48,48,0,1,1,2,50 A48,48,0,1,1,98,50"});
```

A string identifier must be specified for every shape so they will be accessible via the Shape.fromId method.

Shape Libraries

You can use a Shape Library xml file to store your collection of custom shapes. Set the shapeLibraryLocation property to specify the path to the file.

JavaScript

```
diagram.shapeLibraryLocation = "MyShapes.sl";
```

The shapes in the Shape Library file will be automatically added to the shared list of shapes kept in the Shape class, which lets you access them by calling the fromId method.

Shape Designer Tool

The ShapeDesigner tool that comes with JsDiagram lets you create and edit custom shapes interactively and save them to Shape Library files. When the tool starts, a new shape library is created by default with a single basic rectangular shape in it. To modify the shape, drag and drop shape elements from the palette to the right onto the shape canvas. The following drop operations are supported:

- dropping an element over existing element of a shape outline replaces the element (the element to be replaced will be marked in red)
- dropping an element over the joint of two outline elements inserts the element between them (the joint will be marked in blue)
- dropping an element on an empty area of the canvas adds a decoration to the Shape
- dropping a decoration over the shape outline replaces all elements of the outline with ones from the decoration
- dropping a decoration on an empty area of the canvas adds a filled ShapeDecoration to the **Shape**
- dropping a decoration over a **ShapeDecoration** replaces its elements

Elements/decorations can be further customized by dragging their control points.
You can delete, copy and paste dropped elements/decorations, as well as undo and redo drop actions via the options in the Edit menu.
Styling can be specified to elements/decorations by using the controls in the property panel, that is shown when an element/decoration is selected.
The Preview panel lets you see how your shape will look and behave in a real DiagramView control.
New shapes can be added from the Shape/New menu option.
The finished library can be exported to an .sl file from the File/Save Library menu option.

Animations

Item Animations

The animate method of DiagramItem can run following animations:

- FillColorAnimation animates the fill color of nodes or link arrowheads.
- StrokeColorAnimation animates the color of node borders or link segments.
- OpacityAnimation animates items' opacity for fade-in and fade-out effects.

The following animations can be applied to DiagramNodes:

- BounceAnimation animates vertical position of nodes.
- ShakeAnimation animates horizontal position of nodes.
- PulseAnimation animates node's local scale factor.
- PathAnimation moves nodes along specified path.

Item's animated values are considered transient and do not change actual property values. If you want to assign last value from animation to actual property such as bounds or opacity, set the *keepLastValue* flag in animation parameters:

JavaScript

```
node.animate(
    new PathAnimation(
        [
            new Point(30, 30),
            new Point(50, 30),
            new Point(75, 66),
            new Point(125, 88)
        ],
        {
            keepLastValue: true
        }));
```

Other animation parameters include duration, repeat, repeatCount, from / to values, animation curve and easing options:

JavaScript

```
node.animate(
    new OpacityAnimation(
        {
            fromValue: 1,
            toValue: 0.5,
            duration: 250,
            repeat: true,
            repeatCount: 4,
            reverse: true,
            animationType: Animations.AnimationType.BackBow
        }));
```

Item and view animations that affect different properties can run simultaneously.

View Animations

The animate method of DiagramView can run following animations:

- ScrollAnimation animates the scroll position of a view.
- ZoomAnimation animates the zoom factor of a view.

JavaScript

```
diagramView.animate(
        new ScrollAnimation(
        {
            fromValue: new Point(0, 0),
            toValue: new Point(100, 60),
            animationType: Animations.AnimationType.Power,
            easingType: Animations.EasingType.EaseInOut
        }));
```

View animations are not transient and always set the scroll* or zoomFactor properties.

Looping Animations

To create looping animations, use the repeat and reverse properties. To stop an animation, call its stop method.

JavaScript

```
// Use this property to repeat the animation until stopped.
animation.repeat = true;
// Use this property to reverse the to and from values.
animation.reverse = true;

// Use the stop method to complete the animation.
animation.stop();
```

Custom Animations

To animate arbitrary attributes of any type, an Animation instance can receive an **updateCallback** function as a parameter to the constructor. This callback function can be used to set item coordinates or property values of the object being animated. The callback signature should contain two arguments - the animation object instance and a value representing the animated property delta. This callback is called at every animation timer "tick". The following example creates a pulse-like effect, using an update callback:

JavaScript

```
var Animation = MindFusion.Animations.Animation;
var Utils = MindFusion.Diagramming.Utils;

var options =
    {
        duration: 550,
        fromValue: -0.15,
        toValue: 0.15,
        repeat: true,
        reverse: true
    };

/**
* @param [Animation] animation The animation object instance.
* @param [Number] animationDelta A number representing the percentage of displacement
* of the animated property's value in the [fromValue, toValue] range.
*/
function pulse(animation, animationDelta)
{
    var bounds = animation.item.bounds.clone();
    var to = animation.toValue;
    bounds = Utils.inflate(bounds, to * animationDelta, to * animationDelta);
    animation.item.setBounds(bounds, true);
}

var animation = new Animation(node, options, pulse);
animation.start();
```

Alternatively, a class derived from **Animation** could override the virtual **onUpdate** method.

Default Animation and Easing Functions

There are seven default animation type functions that can be used to animate items, designated by respective elements of AnimationType enumeration:

- Linear: represents an animation that accelerates / decelerates using a linear function.
- Power: represents an animation that accelerates / decelerates through exponentiation formula - x^n where n can be set through the animation's param property. The default exponent value is 2.
- Exponential: represents an animation that accelerates / decelerates using an exponential function. The exponent can be set through the animation's **param** property. The default exponent value is 10.
- Circular: represents an animation that accelerates / decelerates using a circular function.
- BackBow: represents an effect that simulates a retraction in the motion of the animation. The amplitude of the retraction can be controlled through the animation's **param** property. The default param value is 1.5.
- Bounce: represents a bouncing effect animation.
- Elastic: represents an oscillating effect animation. The oscillation frequency can be controlled through the animation's **param** property. The default param value is 0.5.

The default animations can be additionally customized by an easing function, specified via EasingType enumeration:

- EaseIn: follows the animation function associated with the Animation.
- EaseOut: inverses the animation function associated with the **Animation**.
- EaseInOut: uses **EaseIn** for the first half of the **Animation**'s duration and **EaseOut** for the second half.
- EaseOutIn: uses **EaseOut** for the first half of the **Animation**'s duration and **EaseIn** for the second half.

Custom Animation Functions

In addition to the provided default animation functions, it is also possible to define custom animation effects. Set the animationType property to Custom and pass a custom animation function callback to the Animation constructor. The callback signature should contain two arguments - the time progressed as a percentage from the start of the animation and the user-set parameter, used in the animation function. The following example uses a custom function where x is a value between 0 and 1 and y is a parameter, passed as an argument to the **Animation** constructor.

JavaScript

```
var Animation = MindFusion.Animations.Animation;
var AnimationType = MindFusion.Animations.AnimationType;

var options =
    {
        duration: 500,
        fromValue: 5,
        toValue: 100,
        param: 5,
        animationType: AnimationType.Custom
    };

/**
* @param [Number] progress The time progress in the animation loop. A value between 0 and 1.
* @param [Object] param The user-set value, used to parameterize the animation function.
*/
function custom(progress, param)
{
    return Math.pow(Math.E, 1 - (1 / Math.pow(progress, param)));
}

var animation = new Animation(node, options, null, custom);
animation.start();
```

Serializing Diagrams

A diagram can be saved as JSON string by calling the toJson method. The produced JSON string could be saved in browser's local storage or sent to a web server to store in a database or a file. Later the diagram's contents can be restored by calling fromJson. The JSON strings are also compatible with MindFusion's ASP.NET MVC API and can be loaded in .NET Diagram object to inspect the diagram model on server side.

It is also possible to save a diagram as an XML document. To do that, call the saveToXml method of the Diagram class. To deserialize a diagram back from its XML representation, call the loadFromXml method. The same XML format is supported by all MindFusion diagram components for mobile, desktop and web platforms.

The saveToString and loadFromString methods let you serialize or deserialize to/from either JSON or XML strings produced by the other methods described above.

Miscellaneous

Associate Custom Data with Items

You can associate custom values from any data type with diagram elements by setting their tag and id properties. Their values will be automatically serialized to JSON format using browser's JSON serializer when saving diagram JSON files. If saving XML files, you must handle the serializeTag and deserializeTag events to save non-string values in XML format. Apart from **tag** and **id**, JavaScript lets you create new object fields dynamically - these however will not be saved automatically in diagram files.

Fluent API

Builder objects with property setter methods and shortcut methods for fonts and brushes add support for fluent programming style. Static **With** and instance **init** methods in DiagramItem, Style and Layout -derived classes return a builder instance that can be used to set up respective new or existing objects:

JavaScript

```
diagram.addItem(
    ShapeNode.With()
        .brush("lightGray")
        .font("Arial", 4)
        .enableStyledText(true)
        .text("Task <i>1</i>")
        .tooltip("This is the task")
        .create());

diagram.getFactory()
    .createShapeNode(20, 20, 20, 20).init()
        .brush("lightGray", "white", 20)
        .font("Arial", 4)
        .enableStyledText(true)
        .text("Task <i>2</i>")
        .tooltip("This is the second task");

diagram.arrange(
    TreeLayout.With()
        .levelDistance(20)
        .nodeDistance(20)
        .linkType(TreeLayoutLinkType.Cascading3)
        .create());
```

Tutorials

Tutorial 1: Loading Graph Data

This tutorial shows how to load graph data from a Json string and create diagram objects corresponding to the graph nodes and edges. The code is also available in Samples folder of JsDiagram distribution.

1. Create a text file called Tutorial1.txt and add the following Json string to it:

XML

```
{"nodes":[
    {"id":0,"name":"start"},
    {"id":1,"name":"activity 1"},
    {"id":2,"name":"task 1"},
    {"id":3,"name":"task 2"},
    {"id":4,"name":"activity 2"},
    {"id":5,"name":"task 3"},
    {"id":6,"name":"task 4"},
    {"id":7,"name":"activity 3"},
    {"id":8,"name":"task 5"},
    {"id":9,"name":"task 6"},
    {"id":10,"name":"end"}
],
"links":[
    {"origin":0,"target":1},
    {"origin":1,"target":2},
    {"origin":1,"target":3},
    {"origin":2,"target":4},
    {"origin":3,"target":4},
    {"origin":4,"target":5},
    {"origin":4,"target":6},
    {"origin":5,"target":10},
```

```
    {"origin":6,"target":10},
    {"origin":0,"target":7},
    {"origin":7,"target":8},
    {"origin":8,"target":9},
    {"origin":2,"target":8},
    {"origin":9,"target":10}
]}
```

2. Create empty Tutorial1.html and Tutorial1.js files, and add DIV and CANVAS elements to the HTML page.

HTML

```
<!-- The DiagramView component is bound to the canvas element below -->
<div style="width: 100%; height: 100%; overflow: auto;">
    <canvas id="diagram" width="2100" height="2100">
        This page requires a browser that supports HTML 5 Canvas element.
    </canvas>
</div>
```

3. Add script references to the MindFusion UMD scripts and Tutorial1 JavaScript files:

HTML

```
<script src="umd/collections.js" type="text/javascript"></script>
<script src="umd/drawing.js" type="text/javascript"></script>
<script src="umd/controls.js" type="text/javascript"></script>

<script src="umd/animations.js" type="text/javascript"></script>
<script src="umd/graphs.js" type="text/javascript"></script>
<script src="umd/diagramming.js" type="text/javascript"></script>

<script src="Tutorial1.js" type="text/javascript"></script>
```

4. In Tutorial1.js, assign shorter names to the DiagramView, Diagram, ShapeNode, DiagramLink, LayeredLayout and Rect classes (if using diagram's ES6 or CommonJS modules, these would be **import** or **require** statements for respective classes and modules).

JavaScript

```
var DiagramView = MindFusion.Diagramming.DiagramView;
var Diagram = MindFusion.Diagramming.Diagram;
var DiagramLink = MindFusion.Diagramming.DiagramLink;
var ShapeNode = MindFusion.Diagramming.ShapeNode;

var Rect = MindFusion.Drawing.Rect;
var LayeredLayout = MindFusion.Graphs.LayeredLayout;
```

5. Add a **DOMContentLoaded** handler that creates a **DiagramView** instance wrapping the HTML Canvas element:

JavaScript

```
var diagram;

document.addEventListener("DOMContentLoaded", function ()
{
    // create a DiagramView component that wraps the "diagram" canvas
    var diagramView = DiagramView.create(document.getElementById("diagram"));
    diagram = diagramView.diagram;
}
```

6. Add the following code to the **DOMContentLoaded** handler to create a network request that loads the Json file from the server:

JavaScript

```
// pretend we are calling a web service
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function ()
{
```

```
};
xhttp.open("GET", "Tutorial1.txt", true);
xhttp.send();
```

7. Implement the **onreadystatechange** handler to parse the json file:

JavaScript

```
if (this.readyState == 4 && this.status == 200)
{
    var graph = JSON.parse(xhttp.responseText);
    buildDiagram(graph);
}
```

8. Add the buildDiagram function that creates diagram nodes and links for each object from the parsed Json graph:

JavaScript

```
function buildDiagram(graph)
{
    var nodeMap = [];

    // load node data
    var nodes = graph.nodes;
    for (var node of nodes)
    {
        var diagramNode = diagram.factory.createShapeNode(0, 0, 18, 8);
        nodeMap[node.id] = diagramNode;
        diagramNode.text = node.name;
        diagramNode.brush = "#e0e9e9";
    }

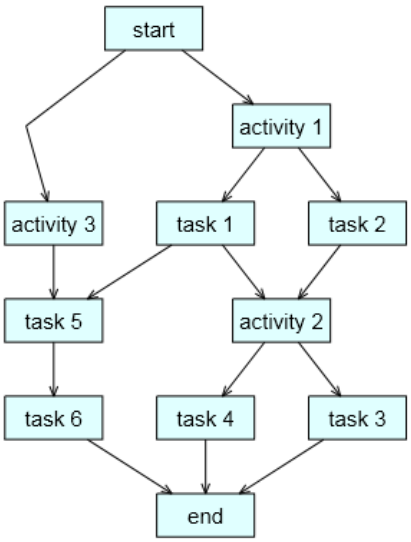
    // load link data
    var links = graph.links;
    for (var link of links)
    {
        diagram.factory.createDiagramLink(
            nodeMap[link.origin],
            nodeMap[link.target]);
    }
}
```

9. Finally, apply **LayeredLayout** to arrange the diagram automatically:

JavaScript

```
// arrange the graph
var layout = new LayeredLayout();
layout.nodeDistance = 10;
layout.layerDistance = 10;
diagram.arrange(layout);
```

10. Publish MindFusion.*.js and tutorial files using web server of your choice. Now you should see this if you open Tutorial1.html in a web browser:



Tutorial 2: Loading Hierarchical Data

This tutorial shows how to load hierarchical data recursively from a Json string and create diagram items corresponding to the hierarchy. The code is also available in Samples folder of JsDiagram distribution.

1. Create a text file called Tutorial2.txt and add the following Json string to it:

XML

```
{ "project" :
 {"name": "project", "activities":[
  {"name":"Activity 1", "activities":[
    {"name":"sub-activity 1", "activities":[
      {"name":"sub-activity 1-1"},
      {"name":"sub-activity 1-2"}]},
    {"name":"sub-activity 2", "activities":[
      {"name":"sub-activity 2-1"},
      {"name":"sub-activity 2-2"},
      {"name":"sub-activity 2-3"}]},
    {"name":"sub-activity 3", "activities":[
      {"name":"sub-activity 3-1"},
      {"name":"sub-activity 3-2"}]}
  ]},
  {"name":"Activity 2", "activities":[
    {"name":"sub-activity 1", "activities":[
      {"name":"sub-activity 1-1"},
      {"name":"sub-activity 1-2"},
      {"name":"sub-activity 1-3"}]},
    {"name":"sub-activity 2", "activities":[
      {"name":"sub-activity 2-1"},
      {"name":"sub-activity 2-2"}]}
  ]},
  {"name":"Activity 3", "activities":[
    {"name":"sub-activity 1", "activities":[
      {"name":"sub-activity 1-1"},
      {"name":"sub-activity 1-2"},
      {"name":"sub-activity 1-3"},
      {"name":"sub-activity 1-4"}]},
    {"name":"sub-activity 2"},
    {"name":"sub-activity 3", "activities":[
      {"name":"sub-activity 3-1"},
      {"name":"sub-activity 3-2"},
      {"name":"sub-activity 3-3"}]}
  ]}
 ]}
}
```

2. Create empty Tutorial2.html and Tutorial2.js files, and add DIV and CANVAS elements to the HTML page.

HTML

```
<!-- The DiagramView component is bound to the canvas element below -->
<div style="width: 100%; height: 100%; overflow: auto;">
    <canvas id="diagram" width="2100" height="2100">
        This page requires a browser that supports HTML 5 Canvas element.
    </canvas>
</div>
```

3. Add script references to the MindFusion UMD scripts and Tutorial2 JavaScript files:

HTML

```
<script src="umd/collections.js" type="text/javascript"></script>
<script src="umd/drawing.js" type="text/javascript"></script>
<script src="umd/controls.js" type="text/javascript"></script>

<script src="umd/animations.js" type="text/javascript"></script>
<script src="umd/graphs.js" type="text/javascript"></script>
<script src="umd/diagramming.js" type="text/javascript"></script>

<script src="Tutorial2.js" type="text/javascript"></script>
```

4. In Tutorial2.js, assign shorter names to the DiagramView, Diagram, ShapeNode, DiagramLink, BorderedTreeLayout and Rect classes (if using diagram's ES6 or CommonJS modules, these would be **import** or **require** statements for respective classes and modules).

JavaScript

```
var DiagramView = MindFusion.Diagramming.DiagramView;
var Diagram = MindFusion.Diagramming.Diagram;
var DiagramLink = MindFusion.Diagramming.DiagramLink;
var ShapeNode = MindFusion.Diagramming.ShapeNode;

var Rect = MindFusion.Drawing.Rect;
var BorderedTreeLayout = MindFusion.Graphs.BorderedTreeLayout;
```

5. Add a **DOMContentLoaded** handler that creates a **DiagramView** instance wrapping the HTML Canvas element:

JavaScript

```
var diagram = null;

document.addEventListener("DOMContentLoaded", function ()
{
    // create a DiagramView component that wraps the "diagram" canvas
    var diagramView = DiagramView.create(document.getElementById("diagram"));
    diagram = diagramView.diagram;
}
```

6. Add the following members to the script to hold the default node size and color.

JavaScript

```
var bounds = new Rect(0, 0, 30, 6);
var brush = "LightBlue";
```

7. Add the following code to the load handler to create a network request that loads the Json file from the server:

JavaScript

```
// pretend we are calling a web service
var xhttp = new XMLHttpRequest();
xhttp.onreadystatechange = function ()
{
};
```

```
xhttp.open("GET", "Tutorial2.txt", true);
xhttp.send();
```

8. Implement the **onreadystatechange** handler to parse the json file:

JavaScript

```
if (this.readyState == 4 && this.status == 200)
{
    var graph = JSON.parse(xhttp.responseText);
    buildDiagram(graph);
}
```

9. Add the buildDiagram function that creates the tree root node, and call the addActivities function, which builds the tree recursively:

JavaScript

```
function buildDiagram(graph)
{
    // load activity data and create
    var project = graph.project;
    var diagramNode = diagram.factory.createShapeNode(bounds);
    diagramNode.text = project.name;
    diagramNode.brush = brush;
    if (project.activities)
        addActivities(diagramNode, project);
}
```

10. Define addActivities as follows. It takes as parameters the parent DiagramNode and parent JSON element, iterates the child JSON elements to build the next level of the hierarchy, and creates corresponding diagram items by calling createShapeNode and createDiagramLink.

JavaScript

```
function addActivities(parentNode, parent)
{
    for (var activity of parent.activities)
    {
        var diagramNode = diagram.factory.createShapeNode(bounds);
        diagramNode.text = activity.name;
        diagramNode.brush = brush;
        diagram.factory.createDiagramLink(parentNode, diagramNode);

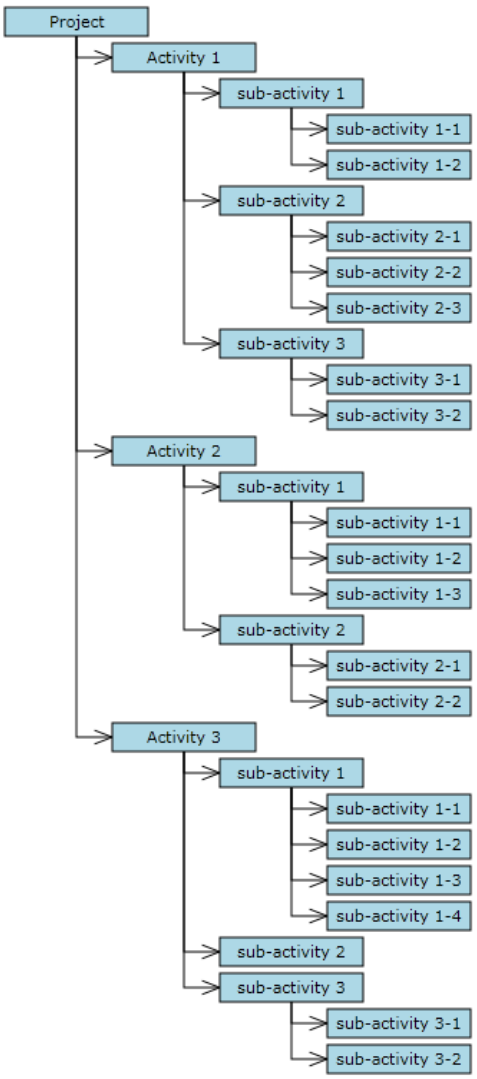
        if (activity.activities)
            addActivities(diagramNode, activity);
    }
}
```

11. Use **BorderedTreeLayout** to arrange the diagram. Add the following code to the end of the buildDiagram function.

JavaScript

```
var layout = new BorderedTreeLayout();
layout.direction = MindFusion.Graphs.LayoutDirection.LeftToRight;
layout.linkType = MindFusion.Graphs.TreeLayoutLinkType.Cascading;
layout.nodeDistance = 1;
layout.levelDistance = -8; // let horizontal positions overlap
diagram.arrange(layout);
diagram.resizeToFitItems(30);
```

12. Publish MindFusion.*.js and tutorial files using web server of your choice. Now you should see this if you open Tutorial2.html in a web browser:



Tutorial 3: Create a Custom Node Type

This tutorial shows how to define a custom node class that adds several new properties and graphics elements through a template. The code is also available in Samples folder of JsDiagram distribution.

1. Create empty Tutorial3.html and Tutorial3.js files, and add DIV and CANVAS elements to the HTML page.

HTML

```
<!-- The DiagramView component is bound to the canvas element below -->
<div style="width: 100%; height: 100%; overflow: auto;">
    <canvas id="diagram" width="2100" height="2100">
        This page requires a browser that supports HTML 5 Canvas element.
    </canvas>
</div>
```

2. Add script references to the MindFusion UMD scripts and Tutorial3 JavaScript files:

HTML

```
<script src="umd/collections.js" type="text/javascript"></script>
<script src="umd/drawing.js" type="text/javascript"></script>
<script src="umd/controls.js" type="text/javascript"></script>

<script src="umd/animations.js" type="text/javascript"></script>
<script src="umd/graphs.js" type="text/javascript"></script>
<script src="umd/diagramming.js" type="text/javascript"></script>
```

```
<script src="Tutorial3.js" type="text/javascript"></script>
```

3. In Tutorial3.js, assign shorter names to the DiagramView, Diagram, CompositeNode, Behavior, Alignment and Rect types (if using diagram's ES6 or CommonJS modules, these would be **import** or **require** statements for respective classes and modules).

JavaScript

```
var DiagramView = MindFusion.Diagramming.DiagramView;
var Diagram = MindFusion.Diagramming.Diagram;
var CompositeNode = MindFusion.Diagramming.CompositeNode;
var Behavior = MindFusion.Diagramming.Behavior;

var Alignment = MindFusion.Drawing.Alignment;
var Rect = MindFusion.Drawing.Rect;
```

4. In Tutorial3.js call the classFromTemplate static method of **CompositeNode** to generate a node class from a template. The template is defined through an object literal that lists layout containers and graphics primitives from which a node should be composed.

JavaScript

```
var OrgChartNode = CompositeNode.classFromTemplate("OrgChartNode",
{
    component: "GridPanel",
    rowDefinitions: ["*"],
    columnDefinitions: ["22", "*"],
    children:
    [
        {
            component: "Rect",
            name: "Background",
            pen: "black",
            brush: "white",
            columnSpan: 2
        },
        {
            component: "Image",
            name: "Image",
            autoProperty: true,
            location: "ceo.png",
            margin: "1",
            imageAlign: "Fit"
        },
        {
            component: "StackPanel",
            orientation: "Vertical",
            gridColumn: 1,
            margin: "1",
            verticalAlignment: "Near",
            children:
            [
                {
                    component: "Text",
                    name: "Title",
                    autoProperty: true,
                    text: "title",
                    font: "Calibri bold"
                },
                {
                    component: "Text",
                    name: "FullName",
                    autoProperty: true,
                    text: "full name",
                    pen: "blue",
                    font: "Calibri 4",
                    padding: "1,0,1,0"
                },
                {
                    component: "Text",
```

```
                    name: "Details",
                    autoProperty: true,
                    text: "details",
                    font: "Calibri 3"
                }
            ]
        }
    ]
});

Diagram.registerClass(OrgChartNode, "OrgChartNode", "OrgChartNode", 1, CompositeNode);
```

5. Create a **DiagramView** instance wrapping the HTML Canvas element. Add some nodes of the new class and create links between them. The **autoProperty** attribute in template above will generate setter methods for properties whose name is specified via **name** attribute.

JavaScript

```
var diagram = null;

document.addEventListener("DOMContentLoaded", function ()
{
    // create a DiagramView component that wraps the "diagram" canvas
    var diagramView = DiagramView.create(document.getElementById("diagram"));
    diagram = diagramView.diagram;

    // enable drawing of custom nodes interactively
    diagram.customNodeType = OrgChartNode;
    diagramView.behavior = Behavior.Custom;

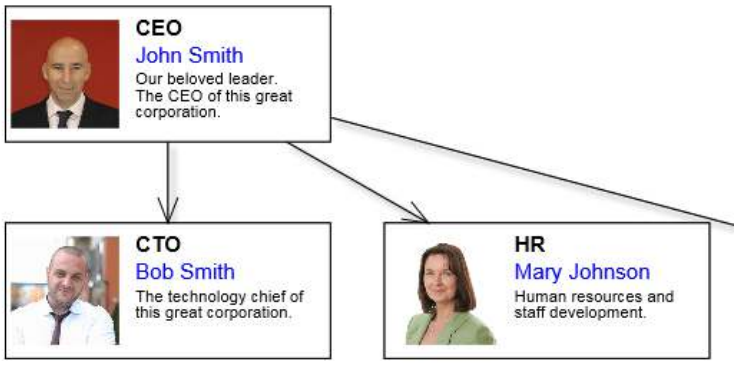
    var node1 = new OrgChartNode(diagram);
    node1.bounds = new Rect(25, 15, 60, 25);
    node1.title = "CEO";
    node1.fullName = "John Smith";
    node1.details =
        "Our beloved leader. \r\n" +
        "The CEO of this great corporation.";
    node1.image = "ceo.png";
    diagram.addItem(node1);

    var node2 = new OrgChartNode(diagram);
    node2.bounds = new Rect(25, 55, 60, 25);
    node2.title = "CTO";
    node2.fullName = "Bob Smith";
    node2.details = "The technology chief of this great corporation.";
    node2.image = "cto.png";
    diagram.addItem(node2);

    // ...

    diagram.factory.createDiagramLink(node1, node2);
});
```

6. Publish MindFusion.*.js and tutorial files using web server of your choice. Now you should see this if you open Tutorial3.html in a web browser:



Tutorial 4: Custom Nodes Integration

This tutorial builds upon the OrgChartNode type from Tutorial 3. It shows how to fully integrate the custom type with other MindFusion.Diagramming for JavaScript features, such as clipboard support, undo support, creating nodes by drag-and-drop or drawing with the mouse on the canvas. The code is also available in Samples folder of JsDiagram distribution.

1. Create copies of **Tutorial 3** files - Tutorail3.html and Tutorial3.js and name them Tuturial4.html and Tutorial4.js respectively.

2. Add a DIV and CANVAS elements to Tutorial4.html that will represent a NodeListView control:

HTML

```
<!-- The NodeList component is bound to the canvas element below -->
<div style="position: absolute; top: 0px; bottom: 0px; left: 2px; right: 0px; width: 240px;
height: 100%;">
    <canvas id="nodeListView" width="240">
    </canvas>
</div>
```

3. Add five buttons to the page - copy, cut, paste, undo and redo - and define click handlers that will be used to call the corresponding diagram methods.

HTML

```
<button onclick="diagramView.copyToClipboard()">copy</button>
<button onclick="diagramView.cutToClipboard()">cut</button>
<button onclick="diagramView.pasteFromClipboard(5, 5)">paste</button>
<button onclick="diagram.undo()">undo</button>
<button onclick="diagram.redo()">redo</button>
```

4. Add shortcuts to the **NodeListView** and Size classes in Tutorial4.js and define a nodeListView variable.

JavaScript

```
var NodeListView = MindFusion.Diagramming.NodeListView;

var Size = MindFusion.Drawing.Size;

var nodeListView = null;
```

5. Classes generated by calling classFromTemplate implement undo/redo, clipboard and serialization support for their auto-properties out of the box. When creating more complex types with their own data properties, you'd need to implement serialization and cloning for them yourself. Lets show how to implement the various operations by creating a new class that adds a custom property. Start by renaming the class from tutorial 3 to TemplatedBase, and creating a new OrgChartNode class that derives from it.

JavaScript

```
var TemplatedBase = CompositeNode.classFromTemplate("TemplatedBase",
    // ...

function OrgChartNode(parent) {
    var _this = TemplatedBase.call(this, parent);
    _this.cost = "low";
    return _this;
};

OrgChartNode.prototype.getCost = function () {
    return this.cost;
};

OrgChartNode.prototype.setCost = function (value) {
    this.cost = value;
};

Diagram.registerClass(OrgChartNode, "OrgChartNode", "OrgChartNode", 1, TemplatedBase);
```

6. Add a clone method override to the OrgChartNode prototype.

JavaScript

```
// support for the NodeListView drag and drop
OrgChartNode.prototype.clone = function () {
    var copy = TemplatedBase.prototype.clone.call(this);
    copy.cost = this.cost;
    return copy;
};
```

7. To implement clipboard support of any new properties added by a custom item class, add fromJson and toJson method overrides to the prototype definition:

JavaScript

```
// clipboard and serialization support
OrgChartNode.prototype.toJson = function () {
    var json = TemplatedBase.prototype.toJson.call(this);
    json.cost = this.cost;
    return json;
};

OrgChartNode.prototype.fromJson = function (json) {
    TemplatedBase.prototype.fromJson.call(this, json);
    this.cost = json.cost;
};
```

8. In order to enable undo / redo of changes done on OrgChartNode, add saveState and restoreState method overrides to the prototype definition:

JavaScript

```
// undo/redo
OrgChartNode.prototype.saveState = function () {
    var state = TemplatedBase.prototype.saveState.call(this);
    state.cost = this.cost;
    return state;
};

OrgChartNode.prototype.restoreState = function (state) {
    TemplatedBase.prototype.restoreState.call(this, state);
    this.cost = state.cost;
};
```

9. In order to enable drawing OrgChartNodes with the mouse, add the following code to the **DOMContentLoaded** handler:

JavaScript

```
// enable drawing of custom nodes interactively
diagramView.behavior = Behavior.Custom;
diagram.customNodeType = OrgChartNode;
```

10. To enable undo / redo support on the Diagram object, set its undoEnabled property to **true** in the **DOMContentLoaded** handler.

JavaScript

```
// enable undo/redo support
diagram.undoEnabled = true;
```

11. Finally, add initialization code for the **NodeListView** to the load hadler.

JavaScript

```
// create a NodeListView component that wraps the "nodeListView" canvas
nodeListView = NodeListView.create(document.getElementById("nodeListView"));
nodeListView.measureUnit = 6;
nodeListView.padding = 1;
nodeListView.iconSize = new Size(65, 25);
nodeListView.defaultNodeSize = new Size(65, 25);

// create and add some nodes to the node list
var node1 = new OrgChartNode(nodeListView);
node1.title = "Skiing";
node1.fullName = "Winter";
node1.details =
    "Vacation in the snow. \r\n" +
    "Ski lessons, ski equipment and winter sports.";
node1.image = "snowman.png";
node1.cost = "high";
nodeListView.addNode(node1, "");

var node2 = new OrgChartNode(nodeListView);
node2.title = "Seaside";
node2.fullName = "Summer";
node2.details = "Sun bathing, swimming, catching tan.";
node2.image = "seaside.png";
node2.cost = "high";
nodeListView.addNode(node2, "");
```

12. Publish MindFusion.*.js and tutorial files using web server of your choice. Open Tutorial4.html in a web browser and you should be able to create nodes by drag and drop from **NodeListView**, copy and paste them, and redo/undo operations done on them.

Integration with UI Frameworks

React

Functional components

The **DiagramView** React component allows integrating the MindFusion.Diagramming API into React applications. It renders a Diagram instance assigned to "diagram" prop as its model.

At this time, the following properties can be set from JSX too: allowInplaceEdit, autoResize, backBrush, behavior, defaultShape, enabled, linkHeadShapeSize, linkShape, roundedLinks, routeLinks, showAnchors, showGrid, zoomFactor. All diagram events can be handled through JSX syntax as well:

JSX

```
const [diagram] = useState(new Diagramming.Diagram());
...
<DiagramView diagram={diagram} {...props}
    onDiagramChanged={(diagram, args) => onDiagramChanged(diagram, args)}
    onNodeCreated={(diagram, args) => onDiagramNodeCreated(diagram, args)} />
```

The **DiagramView** control exposes a **forwardRef**, that can be passed on to other controls, such as the Overview and ZoomControl. To obtain a reference to the underlying core controls, use the respective find method of the **ref**.

DiagramView can be installed from the @mindfusion/diagramming-react package on npm:

npm i @mindfusion/diagramming-react

Several React examples are included in JsDiagram's distribution. In order to run them, execute the following npm commands from command line in Samples/React folder:

npm install
npm start

Class components

Older class-component wrappers have been moved to the @mindfusion/diagramming-react-cc package. Using them looks like this:

JSX

```
this.state.diagram = new Diagramming.Diagram();
...
<DiagramView
    diagram={this.state.diagram}
    {...props}
    onNodeClicked={(diagram, args) => this.onDiagramNodeClicked(diagram, args)}
    onLinkCreating={(diagram, args) => this.onDiagramLinkCreating(diagram, args)}
    onDiagramChanged={diagram => this.onDiagramChanged(diagram)} />
```

Class components can be installed from the @mindfusion/diagramming-react-cc package on npm:

npm i @mindfusion/diagramming-react-cc

Vue.js

The diagram-view Vue.js component allows integrating the MindFusion.Diagramming API into Vue.js applications. It renders a Diagram instance assigned to "diagram" prop as its model.

At this time the following properties can be set from vue templates too: allowInplaceEdit, autoResize, backBrush, behavior, defaultShape, enabled, linkHeadShapeSize, linkShape, roundedLinks, routeLinks, showAnchors, showGrid, zoomFactor. All diagram events can be handled through the v-on directive as well:

vue

```
this.diagram = new Diagramming.Diagram();
...
<diagram-view
    id="diagram"
    :diagram="this.diagram"
    :allow-inplace-edit="true"
    :route-links="true"
    :show-grid="true"
    back-brush="lightcyan"
    v-on:node-created="onDiagramNodeCreated"
    v-on:diagram-changed="onDiagramChanged"
/>
```

diagram-view can be installed from the @mindfusion/diagramming-vue package on npm:

npm i @mindfusion/diagramming-vue

Several Vue.js examples are included in JsDiagram's distribution. In order to run them, execute the following npm commands from command line in Samples/Vue folder:

npm install
npm run serve

Angular

The DiagramView Angular component (diagram-view selector) allows integrating the MindFusion.Diagramming API into Angular applications. It renders a Diagram instance assigned to "diagram" property as its model.

At this time the following properties can be set from HTML templates too: allowInplaceEdit, autoResize, backBrush, behavior, defaultShape, enabled, linkHeadShapeSize, linkShape, roundedLinks, routeLinks, showAnchors, showGrid, zoomFactor. All diagram events can be handled through event binding syntax as well:

app.module.ts

```
import { DiagrammingAngularModule } from '@mindfusion/diagramming-angular';
```

app.component.html

```
<diagram-view
    id='diagram'
    [behavior]="behavior"
    [diagram]="diagram"
    [allowInplaceEdit]="true"
    [routeLinks]="true"
    [showGrid]="true"
    [backBrush]="'lightcyan'"
    (nodeCreated)="onNodeCreated($event)"
    (nodeDeleted)="onNodeDeleted($event)"
    (linkCreated)="onLinkCreated($event)"
    (linkDeleted)="onLinkDeleted($event)">
</diagram-view>
```

app.component.ts

```
import * as Diagramming from '@mindfusion/diagramming';
...
// declare properties
autoResize = Diagramming.AutoResize.None;
behavior = Diagramming.Behavior.DrawShapes;
diagram : Diagramming.Diagram;
...
// instantiate a Diagram model
this.diagram = new Diagramming.Diagram();
```

DiagramView can be installed from the *@mindfusion/diagramming-angular* package on npm:

npm i @mindfusion/diagramming-angular

Several Angular examples are included in JsDiagram's distribution. In order to run them, execute the following npm commands from command line in Samples/Angular folder:

npm install
npm start

Blazor

JsDiagram's Blazor bindings allow integrating the JavaScript diagramming library into Blazor applications. It contains a set of .NET wrapper classes, that use Blazor's JSInterop API to create and manipulate the client-side objects. The package provides easy access from C# code to the most of the MindFusion.Diagramming functionality, including different node types, layouts and styling. Additional UI components such as Overview, NodeListView, ZoomControl and Ruler are included too.

The following steps describe how to add a DiagramView to Blazor application:

1. Create a new Blazor WebAssembly App project.
2. Install the MindFusion.Diagramming.JavaScript NuGet package.
3. Add the importmap script for @mindfusion modules:

wwwroot/index.html:

```
<head>
    <script type="importmap">
    { "imports":
        {

"@mindfusion/collections":"./_content/MindFusion.Diagramming.JavaScript/collections.js",
            "@mindfusion/controls":"./_content/MindFusion.Diagramming.JavaScript/controls.js",
            "@mindfusion/drawing":"./_content/MindFusion.Diagramming.JavaScript/drawing.js",

"@mindfusion/diagramming":"./_content/MindFusion.Diagramming.JavaScript/diagramming.js",
            "@mindfusion/graphs":"./_content/MindFusion.Diagramming.JavaScript/graphs.js",

"@mindfusion/animations":"./_content/MindFusion.Diagramming.JavaScript/animations.js"
        }
    }
    </script>
    ...
```

4. Add imports to the top of _Imports.razor (or the top of Index.razor page):

razor

```
@using MindFusion.Diagramming.JavaScript
```

5. Add the **DiagramView** component razor tag:

razor

```
<DiagramView @ref="diagramView"></DiagramView>
```

Optionally add styles:

razor

```
<style>
    .mf_diagramView {
        height: 80vh;
    }
</style>
```

Optionally set component parameters and add event handlers:

razor

```
<DiagramView @ref="diagramView"
    Behavior="Behavior.LinkTables"
    NodeCreated="OnNodeCreated"></DiagramView>
```

6. Create the **DiagramView** instance:

C#

```
@code
{
    private DiagramView diagramView = null!;

    protected override async Task OnAfterRenderAsync(bool firstRender)
    {
        if (firstRender)
        {
            await diagramView.CreateAsync();
        }
    }

    // The event handler for the NodeCreated event
    private async Task OnNodeCreated(NodeEventArgs args)
    {
        args.Node.Text = "some text";
    }
}
```

7. Obtain a reference to the Diagram instance and add nodes and links

C#

```
@code
{
    private DiagramView diagramView = null!;
    private Diagram diagram = null!;

    protected override async Task OnAfterRenderAsync(bool firstRender)
    {
        if (firstRender)
        {
            await diagramView.CreateAsync();

            diagram = diagramView.Diagram;
            diagram.RouteLinks = true;

            // create a Container node
            var container = await diagram.Factory.CreateContainerNode(10, 20, 40, 60);

            // create a Shape node
            var shapeNode = await diagram.Factory.CreateShapeNode(100, 20, 20, 20);
            shapeNode.Shape = Shape.FromId("Decision");

            // create a link
            diagram.Factory.CreateDiagramLink(container, shapeNode);
        }
    }
}
```

Alternatively, the **Diagram** instance can be created first and then passed to the **DiagramView** component as a parameter of its CreateAsync method. To create the **Diagram** instance use the **Diagram**'s static CreateAsync method.

C#

```
@inject IJSRuntime JS
...
    await MindFusion.JavaScript.BlazorInterop.ImportModules(JS!);
    Diagram diagram = await Diagram.CreateAsync();
    // ... set diagram properties/items/etc. here
    await diagramView.CreateAsync(diagram);

// or

    Diagram diagram = await Diagram.CreateAsync(JS!);
    // ... set diagram properties/items/etc. here
    await diagramView.CreateAsync(diagram);
```

References to the JS modules can be disposed by using the DisposeModules method

C#

```
async ValueTask IAsyncDisposable.DisposeAsync()
{
    await MindFusion.JavaScript.BlazorInterop.DisposeModules();
}
```

Reference

MindFusion.Animations Namespace

Animation Class

Represents an object that is used to animate diagram items on the canvas.

Syntax

JavaScript

```
// class
Animation.prototype = {}
```

Remarks

For more information on how to use the **Animation** class, please refer to the Animations topic in the Programming Interface Overview.

Inheritance Hierarchy

MindFusion.Animations.Animation
MindFusion.Diagramming.Animations.ItemAnimation
MindFusion.Diagramming.Animations.ViewAnimation

See Also

Animation Members
MindFusion.Animations Namespace

Animation Members

Represents an object that is used to animate diagram items on the canvas.

The following tables list the members exposed by the Animation type.

Public Constructors

	Name	Description
	Animation	Initializes a new instance of the **Animation** class.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation.
	duration	Gets or sets the duration of the animation function in milliseconds.
	easingType	Gets or sets the easing type of the animation.
	fromValue	Gets or sets the initial value of the property that will be animated.
	isRunning	Indicates whether the animation is currently running.
	param	Gets or sets an optional value passed to the animation function.
	progress	Gets the progress of the current animation iteration.
	repeat	Gets or sets a value indicating whether to restart the animation after it completes.
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes.
	toValue	Gets or sets the target value of the property that will be animated.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**.
	dispose	Sys.IDisposable implementation.
	raiseEvent	Raises an event.
	removeEventListener	Removes a single event listener attached to the **Animation**.
	start	Starts or resumes the animation.
	stop	Stops the animation.

See Also

Animation Class
MindFusion.Animations Namespace

Animation Constructor

Initializes a new instance of the Animation class.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function Animation (item, [options, [onUpdateCallback, [onCustomFunctionCallback]]])
```

Parameters

item
The diagram item to animate.

options
Optional.
An object used to specify various property values for the animation.

onUpdateCallback
Optional.
A callback function called on every animation tick.

onCustomFunctionCallback
Optional.
A callback function called when the animation type is set to Custom.

See Also

Animation Class
Animations
MindFusion.Animations Namespace

Animation Methods

Animation.addEventListener Method

Registers a single event listener on the Animation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function addEventListener (eventName, handler)
```

Parameters

eventName
The name of the event.

handler
Represents the method that will handle the event specified with eventName.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.dispose Method

Sys.IDisposable implementation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function dispose ()
```

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.raiseEvent Method

Raises an event.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function raiseEvent (eventName, args)
```

Parameters

eventName
The name of the event to raise.

args
An instance of type EventArgs that holds data for the event specified with eventName.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.removeEventListener Method

Removes a single event listener attached to the Animation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function removeEventListener (eventName, handler)
```

Parameters

eventName
The name of the event.

handler
Represents the method that handles the event specified with eventName.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.start Method

Starts or resumes the animation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function start ()
```

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.stop Method

Stops the animation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
function stop ()
```

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation Properties

Animation.animationType Property

Gets or sets the type of the animation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get animationType() {}
set animationType(value) {}
```

Property Value

AnimationType

A member of the AnimationType enumeration. The default value is **Linear**.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.duration Property

Gets or sets the duration of the animation function in milliseconds.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get duration() {}
set duration(value) {}
```

Property Value

Number

The duration of the animation. The default value is 1000.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.easingType Property

Gets or sets the easing type of the animation.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get easingType() {}
set easingType(value) {}
```

Property Value

EasingType

A member of the EasingType enumeration. The default value is **EaseIn**.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.fromValue Property

Gets or sets the initial value of the property that will be animated.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get fromValue() {}
set fromValue(value) {}
```

Property Value

Object

The initial value.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.isRunning Property

Indicates whether the animation is currently running.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get isRunning() {}
set isRunning(value) {}
```

Property Value

Boolean

true if animation currently in progress, or **false** otherwise.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.param Property

Gets or sets an optional value passed to the animation function.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get param() {}
set param(value) {}
```

Property Value

Object

The optional parameter. The default value is **null**.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.progress Property

Gets the progress of the current animation iteration.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get progress() {}
set progress(value) {}
```

Property Value

Number

Animation progress in 0-1 range.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.repeat Property

Gets or sets a value indicating whether to restart the animation after it completes.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
get repeat() {}
set repeat(value) {}
```

Property Value

Boolean

true to loop the animation, or **false** otherwise. The default value is **false**.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.reverse Property

Gets or sets a value indicating whether to swap the from and to values after an animation loop completes.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

JavaScript

```
get reverse() {}
set reverse(value) {}
```

Property Value

Boolean

true to reverse the animation, or **false** otherwise. The default value is **false**.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

Animation.toValue Property

Gets or sets the target value of the property that will be animated.

Namespace: MindFusion.Animations
File: Animation.js

Syntax

```
JavaScript
get toValue() {}
set toValue(value) {}
```

Property Value

Object

The target value.

See Also

Animation Members
Animation Class
MindFusion.Animations Namespace

AnimationType Enumeration

Specifies the animation function used by an Animation object.

Namespace: MindFusion.Animations
File: Enum.js

Syntax

```
JavaScript
// enum
AnimationType = {}
```

Members

	Member name	Description
	BackBow	Represents an effect that simulates a retraction in the motion of the animation. The amplitude of the retraction can be controlled through the Animation's param property. The default param value is 1.5;
	Bounce	Represents a bouncing effect animation.
	Circular	Represents an animation that accelerates/deccelerates using a circular function.

	Custom	Specifies a user-set function. When this value is used the actual animation function must be set through the Animation's custom function callback.
	Elastic	Represents an oscillating effect animation. The oscillation frequency can be controlled through the Animation's param property. The default param value is 0.5;
	Exponential	Represents an animation that accelerates/deccelerates using an exponential function. The exponent can be set through the Animation's param property. The default exponent equals 10.
	Linear	Represents an animation that accelerates/deccelerates using a linear function.
	Power	Represents an animation that accelerates/deccelerates through exponentiation formula - x^n where n can be set through the Animation's param property. The default exponent equals 2.

Remarks

Members of this enumeration can be assigned to the animationType property of Animation.

See Also

MindFusion.Animations Namespace

EasingType Enumeration

Specifies the easing applied to an animation.

Namespace: MindFusion.Animations
File: Enum.js

Syntax

JavaScript

```
// enum
EasingType = {}
```

Members

	Member name	Description
	EaseIn	Follows the animation function associated with the Animation.
	EaseInOut	Uses EaseIn for the first half the the Animation's duration and EaseOut for the second half.
	EaseOut	Inverses the animation function associated with the **Animation**.
	EaseOutIn	Uses EaseOut for the first half the the Animation's duration and EaseIn for the second half.

Remarks

Members of this enumeration can be assigned to the easingType property of **Animation**.

See Also

MindFusion.Animations Namespace

Events Class

Defines all events raised in the Animations namespace.

Syntax

JavaScript

```
// class
Events.prototype = {}
```

Inheritance Hierarchy

MindFusion.Animations.Events

See Also

Events Members
MindFusion.Animations Namespace

Events Members

Defines all events raised in the Animations namespace.

The following tables list the members exposed by the Events type.

Public Properties

	Name	Description
	animationComplete	Raised when an Animation is complete.

See Also

Events Class
MindFusion.Animations Namespace

Events Fields

Events Properties

Events.animationComplete Property

Raised when an Animation is complete.

Namespace: MindFusion.Animations
File: Events.js

Syntax

```
JavaScript
get animationComplete() {}
set animationComplete(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Animations Namespace

MindFusion.Collections Namespace

Grid Class

Represents a collection of tabular data.

Syntax

```
JavaScript
// class
Grid.prototype = {}
```

Inheritance Hierarchy

See Also

Grid Members
MindFusion.Collections Namespace

Grid Members

Represents a collection of tabular data.

The following tables list the members exposed by the Grid type.

Public Constructors

	Name	Description
	Grid	Initializes a new instance of the Grid class.

See Also

Grid Class
MindFusion.Collections Namespace

Grid Constructor

Initializes a new instance of the Grid class.

Namespace: MindFusion.Collections
File: Grid.js

Syntax

```
JavaScript
function Grid ()
```

See Also

Grid Class
MindFusion.Collections Namespace

Queue Class

Represents a first-in, first-out collection of objects.

Syntax

```
JavaScript
// class
Queue.prototype = {}
```

Inheritance Hierarchy

See Also

Queue Members
MindFusion.Collections Namespace

Queue Members

Represents a first-in, first-out collection of objects.

The following tables list the members exposed by the Queue type.

Public Constructors

	Name	Description
	Queue	Initializes a new instance of the **Queue** class.

Public Methods

	Name	Description
	dequeue	Removes and returns the object at the beginning of the queue.
	enqueue	Adds an object to the end of the queue.
	getSize	Gets the number of elements in the queue.

See Also

Queue Class
MindFusion.Collections Namespace

Queue Constructor

Initializes a new instance of the Queue class.

Namespace: MindFusion.Collections
File: Queue.js

Syntax

JavaScript
`function Queue ()`

See Also

Queue Class
MindFusion.Collections Namespace

Queue Methods

Queue.dequeue Method

Removes and returns the object at the beginning of the queue.

Namespace: MindFusion.Collections
File: Queue.js

Syntax

JavaScript
`function dequeue ()`

Return Value

The object that is removed from the beginning of the queue.

Remarks

If the queue is empty this method will throw an exception.

See Also

Queue Members
Queue Class
MindFusion.Collections Namespace

Queue.enqueue Method

Adds an object to the end of the queue.

Namespace: MindFusion.Collections
File: Queue.js

Syntax

JavaScript

```
function enqueue (value)
```

Parameters

value
The object to add.

See Also

Queue Members
Queue Class
MindFusion.Collections Namespace

Queue.getSize Method

Gets the number of elements in the queue.

Namespace: MindFusion.Collections
File: Queue.js

Syntax

JavaScript

```
function getSize ()
```

Return Value

See Also

Queue Members
Queue Class
MindFusion.Collections Namespace

MindFusion.Controls Namespace

Alignment Enumeration

Specifies the alignment of text relative to its layout rectangle.

Namespace: MindFusion.Controls
File: Enum.js

Syntax

```
JavaScript
// enum
Alignment = {}
```

Members

	Member name	Description
	Center	The text is drawn in the center of the layout rectangle.
	Far	The text is drawn in the far corner of the layout rectangle.
	Near	The text is drawn in the near corner of the layout rectangle.

See Also

MindFusion.Controls Namespace

CancelEventArgs Class

Provides a value to use with cancellable events.

Syntax

```
JavaScript
// class
CancelEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.EventArgs
  MindFusion.Controls.CancelEventArgs
    MindFusion.Diagramming.ItemEventArgs
    MindFusion.Diagramming.SelectionEventArgs
    MindFusion.Diagramming.TreeItemEventArgs

See Also

CancelEventArgs Members
MindFusion.Controls Namespace

CancelEventArgs Members

Provides a value to use with cancellable events.

The following tables list the members exposed by the CancelEventArgs type.

Public Constructors

	Name	Description
	CancelEventArgs	Initializes a new instance of the CancelEventArgs class.

Public Properties

	Name	Description
	cancel	Gets or sets a value indicating whether to allow the current operation.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)

See Also

CancelEventArgs Class
MindFusion.Controls Namespace

CancelEventArgs Constructor

Initializes a new instance of the CancelEventArgs class.

Namespace: MindFusion.Controls
File: EventArgs.js

Syntax

JavaScript

```
function CancelEventArgs ()
```

See Also

CancelEventArgs Class
MindFusion.Controls Namespace

CancelEventArgs Properties

CancelEventArgs.cancel Property

Gets or sets a value indicating whether to allow the current operation.

Namespace: MindFusion.Controls
File: EventArgs.js

Syntax

JavaScript
`get cancel() {}` `set cancel(value) {}`

Property Value

Boolean

true to cancel the operation, or **false** otherwise. The default value is **false**.

See Also

CancelEventArgs Members
CancelEventArgs Class
MindFusion.Controls Namespace

Canvas Class

Handles drawing on a CanvasRenderingContext2D.

Syntax

JavaScript
`// class` `Canvas.prototype = {}`

Inheritance Hierarchy

MindFusion.Controls.Disposable
  MindFusion.Controls.Canvas

See Also

Canvas Members
MindFusion.Controls Namespace

Canvas Members

Handles drawing on a CanvasRenderingContext2D.

The following tables list the members exposed by the Canvas type.

Public Constructors

	Name	Description
	Canvas	Initializes a new instance of the Canvas class.

Public Properties

	Name	Description
	bounds	Gets the underlying **Canvas** element's logical bounds.

	element	Returns a reference to the control's DOM element. (Inherited from Disposable.)
	enabled	Gets or sets whether mouse events are enabled. (Inherited from Disposable.)
	measureUnit	Gets or sets the unit of measure used for logical coordinates.
	minVisibleFontSize	Gets or sets a threshold value that hides text if scaled font sizes become smaller.
	scale	Gets the current scale of this **Canvas**.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
	dispose	Overrides Disposable.dispose
	invalidate	Invalidates the canvas or a region of the canvas, causing it to be repainted.
	raiseEvent	Raises an event. (Inherited from Disposable.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
	repaint	Repaints the canvas.

See Also

Canvas Class
MindFusion.Controls Namespace

Canvas Constructor

Initializes a new instance of the Canvas class.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

JavaScript

```
function Canvas ()
```

See Also

Canvas Class
MindFusion.Controls Namespace

Canvas Methods

Canvas.dispose Method

Overrides Disposable.dispose

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

JavaScript

```
function dispose ()
```

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace

Canvas.invalidate Method

Invalidates the canvas or a region of the canvas, causing it to be repainted.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

JavaScript

```
function invalidate ([rect, [force]])
```

Parameters

rect
Optional.
Rect. A Rect instance specifying the region that should be repainted. If not specified, the whole diagram will be repainted.

force
Optional.
Boolean. **true** to force redraw even if currently painting, or **false** otherwise.

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace

Canvas.repaint Method

Repaints the canvas.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

```
JavaScript
function repaint ([printOptions])
```

Parameters

printOptions
Optional.
Object. For internal use.

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace

Canvas Properties

Canvas.bounds Property

Gets the underlying Canvas element's logical bounds.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

```
JavaScript
get bounds() {}
set bounds(value) {}
```

Property Value

Rect

A Rect instance specifying canvas bounds in current measureUnit.

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace
Diagram.enableSpatialIndex Property

Canvas.measureUnit Property

Gets or sets the unit of measure used for logical coordinates.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

JavaScript

```
get measureUnit() {}
set measureUnit(value) {}
```

Property Value

GraphicsUnit

A member of the GraphicsUnit enumeration.

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace

Canvas.minVisibleFontSize Property

Gets or sets a threshold value that hides text if scaled font sizes become smaller.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

JavaScript

```
get minVisibleFontSize() {}
set minVisibleFontSize(value) {}
```

Property Value

Number

A value specifying the smallest visible font size.

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace

Canvas.scale Property

Gets the current scale of this Canvas.

Namespace: MindFusion.Controls
File: Canvas.js

Syntax

JavaScript

```
get scale() {}
set scale(value) {}
```

Property Value

Number

The canvas scale.

See Also

Canvas Members
Canvas Class
MindFusion.Controls Namespace

CanvasControl Class

The CanvasControl class represents a wrapper class for the HTML5 Canvas element.

Syntax

JavaScript
`// class` `CanvasControl.prototype = {}`

Inheritance Hierarchy

MindFusion.Controls.Disposable
  MindFusion.Controls.Control
    MindFusion.Controls.CanvasControl
      MindFusion.Controls.ZoomControl
      MindFusion.Diagramming.ListViewBase
      MindFusion.Diagramming.Overview

See Also

CanvasControl Members
MindFusion.Controls Namespace

CanvasControl Members

The CanvasControl class represents a wrapper class for the HTML5 Canvas element.

The following tables list the members exposed by the CanvasControl type.

Public Constructors

	Name	Description
	CanvasControl	Initializes a new instance of the CanvasControl class.

Public Properties

	Name	Description
	element	Returns a reference to the control's DOM element. (Inherited from Control.)

	enabled	Gets or sets whether mouse events are enabled. (Inherited from Control.)
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	measureUnit	Gets the unit of measure used for logical coordinates.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)
	clientToDoc	Transforms a point from client to document coordinates.
	docToClient	Transforms a point from document to client coordinates.
S	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data. (Inherited from Control.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
S	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description
	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
	controlUnloading	Raised when the control is unloading. (Inherited from Control.)

See Also

CanvasControl Class
MindFusion.Controls Namespace

CanvasControl Constructor

Initializes a new instance of the CanvasControl class.

Namespace: MindFusion.Controls
File: CanvasControl.js

Syntax

```
JavaScript
function CanvasControl (element, [canvas])
```

Parameters

element
HTMLCanvasElement. The Canvas DOM Element this CanvasControl is associated with.

canvas
Optional.
Canvas. The Canvas instance this CanvasControl is associated with.

See Also

CanvasControl Class
MindFusion.Controls Namespace

CanvasControl Methods

CanvasControl.clientToDoc Method

Transforms a point from client to document coordinates.

Namespace: MindFusion.Controls
File: CanvasControl.js

Syntax

```
JavaScript
function clientToDoc (point)
```

Parameters

point
Point. The point to transform.

Return Value

Point. The transformed point.

See Also

CanvasControl Members
CanvasControl Class
MindFusion.Controls Namespace

CanvasControl.docToClient Method

Transforms a point from document to client coordinates.

Namespace: MindFusion.Controls
File: CanvasControl.js

Syntax

```
JavaScript
function docToClient (point)
```

Parameters

point
Point. The point to transform.

Return Value

Point. The transformed point.

See Also

CanvasControl Members
CanvasControl Class
MindFusion.Controls Namespace

CanvasControl Properties

CanvasControl.measureUnit Property

Gets the unit of measure used for logical coordinates.

Namespace: MindFusion.Controls
File: CanvasControl.js

Syntax

```
JavaScript
get measureUnit() {}
set measureUnit(value) {}
```

Property Value

GraphicsUnit

An instance of the GraphicsUnit class.

See Also

CanvasControl Members
CanvasControl Class
MindFusion.Controls Namespace

CanvasControl Events

Control.controlLoaded Event

Raised when the control is loaded.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
diagramView.controlLoaded.addEventListener(onViewLoaded);

function onViewLoaded(sender, args)
{
}
```

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.controlUnloaded Event

Raised when the control is unloaded.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
diagramView.controlUnloaded.addEventListener(onViewUnloaded);

function onViewUnloaded(sender, args)
{
}
```

Remarks

This event is raised by the dispose method. In vanilla component, that usually happens when the page unloads. In diagram wrapper components for JavaScrript UI frameworks, it's raised by component's lifecycle hooks indicating that DOM element is being detached, such as **ngOnDestroy**.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.controlUnloading Event

Raised when the control is unloading.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
diagramView.controlUnloading.addEventListener(onViewUnloading);

function onViewUnloading(sender, args)
{
}
```

Remarks

This event is raised by the dispose method, before detaching DOM elements. In vanilla component, that usually happens when the page unloads. In diagram wrapper components for JavaScrript UI frameworks, it's raised by component's lifecycle hooks indicating that DOM element is being detached, such as **ngOnDestroy**.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control Class

A base class for MindFusion controls.

Syntax

```
JavaScript
// class
Control.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.Disposable
  MindFusion.Controls.Control
    MindFusion.Controls.CanvasControl
    MindFusion.Diagramming.Ruler

See Also

Control Members
MindFusion.Controls Namespace

Control Members

A base class for MindFusion controls.

The following tables list the members exposed by the Control type.

Public Constructors

	Name	Description

	Control	Initializes a new instance of the Control class.

Public Properties

	Name	Description
	element	Returns a reference to the control's DOM element.
	enabled	Gets or sets whether mouse events are enabled.
	licenseLocation	Gets or sets the location of the license key file.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element.
S	clearHandlers	Removes all event listeners from a DOM element.
S	fromJson	Constructs a JavaScript object from a JSON string.
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data.
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
S	toJson	Converts a JavaScript object to a JSON string.

Public Events

	Name	Description
	controlLoaded	Raised when the control is loaded.
	controlUnloaded	Raised when the control is unloaded.
	controlUnloading	Raised when the control is unloading.

See Also

Control Class
MindFusion.Controls Namespace

Control Constructor

Initializes a new instance of the Control class.

Namespace: MindFusion.Controls
File: Control.js

Syntax

JavaScript

```
function Control (element)
```

Parameters

element
HTMLElement. The DOM Element this Control is associated with.

See Also

Control Class
MindFusion.Controls Namespace

Control Methods

Control.addHandlers Method

Adds an event listener to a DOM element.

Namespace: MindFusion.Controls
File: Control.js

Syntax

JavaScript

```
function addHandlers ()
```

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.clearHandlers Method

Removes all event listeners from a DOM element.

Namespace: MindFusion.Controls
File: Control.js

Syntax

JavaScript

```
function clearHandlers ()
```

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.fromJson Method

Constructs a JavaScript object from a JSON string.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
function fromJson ()
```

Return Value

Object. object The corresponding to the given JSON text.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.registerForSubmit Method

Registers an onsubmit handler for the Control's parent html form to flush postback data.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
function registerForSubmit (id)
```

Parameters

id
String. The id of the hidden field to flush the data to.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.toJson Method

Converts a JavaScript object to a JSON string.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
function toJson (object)
```

Parameters

object
Object. The object to stringify.

Return Value

String. The JSON string.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control Properties

Control.element Property

Returns a reference to the control's DOM element.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
get element() {}
set element(value) {}
```

Property Value

HTMLElement

The HTML element wrapped by this control.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.enabled Property

Gets or sets whether mouse events are enabled.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
get enabled() {}
set enabled(value) {}
```

Property Value

Boolean

true if the control should respond to mouse events, or false otherwise.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control.licenseLocation Property

Gets or sets the location of the license key file.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
get licenseLocation() {}
set licenseLocation(value) {}
```

Property Value

String

the location of license key file.

Remarks

Control.licenseLocation can also be set as a static property, which applies the key to all instances of controls from MindFusion.JavaScript Pack.

See Also

Control Members
Control Class
MindFusion.Controls Namespace

Control Events

Disposable Class

Disposable abstract class.

Syntax

JavaScript

```
// class
Disposable.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.Disposable
MindFusion.Controls.Canvas
MindFusion.Controls.Control

See Also

Disposable Members
MindFusion.Controls Namespace

Disposable Members

Disposable abstract class.

The following tables list the members exposed by the Disposable type.

Public Properties

	Name	Description
	element	Returns a reference to the control's DOM element.
	enabled	Gets or sets whether mouse events are enabled.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance.
	raiseEvent	Raises an event.
	removeEventListener	Removes a single event listener attached to the instance.

See Also

Disposable Class
MindFusion.Controls Namespace

Disposable Methods

Disposable.addEventListener Method

Registers a single event listener on the instance.

Namespace: MindFusion.Controls
File: Control.js

Syntax

JavaScript

```
function addEventListener (eventName, handler, [element])
```

Parameters

eventName
String. The name of the event.

handler
Method. Represents the method that will handle the event specified with eventName.

element
Optional.
Object. For internal use.

See Also

Disposable Members
Disposable Class
MindFusion.Controls Namespace
removeEventListener Method

Disposable.raiseEvent Method

Raises an event.

Namespace: MindFusion.Controls
File: Control.js

Syntax

JavaScript

```
function raiseEvent (eventName, args, [element])
```

Parameters

eventName
String. The name of the event to raise.

args
EventArgs. An instance of type EventArgs that holds data for the event specified with eventName.

element
Optional.
Object. For internal use.

See Also

Disposable Members
Disposable Class
MindFusion.Controls Namespace

Disposable.removeEventListener Method

Removes a single event listener attached to the instance.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
function removeEventListener (eventName, handler, [element])
```

Parameters

eventName
String. The name of the event.

handler
Method. Represents the method that handles the event specified with eventName.

element
Optional.
Object. For internal use.

See Also

Disposable Members
Disposable Class
MindFusion.Controls Namespace
addEventListener Method

Disposable Properties

Disposable.element Property

Returns a reference to the control's DOM element.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
get element() {}
set element(value) {}
```

Property Value

HTMLElement

The HTML element wrapped by this control.

See Also

Disposable Members
Disposable Class
MindFusion.Controls Namespace

Disposable.enabled Property

Gets or sets whether mouse events are enabled.

Namespace: MindFusion.Controls
File: Control.js

Syntax

```
JavaScript
get enabled() {}
set enabled(value) {}
```

Property Value

Boolean

true if the control should respond to mouse events, or false otherwise.

See Also

Disposable Members
Disposable Class
MindFusion.Controls Namespace

DomUtils Class

Contains DOM-related helper functions.

Syntax

```
JavaScript
// class
DomUtils.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.DomUtils

See Also

DomUtils Members
MindFusion.Controls Namespace

DomUtils Members

Contains DOM-related helper functions.

The following tables list the members exposed by the DomUtils type.

Public Methods

	Name	Description

	formatString	Formats the specified string according to the specified parameters.
	getBounds	Gets the bounds of the specified element.
	getElementAttributes	Gets an object containing element attributes that have values.
	HTMLtoXHTML	Converts HTML to XHMTL.
	loadImageList	Loads the images from the specified list of image url-s.
	setBounds	Sets the bounds of the specified element.
	setSize	Sets the size of the specified element.
	toDataUrl	Converts an image to base64 string.

See Also

DomUtils Class
MindFusion.Controls Namespace

DomUtils Methods

DomUtils.formatString Method

Formats the specified string according to the specified parameters.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function formatString ()
```

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.getBounds Method

Gets the bounds of the specified element.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function getBounds (element, [parent, [includeScroll]])
```

Parameters

element
HTMLElement. The element to check.

parent
Optional.
HTMLElement. The parent of the element. If supplied, the returned bounds will be relative to the parent element bounds.

includeScroll
Optional.
Boolean. For internal use.

Return Value

Rect. The bounding rectangle.

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.getElementAttributes Method

Gets an object containing element attributes that have values.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function getElementAttributes ()
```

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.HTMLtoXHTML Method

Converts HTML to XHMTL.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function HTMLtoXHTML (html)
```

Parameters

html
String. The HTML to convert.

Return Value

String. The resulting XHTML.

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.loadImageList Method

Loads the images from the specified list of image url-s.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function loadImageList (images)
```

Parameters

images
Array. An array of image url-s.

Return Value

Promise. Promise object contaning an array of images in base64 format.

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.setBounds Method

Sets the bounds of the specified element.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function setBounds (element, bounds)
```

Parameters

element

HTMLElement. The element.

bounds
Rect. The bounding rectangle.

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.setSize Method

Sets the size of the specified element.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function setSize (element, size)
```

Parameters

element
HTMLElement. The element.

size
Size. The new size.

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

DomUtils.toDataUrl Method

Converts an image to base64 string.

Namespace: MindFusion.Controls
File: DomUtils.js

Syntax

JavaScript

```
function toDataUrl (url, callback)
```

Parameters

url
String. The url of the image.

callback
function. The callback function.

Return Value

String. The base64-encoded string, representing the image.

See Also

DomUtils Members
DomUtils Class
MindFusion.Controls Namespace

EventArgs Class

The base type of classes that define arguments passed to event handler functions.

Syntax

JavaScript
`// class` `EventArgs.prototype = {}`

Inheritance Hierarchy

MindFusion.Controls.EventArgs
- MindFusion.Controls.CancelEventArgs
- MindFusion.Diagramming.DiagramEventArgs
- MindFusion.Diagramming.HeaderEventArgs
- MindFusion.Diagramming.InplaceEditEventArgs
- MindFusion.Diagramming.SelectionChangedEventArgs

See Also

EventArgs Members
MindFusion.Controls Namespace

EventArgs Members

The base type of classes that define arguments passed to event handler functions.

The following tables list the members exposed by the EventArgs type.

Public Constructors

	Name	Description
	EventArgs	Initializes a new instance of the EventArgs class.

Public Properties

	Name	Description
	handled	Gets or sets a value indicating whether the event has been handled.

See Also

EventArgs Class
MindFusion.Controls Namespace

EventArgs Constructor

Initializes a new instance of the EventArgs class.

Namespace: MindFusion.Controls
File: EventArgs.js

Syntax

JavaScript

```
function EventArgs ()
```

See Also

EventArgs Class
MindFusion.Controls Namespace

EventArgs Properties

EventArgs.handled Property

Gets or sets a value indicating whether the event has been handled.

Namespace: MindFusion.Controls
File: EventArgs.js

Syntax

JavaScript

```
get handled() {}
set handled(value) {}
```

Property Value

Boolean

true to indicate that the event has been handled, or false otherwise.

See Also

EventArgs Members
EventArgs Class
MindFusion.Controls Namespace

EventDispatcher Class

Represents a dispatcher for an event.

Syntax

JavaScript
`// class` `EventDispatcher.prototype = {}`

Inheritance Hierarchy

MindFusion.Controls.EventDispatcher

See Also

EventDispatcher Members
MindFusion.Controls Namespace

EventDispatcher Members

Represents a dispatcher for an event.

The following tables list the members exposed by the EventDispatcher type.

Public Constructors

	Name	Description
	EventDispatcher	Creates a new instance of the EventDispatcher class.

Public Methods

	Name	Description
	addEventListener	Subcribes an event listener to this event.
	raiseEvent	Raises the event.
	raiseEventAsync	Raises the event asynchronously.
	removeEventListener	Removes an event listener from this event.

See Also

EventDispatcher Class
MindFusion.Controls Namespace

EventDispatcher Constructor

Creates a new instance of the EventDispatcher class.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

JavaScript

```
function EventDispatcher ()
```

See Also

EventDispatcher Class
MindFusion.Controls Namespace

EventDispatcher Methods

EventDispatcher.addEventListener Method

Subcribes an event listener to this event.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

JavaScript

```
function addEventListener ()
```

See Also

EventDispatcher Members
EventDispatcher Class
MindFusion.Controls Namespace

EventDispatcher.raiseEvent Method

Raises the event.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

JavaScript

```
function raiseEvent ()
```

See Also

EventDispatcher Members
EventDispatcher Class
MindFusion.Controls Namespace

EventDispatcher.raiseEventAsync Method

Raises the event asynchronously.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

```
JavaScript
function raiseEventAsync ()
```

See Also

EventDispatcher Members
EventDispatcher Class
MindFusion.Controls Namespace

EventDispatcher.removeEventListener Method

Removes an event listener from this event.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

```
JavaScript
function removeEventListener ()
```

See Also

EventDispatcher Members
EventDispatcher Class
MindFusion.Controls Namespace

Events Class

Defines all events raised in the Controls namespace.

Syntax

```
JavaScript
// class
Events.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.Events

See Also

Events Members
MindFusion.Controls Namespace

Events Members

Defines all events raised in the Controls namespace.

The following tables list the members exposed by the Events type.

Public Properties

	Name	Description
	controlLoaded	Raised when the control is loaded.
	controlUnloaded	Raised when the control is unloaded.
	controlUnloading	Raised before the control is unloaded.

See Also

Events Class
MindFusion.Controls Namespace

Events Properties

Events.controlLoaded Property

Raised when the control is loaded.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

JavaScript

```
get controlLoaded() {}
set controlLoaded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Controls Namespace

Events.controlUnloaded Property

Raised when the control is unloaded.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

```
JavaScript
get controlUnloaded() {}
set controlUnloaded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Controls Namespace

Events.controlUnloading Property

Raised before the control is unloaded.

Namespace: MindFusion.Controls
File: EventDispatcher.js

Syntax

```
JavaScript
get controlUnloading() {}
set controlUnloading(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Controls Namespace

MouseButton Enumeration

Specifies which button was pressed to trigger the event.

Namespace: MindFusion.Controls
File: Enum.js

Syntax

```
JavaScript
// enum
MouseButton = {}
```

Members

	Member name	Description
	Auxiliary	Auxiliary button pressed, usually the wheel button or the middle button.
	Fifth	Fifth button, typically the Browser Forward button.
	Fourth	Fourth button, typically the Browser Back button.
	Main	Main button pressed, usually the left button or the un-initialized state.
	Secondary	Secondary button pressed, usually the right button.

See Also

MindFusion.Controls Namespace

MouseCursors Enumeration

Specifies the type of the mouse cursor.

Namespace: MindFusion.Controls
File: Enum.js

Syntax

JavaScript

```
// enum
MouseCursors = {}
```

See Also

MindFusion.Controls Namespace

TickPosition Enumeration

Specifies the position of trackbar ticks.

Namespace: MindFusion.Controls
File: Enum.js

Syntax

JavaScript

```
// enum
TickPosition = {}
```

Members

	Member name	Description

	Both	Ticks are rendered on both sides of the trackbar.
	Left	Ticks are rendered on the left side of the trackbar.
	None	There are no ticks rendered.
	Right	Ticks are rendered on the right side of the trackbar.

Remarks

Members of this enumeration can be assigned to the tickPosition property of ZoomControl.

See Also

MindFusion.Controls Namespace

WebComponents Class

Implements registration of web components.

Syntax

JavaScript

```
// class
WebComponents.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.WebComponents

See Also

WebComponents Members
MindFusion.Controls Namespace

WebComponents Members

Implements registration of web components.

The following tables list the members exposed by the WebComponents type.

Public Methods

	Name	Description
	register	Registers the controls in the package as web components.

See Also

WebComponents Class
MindFusion.Controls Namespace

WebComponents Methods

WebComponents.register Method

Registers the controls in the package as web components.

Namespace: MindFusion.Controls
File: WebComponents.js

Syntax

```
JavaScript
function register ()
```

See Also

WebComponents Members
WebComponents Class
MindFusion.Controls Namespace

ZoomControl Class

The **ZoomControl** lets users zoom and pan a target view control interactively.

Syntax

```
JavaScript
// class
ZoomControl.prototype = {}
```

Inheritance Hierarchy

MindFusion.Dom.Control
  MindFusion.Controls.ZoomControl

See Also

ZoomControl Members
MindFusion.Controls Namespace

ZoomControl Members

The ZoomControl lets users zoom and pan a target view control interactively.

The following tables list the members exposed by the **ZoomControl** type.

Public Constructors

	Name	Description
	ZoomControl	Creates an instance of **ZoomControl**.

Public Properties

	Name	Description
	activeColor	Gets or sets the color used to render depressed buttons.
	autoPostBack	Gets or sets a value indicating whether the control will post back to the server when the control's value has changed.
	backColor	Gets or sets the background color of the control.
	borderColor	Gets or sets the color of **ZoomControl** elements' borders.
	cornerRadius	Gets or sets the corner radius of rounded child elements.
	element	Returns a reference to the control's DOM element. (Inherited from Control.)
	enabled	Gets or sets whether mouse events are enabled.
	fill	Gets or sets the color used to fill the **ZoomControl** elements.
	innerColor	Gets or sets the color of plus, minus and arrow icons.
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	maxZoomFactor	Gets or sets the maximum zoom level allowed to set through this control.
	measureUnit	Gets the unit of measure used for logical coordinates. (Inherited from CanvasControl.)
	minZoomFactor	Gets or sets the minimum zoom level allowed to set through this control.
	padding	Gets or sets the padding of the control's contents.
	scrollStep	Gets or sets the scroll offset added when users click the pan arrows.
	shadowColor	Gets or sets the color of the control elements' shadow.
	showLabel	Gets or sets a value indicating whether the label that shows the current zoom level should be visible.
	snapToZoomStep	Gets or sets a value indicating whether the trackbar should snap to zoomStep values when dragged.
	target	Gets or sets the control modified by this **ZoomControl**.
	textColor	Gets or sets the color of the the label that shows the current zoom level.
	tickPosition	Gets or sets the current tick position of the trackbar.
	zoomFactor	Gets or sets the zoom factor.
	zoomStep	Gets or sets the amount by which to change zoom level when + and - buttons are clicked.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)
	clientToDoc	Transforms a point from client to document coordinates. (Inherited from CanvasControl.)
S	create	Creates and initializes a new **ZoomControl** from the specified element. This method is static and can be called without creating an instance of the class.
	docToClient	Transforms a point from document to client coordinates. (Inherited from CanvasControl.)
S	find	Returns the specified **ZoomControl** object. This member is static and can be invoked without creating an instance of the class.
S	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data. (Inherited from Control.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
S	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description
	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
	controlUnloading	Raised when the control is unloading. (Inherited from Control.)

See Also

ZoomControl Class
MindFusion.Controls Namespace

ZoomControl Constructor

Creates an instance of ZoomControl.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
function ZoomControl (element)
```

Parameters

element
HTMLCanvasElement. The Canvas DOM Element this ZoomControl is associated with.

See Also

ZoomControl Class
MindFusion.Controls Namespace

ZoomControl Methods

ZoomControl.create Method

Creates and initializes a new ZoomControl from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
function create (element)
```

Parameters

element
HTMLCanvasElement. The DOM element that the zoomControl should be attached to.

Return Value

A ZoomControl object that represents the newly created zoom control.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.find Method

Returns the specified ZoomControl object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
function find (id)
```

Parameters

id
String. The id of the ZoomControl's DOM element.

Return Value

A **ZoomControl** object with the specified id, if found, or **null** otherwise.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl Properties

ZoomControl.activeColor Property

Gets or sets the color used to render depressed buttons.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get activeColor() {}
set activeColor(value) {}
```

Property Value

String

The color used to render depressed buttons.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.autoPostBack Property

Gets or sets a value indicating whether the control will post back to the server when the control's value has changed.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get autoPostBack() {}
set autoPostBack(value) {}
```

Property Value

Boolean

true if the control should post back, or **false** otherwise.

Remarks

Used to implement auto-post back mode in MindFusion ASP.NET controls.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.backColor Property

Gets or sets the background color of the control.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get backColor() {}
set backColor(value) {}
```

Property Value

String

The color of the control background.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.borderColor Property

Gets or sets the color of ZoomControl elements' borders.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get borderColor() {}
set borderColor(value) {}
```

Property Value

String

The color of ZoomControl elements' borders.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.cornerRadius Property

Gets or sets the corner radius of rounded child elements.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get cornerRadius() {}
set cornerRadius(value) {}
```

Property Value

Number

The corner radius of rounded child elements.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.enabled Property

Gets or sets whether mouse events are enabled.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get enabled() {}
set enabled(value) {}
```

Property Value

Boolean

true to respond to mouse events, or **false** otherwise.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.fill Property

Gets or sets the color used to fill the ZoomControl elements.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get fill() {}
set fill(value) {}
```

Property Value

String

The color used to fill the ZoomControl elements.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.innerColor Property

Gets or sets the color of plus, minus and arrow icons.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get innerColor() {}
set innerColor(value) {}
```

Property Value

String

The color of plus, minus and arrow icons.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.maxZoomFactor Property

Gets or sets the maximum zoom level allowed to set through this control.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get maxZoomFactor() {}
set maxZoomFactor(value) {}
```

Property Value

Number

The maximum zoom level, specified as percentage. The default value is 200.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.minZoomFactor Property

Gets or sets the minimum zoom level allowed to set through this control.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get minZoomFactor() {}
set minZoomFactor(value) {}
```

Property Value

Number

The minimum zoom level.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.padding Property

Gets or sets the padding of the control's contents.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get padding() {}
set padding(value) {}
```

Property Value

Number

The padding of the control's contents.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.scrollStep Property

Gets or sets the scroll offset added when users click the pan arrows.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get scrollStep() {}
set scrollStep(value) {}
```

Property Value

Number

The scroll step.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.shadowColor Property

Gets or sets the color of the control elements' shadow.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get shadowColor() {}
set shadowColor(value) {}
```

Property Value

String

The color of the control elements' shadow.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.showLabel Property

Gets or sets a value indicating whether the label that shows the current zoom level should be visible.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get showLabel() {}
set showLabel(value) {}
```

Property Value

Boolean

true if the label is visible, or **false** otherwise.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.snapToZoomStep Property

Gets or sets a value indicating whether the trackbar should snap to zoomStep values when dragged.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

```
JavaScript
get snapToZoomStep() {}
set snapToZoomStep(value) {}
```

Property Value

Boolean

true if snapping is enabled, or **false** otherwise.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.target Property

Gets or sets the control modified by this ZoomControl.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get target() {}
set target(value) {}
```

Property Value

Control

The control modified by this ZoomControl.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.textColor Property

Gets or sets the color of the the label that shows the current zoom level.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get textColor() {}
set textColor(value) {}
```

Property Value

String

The color of the the label that shows the current zoom level.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.tickPosition Property

Gets or sets the current tick position of the trackbar.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get tickPosition() {}
set tickPosition(value) {}
```

Property Value

TickPosition

A member of the TickPosition enumeration.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.zoomFactor Property

Gets or sets the zoom factor.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get zoomFactor() {}
set zoomFactor(value) {}
```

Property Value

Number

The current zoom factor, specified as percentage.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl.zoomStep Property

Gets or sets the amount by which to change zoom level when + and - buttons are clicked.

Namespace: MindFusion.Controls
File: ZoomControl.js

Syntax

JavaScript

```
get zoomStep() {}
set zoomStep(value) {}
```

Property Value

Number

The zoom step. The default value is 10.

See Also

ZoomControl Members
ZoomControl Class
MindFusion.Controls Namespace

ZoomControl Events

MindFusion.Diagramming Namespace

Action Enumeration

Specifies the possible actions a user can carry out on diagram items.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
Action = {}
```

Members

	Member name	Description
	Create	Draw a new item.
	Modify	Modify an existing item.
	None	Do not perform any action.
	Split	Split a link's segment.

See Also

MindFusion.Diagramming Namespace

AddItemCommand Class

Implements creation of new diagram items.

Syntax

```
JavaScript
// class
AddItemCommand.prototype = {}
```

Remarks

The control automatically creates instances of this class when items are added either interactively or from code.

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.AddItemCommand

See Also

AddItemCommand Members
MindFusion.Diagramming Namespace
RemoveItemCommand Class
ChangeItemCommand Class

AddItemCommand Members

Implements creation of new diagram items.

The following tables list the members exposed by the AddItemCommand type.

Public Constructors

	Name	Description
	AddItemCommand	Initializes a new instance of the **AddItemCommand** class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

AddItemCommand Class
MindFusion.Diagramming Namespace

AddItemCommand Constructor

Initializes a new instance of the AddItemCommand class.

Namespace: MindFusion.Diagramming
File: AddItemCommand.js

Syntax

JavaScript

```
function AddItemCommand (diagram, item)
```

Parameters

diagram
The Diagram modified by this command.

item
A new DiagramItem that should be added to the **Diagram**.

See Also

AddItemCommand Class
MindFusion.Diagramming Namespace

AddItemCommand Methods

AddItemCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: AddItemCommand.js

Syntax

JavaScript

```
function execute ()
```

See Also

AddItemCommand Members
AddItemCommand Class
MindFusion.Diagramming Namespace

AddItemCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: AddItemCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

AddItemCommand Members
AddItemCommand Class
MindFusion.Diagramming Namespace

AddItemCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: AddItemCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

AddItemCommand Members
AddItemCommand Class
MindFusion.Diagramming Namespace

AdjustmentHandles Enumeration

Identifies the distinct selection handles of a node.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
AdjustmentHandles = {}
```

Members

	Member name	Description
	All	All selection handles are enabled and can be used to modify the node.

	Move	The center selection handle is enabled and can be used to move the node.
	None	Specifies that none of the selection handles can be used to modify the node.
	ResizeBottomCenter	The bottom-center handle is enabled and can be used to resize the node vertically.
	ResizeBottomLeft	The bottom-left handle is enabled and can be used to resize the node.
	ResizeBottomRight	The bottom-right handle is enabled and can be used to resize the node.
	ResizeMiddleLeft	The middle-left handle is enabled and can be used to resize the node horizontally.
	ResizeMiddleRight	The middle-right handle is enabled and can be used to resize the node horizontally.
	ResizeTopCenter	The top-center handle is enabled and can be used to resize the node vertically.
	ResizeTopLeft	The top-left handle is enabled and can be used to resize the node.
	ResizeTopRight	The top-right handle is enabled and can be used to resize the node.
	Rotate	The rotation handle is enabled and can be used to rotate the node.

Remarks

A combination of **AdjustmentHandles** elements can be assigned to the enabledHandles property of nodes.

See Also

MindFusion.Diagramming Namespace
HandlesStyle Enumeration

AeroEffect Class

Represents an effect that adds semitransparency to the node and emphasizes its outline by adding a shade and an inner stroke.

Syntax

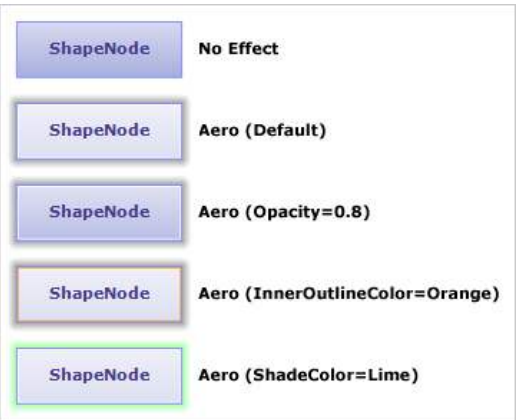
JavaScript

```
// class
AeroEffect.prototype = {}
```

Remarks

This effect can be applied to all nodes in a Diagram by adding **AeroEffect** instance to the nodeEffects collection, or to individual nodes by adding it to their effects array.

The following image illustrates the Aero effect applied to a rectangular node as well as how the different properties affect the output:



Example

The following example demonstrates how to apply aero effect to the nodes of an existing diagram.

JavaScript

```
var AeroEffect = MindFusion.Diagramming.AeroEffect;

var effect = new AeroEffect();
effect.opacity = 0.5;
effect.innerOutlineColor = "rgba(255,255,255,0.5)";
diagram.nodeEffects.push(effect);
```

Inheritance Hierarchy

MindFusion.Diagramming.NodeEffect
  MindFusion.Diagramming.AeroEffect

See Also

AeroEffect Members
MindFusion.Diagramming Namespace
Node Effects
GlassEffect Class

AeroEffect Members

Represents an effect that adds semitransparency to the node and emphasizes its outline by adding a shade and an inner stroke.

The following tables list the members exposed by the AeroEffect type.

Public Constructors

	Name	Description
	AeroEffect	Initializes a new instance of the AeroEffect class.

Public Properties

	Name	Description
	innerOutlineColor	Gets or sets the color of the inner outline.
	opacity	Gets or sets the opacity to apply to node's background.
	shadeColor	Gets or sets the color of the shade.

See Also

AeroEffect Class
MindFusion.Diagramming Namespace

AeroEffect Constructor

Initializes a new instance of the AeroEffect class.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
function AeroEffect ()
```

See Also

AeroEffect Class
MindFusion.Diagramming Namespace

AeroEffect Properties

AeroEffect.innerOutlineColor Property

Gets or sets the color of the inner outline.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
get innerOutlineColor() {}
set innerOutlineColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value. The default value is "white".

See Also

AeroEffect Members
AeroEffect Class
MindFusion.Diagramming Namespace

AeroEffect.opacity Property

Gets or sets the opacity to apply to node's background.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

```
JavaScript
get opacity() {}
set opacity(value) {}
```

Property Value

Number

A floating point value between 0 and 1.

See Also

AeroEffect Members
AeroEffect Class
MindFusion.Diagramming Namespace

AeroEffect.shadeColor Property

Gets or sets the color of the shade.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

```
JavaScript
get shadeColor() {}
set shadeColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value. The default value is "black".

See Also

AeroEffect Members
AeroEffect Class
MindFusion.Diagramming Namespace

Alignment Enumeration

Specifies the alignment of text relative to its layout rectangle.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
Alignment = {}
```

Members

	Member name	Description
	Center	The text is drawn in the center of the layout rectangle.
	Far	The text is drawn in the far corner of the layout rectangle.
	Near	The text is drawn in the near corner of the layout rectangle.

Remarks

Members of this enumeration can be assigned to textAlignment and lineAlignment properties of diagram items.

See Also

MindFusion.Diagramming Namespace

AnchorPattern Class

The **AnchorPattern** class represents a set of anchor points, which specify the exact locations where links are allowed to connect to nodes.

Syntax

JavaScript

```
// class
AnchorPattern.prototype = {}
```

Remarks

A pattern can be assigned to nodes by setting their anchorPattern property. The **AnchorPattern** class exposes several predefined patterns as static fields: decision2In2Out, decision1In3Out, leftInRightOut, topInBottomOut.

Inheritance Hierarchy

MindFusion.Diagramming.AnchorPattern

See Also

AnchorPattern Members
MindFusion.Diagramming Namespace
AnchorPoint Class

AnchorPattern Members

The AnchorPattern class represents a set of anchor points, which specify the exact locations where links are allowed to connect to nodes.

The following tables list the members exposed by the **AnchorPattern** type.

Public Constructors

	Name	Description
	AnchorPattern	Initializes a new instance of the **AnchorPattern** class.

Public Fields

	Name	Description
	decision1In3Out	Gets the predefined anchor pattern decision1In3Out.
	decision2In2Out	Gets the predefined anchor pattern decision2In2Out.
	leftInRightOut	Gets the predefined anchor pattern leftInRightOut.
	topInBottomOut	Gets the predefined anchor pattern topInBottomOut.

Public Properties

	Name	Description
	id	Gets the unique identifier of this AnchorPattern.
	points	Gets or sets the anchor points in this pattern.

Public Methods

	Name	Description
	fromId	Returns a reference to the anchor pattern with the specified string identifier.

See Also

AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern Fields

AnchorPattern.decision1In3Out Field

Gets the predefined anchor pattern decision1In3Out.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

JavaScript

```
var decision1In3Out
```

Remarks

Incoming links connect to the top vertex; outgoing links can start from the other vertices.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern.decision2In2Out Field

Gets the predefined anchor pattern decision2In2Out.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

JavaScript

```
var decision2In2Out
```

Remarks

Incoming links can connect with the top and left vertices; outgoing links can start from the right and bottom vertices.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern.leftInRightOut Field

Gets the predefined anchor pattern leftInRightOut.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

JavaScript

```
var leftInRightOut
```

Remarks

The middle point of a node's bounding rectangle left side is anchor for incoming links; that on the right side - for outgoing.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern.topInBottomOut Field

Gets the predefined anchor pattern topInBottomOut.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

JavaScript

```
var topInBottomOut
```

Remarks

The middle point of a node's bounding rectangle top side is anchor for incoming links; that on the bottom - for outgoing.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern Constructor

Initializes a new instance of the AnchorPattern class.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

JavaScript

```
function AnchorPattern (points, [patternId])
```

Parameters

points
Contains the initial set of AnchorPoint instances for the new pattern.

patternId
Optional.
String. An unique identifier for this anchor pattern.

See Also

AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern Methods

AnchorPattern.fromId Method

Returns a reference to the anchor pattern with the specified string identifier.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

JavaScript

```
function fromId (id)
```

Parameters

id
A string identifying an AnchorPattern instance.

Return Value

AnchorPattern. The AnchorPattern object whose Id is equal to the id parameter.

Remarks

The id argument should match an identifier passed to the constructor of an existing AnchorPattern object.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern Properties

AnchorPattern.id Property

Gets the unique identifier of this AnchorPattern.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

```
JavaScript
get id() {}
```

Property Value

String

A string containing the AnchorPattern id.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPattern.points Property

Gets or sets the anchor points in this pattern.

Namespace: MindFusion.Diagramming
File: AnchorPattern.js

Syntax

```
JavaScript
get points() {}
set points(value) {}
```

Property Value

Array

An array of AnchorPoint objects.

See Also

AnchorPattern Members
AnchorPattern Class
MindFusion.Diagramming Namespace

AnchorPoint Class

The **AnchorPoint** class represents a location in a node to which links are allowed to connect.

Syntax

```
JavaScript
// class
AnchorPoint.prototype = {}
```

Remarks

Anchor points can be assigned to nodes only as constituent elements of an AnchorPattern object.

An anchor point can accept incoming or outgoing links, or both, as specified via the allowIncoming and allowOutgoing properties. The position of the anchor point inside a node's bounding rectangle is set through the x and y properties, expressed as percentage of rectangle's width and height.

How anchor points are marked on the display can be set using markStyle and color properties.

Inheritance Hierarchy

MindFusion.Diagramming.AnchorPoint

See Also

AnchorPoint Members
MindFusion.Diagramming Namespace
AnchorPattern Class

AnchorPoint Members

The AnchorPoint class represents a location in a node to which links are allowed to connect.

The following tables list the members exposed by the **AnchorPoint** type.

Public Constructors

	Name	Description
	AnchorPoint	Initializes a new instance of the AnchorPoint class.

Public Properties

	Name	Description
	allowIncoming	Gets or sets a value indicating whether the anchor point accepts incoming connections.
	allowOutgoing	Gets or sets a value indicating whether the anchor point accepts outgoing connections.
	color	Gets or sets the color of the anchor point mark.
	column	Gets or sets the column index of a cell with which the anchor point is associated.
	markStyle	Gets or sets the appearance of the anchor point mark.
	size	Gets or sets the size of the anchor point mark.
	tag	Gets or sets custom data tag associated with this anchor point.
	toolTip	Gets or sets tooltip text that should be displayed when the mouse hovers over an anchor point mark.
	x	Gets or sets the horizontal position of the anchor point expressed as percentage of a node's width.
	y	Gets or sets the vertical position of the anchor point expressed as percentage of a node's height.

See Also

AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint Constructor

Initializes a new instance of the AnchorPoint class.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

JavaScript

```
function AnchorPoint (x, y, [allowIncoming, [allowOutgoing, [style, [color, [size]]]]])
```

Parameters

x
Number. The horizontal position of the anchor expressed as percent of the width of a node.

y
Number. The vertical position of the anchor expressed as percent of the height of a node.

allowIncoming
Optional.
Boolean. Specifies whether incoming links can connect to this anchor.

allowOutgoing
Optional.
Boolean. Specifies whether outgoing links can connect to this anchor.

style
Optional.
MarkStyle. Defines how the anchor point is marked visually.

color
Optional.
String. Defines the color with which the anchor mark is painted.

size
Optional.
Number. Defines the size of the anchor mark.

See Also

AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint Properties

AnchorPoint.allowIncoming Property

Gets or sets a value indicating whether the anchor point accepts incoming connections.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

```
JavaScript
get allowIncoming() {}
set allowIncoming(value) {}
```

Property Value

Boolean

true to accept incoming connections, or **false** otherwise.

Remarks

If no incoming anchor points are defined for a node, link end points can be placed anywhere inside that node.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace
allowOutgoing Property
validateAnchorPoint Event

AnchorPoint.allowOutgoing Property

Gets or sets a value indicating whether the anchor point accepts outgoing connections.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

```
JavaScript
get allowOutgoing() {}
set allowOutgoing(value) {}
```

Property Value

Boolean

true to accept outgoing connections, or **false** otherwise.

Remarks

If no outgoing anchor points are defined for a node, link start points can be placed anywhere inside that node.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace
allowIncoming Property
validateAnchorPoint Event

AnchorPoint.color Property

Gets or sets the color of the anchor point mark.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

JavaScript

```
get color() {}
set color(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

Remarks

The anchor mark shape specified via MarkStyle is painted with this color.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint.column Property

Gets or sets the column index of a cell with which the anchor point is associated.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

JavaScript

```
get column() {}
set column(value) {}
```

Property Value

Number

The column index of a cell with which the anchor point is associated.

Remarks

If **column** is set to -1, the point relative coordinates specify percentage of row's extents. Otherwise the point coordinates specify percentage of cell's extents.

See Also

AnchorPoint Members
AnchorPoint Class

AnchorPoint.markStyle Property

Gets or sets the appearance of the anchor point mark.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

```
JavaScript
get markStyle() {}
set markStyle(value) {}
```

Property Value

MarkStyle

A member of the MarkStyle enumeration.

Remarks

Specifies how anchor points are marked visually on the screen.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint.size Property

Gets or sets the size of the anchor point mark.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

```
JavaScript
get size() {}
set size(value) {}
```

Property Value

Number

A number specifying the size.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint.tag Property

Gets or sets custom data tag associated with this anchor point.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

JavaScript

```
get tag() {}
set tag(value) {}
```

Property Value

Object

An Object containing custom data.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint.toolTip Property

Gets or sets tooltip text that should be displayed when the mouse hovers over an anchor point mark.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

JavaScript

```
get toolTip() {}
set toolTip(value) {}
```

Property Value

String

A string containing the tooltip text.

Remarks

Tooltip text is displayed in a small popup window near the cursor when the mouse rests for a while over an anchor point.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint.x Property

Gets or sets the horizontal position of the anchor point expressed as percentage of a node's width.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

```
JavaScript
get x() {}
set x(value) {}
```

Property Value

Number

The horizontal position of the anchor point expressed as percentage of a node's width.

Remarks

Keep the property value within 0-100 range, or otherwise it will lead to unexpected behavior.

See Also

AnchorPoint Members
AnchorPoint Class
MindFusion.Diagramming Namespace

AnchorPoint.y Property

Gets or sets the vertical position of the anchor point expressed as percentage of a node's height.

Namespace: MindFusion.Diagramming
File: AnchorPoint.js

Syntax

```
JavaScript
get y() {}
set y(value) {}
```

Property Value

Number

The vertical position of the anchor point expressed as percentage of a node's height.

Remarks

Keep the property value within 0-100 range, or otherwise it will lead to unexpected behavior.

See Also

ArrowHeads Class

Provides acccess to predefined arrowhead shapes.

Syntax

JavaScript

```
// class
ArrowHeads.prototype = {}
```

Remarks

This class lists the predefined shapes designed as arrowheads, which can be assigned to headShape or baseShape property of links.

Inheritance Hierarchy

MindFusion.Diagramming.ArrowHeads

See Also

ArrowHeads Members
MindFusion.Diagramming Namespace

ArrowHeads Members

Provides acccess to predefined arrowhead shapes.

The following tables list the members exposed by the ArrowHeads type.

Public Methods

	Name	Description
	Arrow	Gets the Arrow predefined shape.
	BackSlash	Gets the BackSlash predefined shape.
	BowArrow	Gets the BowArrow predefined shape.
	Circle	Gets the Circle predefined shape.
	DefaultFlow	Gets the DefaultFlow predefined shape.
	DoubleArrow	Gets the DoubleArrow predefined shape.
	None	Do not display an arrow shape.
	Pentagon	Gets the PentagonArrow predefined shape.
	PointerArrow	Gets the PointerArrow predefined shape.

	Quill	Gets the Quill predefined shape.
	Reversed	Gets the Reversed predefined shape.
	RevTriangle	Gets the RevTriangle predefined shape.
	RevWithCirc	Gets the RevWithCirc predefined shape.
	RevWithLine	Gets the RevWithLine predefined shape.
	Rhombus	Gets the Rhombus predefined shape.
	Slash	Gets the Slash predefined shape.
	Tetragon	Gets the Tetragon predefined shape.
	Triangle	Gets the Triangle predefined shape.

See Also

ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads Methods

ArrowHeads.Arrow Method

Gets the Arrow predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

JavaScript

```
function Arrow ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.BackSlash Method

Gets the BackSlash predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

JavaScript

```
function BackSlash ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.BowArrow Method

Gets the BowArrow predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

JavaScript

```
function BowArrow ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Circle Method

Gets the Circle predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

JavaScript

```
function Circle ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.DefaultFlow Method

Gets the DefaultFlow predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function DefaultFlow ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.DoubleArrow Method

Gets the DoubleArrow predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function DoubleArrow ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.None Method

Do not display an arrow shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function None ()
```

Return Value

Shape. A **null** reference.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Pentagon Method

Gets the PentagonArrow predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function Pentagon ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.PointerArrow Method

Gets the PointerArrow predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function PointerArrow ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Quill Method

Gets the Quill predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function Quill ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Reversed Method

Gets the Reversed predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function Reversed ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.RevTriangle Method

Gets the RevTriangle predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function RevTriangle ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.RevWithCirc Method

Gets the RevWithCirc predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function RevWithCirc ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.RevWithLine Method

Gets the RevWithLine predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function RevWithLine ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Rhombus Method

Gets the Rhombus predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

JavaScript

```
function Rhombus ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Slash Method

Gets the Slash predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

JavaScript

```
function Slash ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Tetragon Method

Gets the Tetragon predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function Tetragon ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

ArrowHeads.Triangle Method

Gets the Triangle predefined shape.

Namespace: MindFusion.Diagramming
File: ArrowHeads.js

Syntax

```
JavaScript
function Triangle ()
```

Return Value

Shape. An instance of the Shape class representing the predefined shape.

See Also

ArrowHeads Members
ArrowHeads Class
MindFusion.Diagramming Namespace

AutoResize Enumeration

Defines in which directions automatic resizing can enlarge the diagram scrollable area.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
```

```
// enum
AutoResize = {}
```

Members

	Member name	Description
	AllDirections	Enlarge the diagram area in any direction if an item is moved outside the boundaries.
	None	Do not enlarge the diagram area automatically.
	RightAndDown	Enlarge the diagram area to the right and down if an item is moved outside the boundaries in these directions.

Remarks

Members of this enumeration can be assigned to the autoResize property of Diagram.

See Also

MindFusion.Diagramming Namespace

Behavior Enumeration

Defines values that specify how the component responds to actions performed by the user.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
Behavior = {}
```

Members

	Member name	Description
	Custom	Drawing with the mouse creates instances of the type assigned to CustomNodeType.
	DoNothing	The control ignores users actions with the mouse, but raises the appropriate mouse events. This mode allows applications to implement their own mouse-drawing behavior, disabling the default response to users actions.
	DrawContainers	Drawing with the mouse creates ContainerNode instances.
	DrawControls	Drawing with the mouse creates ControlNode instances.
	DrawFreeForms	Drawing with the mouse creates FreeFormNode instances.
	DrawFreeShapes	Points from user input are shown using a **FreeFormNode** instance. Once the user finishes drawing, the

		FreeFormNode is replaced by a ShapeNode with a matching shape from Diagram.FreeFormTargets list.
	DrawLinks	Drawing started over a node creates a DiagramLink. Otherwise a lasso rectangle is displayed allowing the selection of items.
	DrawShapes	Drawing with the mouse creates **ShapeNode** instances.
	DrawSvgNodes	Drawing with the mouse creates SvgNode instances.
	DrawTables	Drawing with the mouse creates TableNode instances.
	DrawTreeViews	Drawing with the mouse creates TreeViewNode instances.
	LinkContainers	Drawing over empty document area creates a **ContainerNode** instance. Drawing started over a node creates a **DiagramLink**.
	LinkControls	Drawing over empty document area creates a **ControlNode** instance. Drawing started over a node creates a **DiagramLink**.
	LinkFreeForms	Drawing over empty document area creates a **FreeFormNode** instance. Drawing started over a node creates a **DiagramLink**.
	LinkFreeShapes	Points from user input are shown using a **FreeFormNode** instance. Once the user finishes drawing, the **FreeFormNode** is replaced by a **ShapeNode** with a matching shape from Diagram.FreeFormTargets list. Drawing started over a node creates a **DiagramLink**.
	LinkShapes	A mode that is suitable for creating flowcharts and process diagrams. Drawing over empty document area creates a **ShapeNode** instance. Drawing started over a node creates a **DiagramLink**.
	LinkSvgNodes	Drawing over empty document area creates an **SvgNode** instance. Drawing started over a node creates a **DiagramLink**.
	LinkTables	A mode that is suitable for creating entity-relationship diagrams. Drawing over empty document area creates a **TableNode** instance. Drawing started over a node creates a **DiagramLink**.
	LinkTreeViews	Drawing over empty document area creates a **TreeViewNode** instance. Drawing started over a node creates a **DiagramLink**.
	Magnify	Dragging the mouse magnifies the view. Modifying them or drawing new ones is disabled.
	Modify	Objects can be selected and modified. New objects cannot be created.
	MoveLabels	Dragging with the mouse moves an ItemLabel at mouse pointer's location.
	MoveNodes	Dragging with the mouse moves the node under mouse pointer.
	Pan	Dragging with depressed mouse button pans the view.

	PanAndModify	Items can be selected and modified. New items cannot be created, and dragging without grabbing an adjustment handle pans the view.
	SelectOnly	Allow only selection of existing items. Modifying them or drawing new ones is disabled.
	Zoom	Dragging with the mouse draws a lasso tool that zooms the DiagramView to selected rectangular area.

Remarks

Members of this enumeration can be assigned to the behavior property of **DiagramView**.

See Also

MindFusion.Diagramming Namespace

BehaviorBase Class

An abstract base class for types that define how the control reacts to users' actions with a mouse or other pointing devices.

Syntax

```
JavaScript
// class
BehaviorBase.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
MindFusion.Diagramming.DoNothingBehavior
MindFusion.Diagramming.DrawLinksBehavior
MindFusion.Diagramming.DrawNodesBehavior
MindFusion.Diagramming.LinkNodesBehavior
MindFusion.Diagramming.MagnifyBehavior
MindFusion.Diagramming.ModifyBehavior
MindFusion.Diagramming.MoveNodesBehavior
MindFusion.Diagramming.PanBehavior
MindFusion.Diagramming.SelectOnlyBehavior

See Also

BehaviorBase Members
MindFusion.Diagramming Namespace

BehaviorBase Members

An abstract base class for types that define how the control reacts to users' actions with a mouse or other pointing devices.

The following tables list the members exposed by the BehaviorBase type.

Public Constructors

	Name	Description
	BehaviorBase	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object.

Public Methods

	Name	Description
	createController	Creates a controller appropriate for current context and pointer position.

See Also

BehaviorBase Class
MindFusion.Diagramming Namespace

BehaviorBase Constructor

Overload List

Name	Description
BehaviorBase	Initializes a new instance of the BehaviorBase class.
BehaviorBase	Initializes a new instance of the BehaviorBase class.

See Also

BehaviorBase Class
MindFusion.Diagramming Namespace

BehaviorBase Constructor

Initializes a new instance of the BehaviorBase class.

Namespace: MindFusion.Diagramming
File: BehaviorBase.js

Syntax

JavaScript
`function BehaviorBase (parent)`

Parameters

parent
The Diagram this behavior will be associated with.

See Also

BehaviorBase Constructor Overload List
BehaviorBase Class
MindFusion.Diagramming Namespace

BehaviorBase Constructor

Initializes a new instance of the BehaviorBase class.

Namespace: MindFusion.Diagramming
File: BehaviorBase.js

Syntax

JavaScript

```
function BehaviorBase (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

BehaviorBase Constructor Overload List
BehaviorBase Class
MindFusion.Diagramming Namespace

BehaviorBase Methods

BehaviorBase.createController Method

Creates a controller appropriate for current context and pointer position.

Namespace: MindFusion.Diagramming
File: BehaviorBase.js

Syntax

JavaScript

```
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

BehaviorBase Members
BehaviorBase Class
MindFusion.Diagramming Namespace

BehaviorBase Properties

BehaviorBase.diagram Property

Gets the target diagram of this behavior object.

Namespace: MindFusion.Diagramming
File: BehaviorBase.js

Syntax

```
JavaScript
get diagram() {}
set diagram(value) {}
```

Property Value

Diagram

An instance of the Diagram class.

See Also

BehaviorBase Members
BehaviorBase Class
MindFusion.Diagramming Namespace

Border Class

Represents a panel that draws a border around its child components.

Syntax

```
JavaScript
// class
Border.prototype = {}
```

Remarks

Border objects can be used as components of CompositeNode.

Example

This part of a custom **CompositeNode**'s JSON template shows Text component inside a **Border**:

```
JavaScript
```

```
{
    component: "Border",
    gridColumn: 0,
    name: "SurnameBorder",
    pen: "yellow",
    strokeThickness: 2,
    brush: "green",
    margin: "2, 0, 0, 0",
    children:
    [
        {
            component: "Text",
            name: "Surname",
            autoProperty: true,
            text: "",
            font: "Arial"
        }
    ]
},
```

Inheritance Hierarchy

MindFusion.Diagramming.Panel
MindFusion.Diagramming.SimplePanel
MindFusion.Diagramming.Border

See Also

Border Members
MindFusion.Diagramming Namespace

Border Members

Represents a panel that draws a border around its child components.

The following tables list the members exposed by the Border type.

Public Constructors

	Name	Description
	Border	Initializes a new instance of the Border class.

Public Fields

	Name	Description
	brush	The panel background color.
	pen	The border color.
	strokeThickness	The border thickness.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)

	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	getComponents	Gets the child components of this panel. (Inherited from Panel.)

See Also

Border Class
MindFusion.Diagramming Namespace

Border Fields

Border.brush Field

The panel background color.

Namespace: MindFusion.Diagramming
File: Border.js

Syntax

JavaScript

```
var brush
```

See Also

Border Members
Border Class
MindFusion.Diagramming Namespace

Border.pen Field

The border color.

Namespace: MindFusion.Diagramming
File: Border.js

Syntax

JavaScript

```
var pen
```

See Also

Border Members
Border Class
MindFusion.Diagramming Namespace

Border.strokeThickness Field

The border thickness.

Namespace: MindFusion.Diagramming
File: Border.js

Syntax

JavaScript

```
var strokeThickness
```

See Also

Border Members
Border Class
MindFusion.Diagramming Namespace

Border Constructor

Initializes a new instance of the Border class.

Namespace: MindFusion.Diagramming
File: Border.js

Syntax

JavaScript

```
function Border ()
```

See Also

Border Class
MindFusion.Diagramming Namespace

Builder Class

Provides support for fluent API.

Syntax

JavaScript

```
```

```
// class
Builder.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Builder

See Also

Builder Members
MindFusion.Diagramming Namespace

Builder Members

Provides support for fluent API.

The following tables list the members exposed by the Builder type.

Public Constructors

	Name	Description
	Builder	Initializes a new instance of the Builder class.

See Also

Builder Class
MindFusion.Diagramming Namespace

Builder Constructor

Initializes a new instance of the Builder class.

Namespace: MindFusion.Diagramming
File: Builders.js

Syntax

JavaScript
`function Builder ()`

See Also

Builder Class
MindFusion.Diagramming Namespace

Button Class

Represents a button component for CompositeNodes.

Syntax

JavaScript

```
// class
Button.prototype = {}
```

Remarks

Buttons respond to user clicks by calling the JavaScript function whose name is assigned to their **clickHandler** attribute. A button can show a text label assigned to the **text** attribute, or an image whose URL is assigned to **imageLocation**.

Example

The following example adds a button to CompositeNode template:

JavaScript

```
{
    component: "Button",
    brush: "#ccc",
    text: "Delete",
    width: 30,
    cornerRadius: 3.5,
    clickHandler: "onDeleteClick"
}
```

Inheritance Hierarchy

MindFusion.Drawing.Component
  MindFusion.Diagramming.Button

See Also

Button Members
MindFusion.Diagramming Namespace

Button Members

Represents a button component for CompositeNodes.

The following tables list the members exposed by the Button type.

Public Constructors

	Name	Description
	Button	Initializes a new instance of the Button class.

Public Fields

	Name	Description
	brush	The button background color.
	font	A MindFusion.Drawing.Font instance specifying button's font.
	height	The desired height of the button.

	pen	The border color.
	strokeThickness	The border thickness.
	textColor	The text color.
	width	The desired width of the button.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	text	Gets or sets the text caption displayed in this button.
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

See Also

Button Class
MindFusion.Diagramming Namespace

Button Fields

Button.brush Field

The button background color.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

JavaScript
`var brush`

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button.font Field

A MindFusion.Drawing.Font instance specifying button's font.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

```
JavaScript
var font
```

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button.height Field

The desired height of the button.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

```
JavaScript
var height
```

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button.pen Field

The border color.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

```
JavaScript
var pen
```

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button.strokeThickness Field

The border thickness.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

```
JavaScript
var strokeThickness
```

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button.textColor Field

The text color.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

```
JavaScript
var textColor
```

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button.width Field

The desired width of the button.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

```
JavaScript
var width
```

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

Button Constructor

Initializes a new instance of the Button class.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

JavaScript

```
function Button ()
```

See Also

Button Class
MindFusion.Diagramming Namespace

Button Properties

Button.text Property

Gets or sets the text caption displayed in this button.

Namespace: MindFusion.Diagramming
File: Button.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

A string containing button's caption.

See Also

Button Members
Button Class
MindFusion.Diagramming Namespace

CaptionedNode Class

CaptionedNode is an abstract base class from which the TableNode, ContainerNode and TreeViewNode classes derive.

Syntax

JavaScript

```
// class
CaptionedNode.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.CaptionedNode
      MindFusion.Diagramming.ContainerNode
      MindFusion.Diagramming.TableNode
      MindFusion.Diagramming.TreeViewNode

See Also

CaptionedNode Members
MindFusion.Diagramming Namespace

CaptionedNode Members

CaptionedNode is an abstract base class from which the TableNode, ContainerNode and TreeViewNode classes derive.

The following tables list the members exposed by the CaptionedNode type.

Public Constructors

	Name	Description
	CaptionedNode	Initializes a new instance of the CaptionedNode class.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)

	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	captionBackBrush	Gets or sets an object specifying how to fill the caption bar.
	captionFont	Gets or sets the font used to render the node's caption text.
	captionHeight	Gets or sets the height of the node's caption bar.
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent **ContainerNode** containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	dividerStroke	Gets or sets the color used to stroke the divider line between node's caption bar and content area.
	dividerStrokeDashArray	Gets or sets an array of values specifying the dash pattern of the node's divider line.
	dividerStrokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern.
	dividerStrokeDashStyle	Gets or sets the line dash pattern applied when stroking node's divider line.
	dividerStrokeThickness	Gets or sets the width of the divider line between node's caption bar and content area.
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveDividerStrokeThickness	Gets the effective line thickness that should be applied when stroking this node's divider line.
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)

	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)

	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the shape of the node's outline.
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)

	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)

	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramItem.clone override.
	containsPoint	**DiagramItem.containsPoint** override. (Inherited from DiagramNode.)
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramItem.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets the bounding rectangle of this node. (Inherited from DiagramNode.)
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding **rectangle** of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramItem.LoadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)

	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	restoreState	DiagramItem.restoreState override.
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramItem.saveState override.
	saveToXml	Saves the item content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramItem.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

See Also

CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode Constructor

Initializes a new instance of the CaptionedNode class.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
function CaptionedNode ([parent])
```

Parameters

parent
Optional.
Diagram. A Diagram instance whose default node attributes are copied to this node.

See Also

CaptionedNode Methods

CaptionedNode.clone Method

DiagramItem.clone override.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.fromJson Method

DiagramItem.fromJson override.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.loadFromXml Method

DiagramItem.LoadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
```

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.restoreState Method

DiagramItem.restoreState override.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.saveState Method

DiagramItem.saveState override.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
function saveState ()
```

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.saveToXml Method

Saves the item content into an XML element.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.toJson Method

DiagramItem.toJson override.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode Properties

CaptionedNode.captionBackBrush Property

Gets or sets an object specifying how to fill the caption bar.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
get captionBackBrush() {}
set captionBackBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace
dividerStroke Property

CaptionedNode.captionFont Property

Gets or sets the font used to render the node's caption text.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
get captionFont() {}
set captionFont(value) {}
```

Property Value

Font

An object specifying the font of the caption bar.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.captionHeight Property

Gets or sets the height of the node's caption bar.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
```

```
get captionHeight() {}
set captionHeight(value) {}
```

Property Value

Number

A number specifying the height of the caption bar. The default value is 6.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.dividerStroke Property

Gets or sets the color used to stroke the divider line between node's caption bar and content area.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
get dividerStroke() {}
set dividerStroke(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

Remarks

If **dividerStroke** is not set, the divider line is drawn using the color assigned to stroke property.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace
captionBackBrush Property

CaptionedNode.dividerStrokeDashArray Property

Gets or sets an array of values specifying the dash pattern of the node's divider line.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
get dividerStrokeDashArray() {}
set dividerStrokeDashArray(value) {}
```

Property Value

Array

The pattern.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.dividerStrokeDashOffset Property

Gets or sets the the distance between the line start and the beginning of the dash pattern.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
get dividerStrokeDashOffset() {}
set dividerStrokeDashOffset(value) {}
```

Property Value

Number

The offset.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.dividerStrokeDashStyle Property

Gets or sets the line dash pattern applied when stroking node's divider line.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
get dividerStrokeDashStyle() {}
set dividerStrokeDashStyle(value) {}
```

Property Value

DashStyle

A member of the DashStyle enumeration.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.dividerStrokeThickness Property

Gets or sets the width of the divider line between node's caption bar and content area.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
get dividerStrokeThickness() {}
set dividerStrokeThickness(value) {}
```

Property Value

Number

A number specifying the stroke thickness.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

CaptionedNode.effectiveDividerStrokeThickness Property

Gets the effective line thickness that should be applied when stroking this node's divider line.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

JavaScript

```
get effectiveDividerStrokeThickness() {}
set effectiveDividerStrokeThickness(value) {}
```

Property Value

Number

A number specifying the item's divider line stroke thickness.

See Also

CaptionedNode Members
CaptionedNode Class

MindFusion.Diagramming Namespace

CaptionedNode.shape Property

Gets or sets the shape of the node's outline.

Namespace: MindFusion.Diagramming
File: CaptionedNode.js

Syntax

```
JavaScript
get shape() {}
set shape(value) {}
```

Property Value

SimpleShape

A member of the SimpleShape enumeration.

See Also

CaptionedNode Members
CaptionedNode Class
MindFusion.Diagramming Namespace

Cell Class

Represents a cell of a TableNode.

Syntax

```
JavaScript
// class
Cell.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Cell

See Also

Cell Members
MindFusion.Diagramming Namespace

Cell Members

Represents a cell of a TableNode.

The following tables list the members exposed by the Cell type.

Public Constructors

	Name	Description
	Cell	Initializes a new instance of the **Cell** class.

Public Properties

	Name	Description
	brush	Gets or sets the brush used to paint the cell's background.
	columnSpan	Gets or sets the number of columns spanned by this cell.
	font	Gets or sets the font used to render cell's text.
	image	Gets or sets the image displayed by this cell.
	imageAlign	Gets or sets the alignment of cell's image.
	imageContent	Gets or sets a Base64-encoded image to display inside this cell.
	imageLocation	Gets or sets URL of the image to display inside this cell.
	imagePadding	Gets or sets the padding distance between cell's borders and image.
	lineAlignment	Gets or sets the vertical alignment of cell's text.
	rowSpan	Gets or sets the number of rows spanned by this cell.
	tag	Gets or sets custom data tag associated with this cell.
	text	Gets or sets the text label displayed by this cell.
	textAlignment	Gets or sets the horizontal alignment of cell's text.
	textColor	Gets or sets the cell's text color.
	textPadding	Gets or sets the padding distance between cell's borders and its text.
	tooltip	Gets or sets the tooltip text that should be displayed when the mouse hovers over this cell.

Public Methods

	Name	Description
	loadFromXml	Loads the cell's content from an XML element.
	saveToXml	Saves the cell's content into an XML element.

See Also

Cell Class
MindFusion.Diagramming Namespace

Cell Constructor

Initializes a new instance of the Cell class.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
function Cell ()
```

See Also

Cell Class
MindFusion.Diagramming Namespace

Cell Methods

Cell.loadFromXml Method

Loads the cell's content from an XML element.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the cell's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.saveToXml Method

Saves the cell's content into an XML element.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the cell's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell Properties

Cell.brush Property

Gets or sets the brush used to paint the cell's background.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.columnSpan Property

Gets or sets the number of columns spanned by this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get columnSpan() {}
set columnSpan(value) {}
```

Property Value

Number

An integer value specifying the number of spanned columns.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.font Property

Gets or sets the font used to render cell's text.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get font() {}
set font(value) {}
```

Property Value

Font

An instance of the Font class.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.image Property

Gets or sets the image displayed by this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get image() {}
set image(value) {}
```

Property Value

HTMLImageElement

An image object to render as cell's content.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.imageAlign Property

Gets or sets the alignment of cell's image.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get imageAlign() {}
set imageAlign(value) {}
```

Property Value

ImageAlign

A member of the ImageAlign enumeration.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.imageContent Property

Gets or sets a Base64-encoded image to display inside this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get imageContent() {}
set imageContent(value) {}
```

Property Value

String

A Base64-encoded representation of an image.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.imageLocation Property

Gets or sets URL of the image to display inside this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get imageLocation() {}
set imageLocation(value) {}
```

Property Value

String

A URL string that identifies the image to be loaded.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.imagePadding Property

Gets or sets the padding distance between cell's borders and image.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get imagePadding() {}
set imagePadding(value) {}
```

Property Value

Thickness

A Thickness instance specifying image padding.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.lineAlignment Property

Gets or sets the vertical alignment of cell's text.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get lineAlignment() {}
set lineAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.rowSpan Property

Gets or sets the number of rows spanned by this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get rowSpan() {}
set rowSpan(value) {}
```

Property Value

Number

An integer value specifying the number of spanned rows.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.tag Property

Gets or sets custom data tag associated with this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get tag() {}
set tag(value) {}
```

Property Value

Object

An Object containing custom data.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.text Property

Gets or sets the text label displayed by this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

A string containing the cell's text. The default value is empty string "".

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.textAlignment Property

Gets or sets the horizontal alignment of cell's text.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get textAlignment() {}
set textAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.textColor Property

Gets or sets the cell's text color.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string value identifying the text color.

Remarks

If this property is not set, cell's text is painted using table's cellTextColor or textColor properties.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

Cell.textPadding Property

Gets or sets the padding distance between cell's borders and its text.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

```
JavaScript
get textPadding() {}
set textPadding(value) {}
```

Property Value

Thickness

A Thickness instance specifying text padding.

See Also

Cell Members
Cell Class

MindFusion.Diagramming Namespace

Cell.tooltip Property

Gets or sets the tooltip text that should be displayed when the mouse hovers over this cell.

Namespace: MindFusion.Diagramming
File: Cell.js

Syntax

JavaScript

```
get tooltip() {}
set tooltip(value) {}
```

Property Value

String

A string containing the tooltip text.

See Also

Cell Members
Cell Class
MindFusion.Diagramming Namespace

CellEventArgs Class

Contains the arguments passed to handlers of table cell -related events.

Syntax

JavaScript

```
// class
CellEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
MindFusion.Diagramming.CellEventArgs

See Also

CellEventArgs Members
MindFusion.Diagramming Namespace
Events Class

CellEventArgs Members

Contains the arguments passed to handlers of table cell -related events.

The following tables list the members exposed by the CellEventArgs type.

Public Constructors

	Name	Description
	CellEventArgs	Initializes a new instance of the CellEventArgs class.

Public Properties

	Name	Description
	cell	Gets the cell related to the event.
	column	Gets the column index of the cell related to the event.
	context	Gets the canvas rendering context.
	mouseButton	Gets which mouse button has been pressed.
	newText	Gets the text that has just been entered by the user.
	node	Gets the node related to the event.
	oldText	Gets the text that had been displayed in the node before the user edited it.
	row	Gets the row index of the cell related to the event.

Public Methods

	Name	Description
	cancelDrag	Cancels the resize of the node's row or column.

See Also

CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs Constructor

Initializes a new instance of the CellEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript
`function CellEventArgs ()`

See Also

CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs Methods

CellEventArgs.cancelDrag Method

Cancels the resize of the node's row or column.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
function cancelDrag ()
```

Remarks

Calling this method immediately stops the operation.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace
allowResizeRows Property
allowResizeColumns Property

CellEventArgs Properties

CellEventArgs.cell Property

Gets the cell related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get cell() {}
```

Property Value

Cell

A Cell instance.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.column Property

Gets the column index of the cell related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get column() {}
set column(value) {}
```

Property Value

Number

The column index.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.context Property

Gets the canvas rendering context.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get context() {}
set context(value) {}
```

Property Value

CanvasRenderingContext2D

A rendering context used to draw on the Canvas element.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.mouseButton Property

Gets which mouse button has been pressed.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get mouseButton() {}
set mouseButton(value) {}
```

Property Value

Number

An integer mouse button id.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.newText Property

Gets the text that has just been entered by the user.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get newText() {}
set newText(value) {}
```

Property Value

String

A string containing newly entered text.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.node Property

Gets the node related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get node() {}
set node(value) {}
```

Property Value

TableNode

A TableNode instance.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.oldText Property

Gets the text that had been displayed in the node before the user edited it.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get oldText() {}
set oldText(value) {}
```

Property Value

String

A string containing item's original text.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellEventArgs.row Property

Gets the row index of the cell related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get row() {}
set row(value) {}
```

Property Value

Number

The row index.

See Also

CellEventArgs Members
CellEventArgs Class
MindFusion.Diagramming Namespace

CellFrameStyle Enumeration

Specifies the style of the cells' frame lines.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
CellFrameStyle = {}
```

Members

	Member name	Description
	None	Table cells have no border.
	Simple	The border of table cells is simple line.
	System3D	The border has 3D look.

Remarks

Members of this enumeration can be assigned to cellFrameStyle property of TableNode.

See Also

MindFusion.Diagramming Namespace

ChangeItemCommand Class

Handles undo / redo of property changes.

Syntax

JavaScript

```
// class
ChangeItemCommand.prototype = {}
```

Remarks

The ChangeItemCommand constructor records the state of an item, and its undo and redo methods restore the item to either its old or new state.

Example

See executeCommand for an example showing how item's state is saved for later undo / redo.

Inheritance Hierarchy

MindFusion.Diagramming.Command
MindFusion.Diagramming.ChangeItemCommand

See Also

ChangeItemCommand Members
MindFusion.Diagramming Namespace

ChangeItemCommand Members

Handles undo / redo of property changes.

The following tables list the members exposed by the ChangeItemCommand type.

Public Constructors

	Name	Description
	ChangeItemCommand	Initializes a new instance of the **ChangeItemCommand** class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

ChangeItemCommand Class
MindFusion.Diagramming Namespace

ChangeItemCommand Constructor

Initializes a new instance of the ChangeItemCommand class.

Namespace: MindFusion.Diagramming
File: ChangeItemCommand.js

Syntax

JavaScript

```
function ChangeItemCommand (diagram, item)
```

Parameters

diagram

The Diagram modified by this command.

item
The DiagramItem that will be modified.

See Also

ChangeItemCommand Class
MindFusion.Diagramming Namespace

ChangeItemCommand Methods

ChangeItemCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: ChangeItemCommand.js

Syntax

JavaScript

```
function execute ()
```

See Also

ChangeItemCommand Members
ChangeItemCommand Class
MindFusion.Diagramming Namespace

ChangeItemCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: ChangeItemCommand.js

Syntax

JavaScript

```
function redo ()
```

See Also

ChangeItemCommand Members
ChangeItemCommand Class
MindFusion.Diagramming Namespace

ChangeItemCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: ChangeItemCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

ChangeItemCommand Members
ChangeItemCommand Class
MindFusion.Diagramming Namespace

ColumnStyle Enumeration

Specifies how table columns' width is set.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
ColumnStyle = {}
```

Members

	Member name	Description
	AutoWidth	The columns' width changes when the table is resized.
	FixedWidth	The columns has fixed, unchangeable width.

See Also

MindFusion.Diagramming Namespace

Command Class

Represents an action that changes the diagram and whose effects can be undone.

Syntax

```
JavaScript
// class
Command.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
MindFusion.Diagramming.AddItemCommand
MindFusion.Diagramming.ChangeItemCommand
MindFusion.Diagramming.CompositeCommand
MindFusion.Diagramming.ContainerPagesCommand
MindFusion.Diagramming.FoldContainerCommand
MindFusion.Diagramming.PageContainerCommand
MindFusion.Diagramming.RemoveItemCommand
MindFusion.Diagramming.ResizeTableColumnCommand
MindFusion.Diagramming.ResizeTableRowCommand
MindFusion.Diagramming.ScrollTableCommand

See Also

Command Members
MindFusion.Diagramming Namespace

Command Members

Represents an action that changes the diagram and whose effects can be undone.

The following tables list the members exposed by the Command type.

Public Constructors

	Name	Description
	Command	Initializes a new instance of the **Command** class.

Public Methods

	Name	Description
	execute	Carries out an action that changes the diagram or a diagram item.
	redo	Repeats an action that has been undone.
	undo	Undoes an action, restoring the diagram to the state it was in before carrying out the action.

See Also

Command Class
MindFusion.Diagramming Namespace

Command Constructor

Initializes a new instance of the Command class.

Namespace: MindFusion.Diagramming
File: Command.js

Syntax

JavaScript

```
function Command (diagram)
```

Parameters

diagram
The Diagram modified by this command.

See Also

Command Class
MindFusion.Diagramming Namespace

Command Methods

Command.execute Method

Carries out an action that changes the diagram or a diagram item.

Namespace: MindFusion.Diagramming
File: Command.js

Syntax

JavaScript

```
function execute ()
```

See Also

Command Members
Command Class
MindFusion.Diagramming Namespace

Command.redo Method

Repeats an action that has been undone.

Namespace: MindFusion.Diagramming
File: Command.js

Syntax

JavaScript

```
function redo ()
```

See Also

Command Members
Command Class
MindFusion.Diagramming Namespace
actionRedone Event

Command.undo Method

Undoes an action, restoring the diagram to the state it was in before carrying out the action.

Namespace: MindFusion.Diagramming
File: Command.js

Syntax

```
JavaScript
function undo ()
```

See Also

Command Members
Command Class
MindFusion.Diagramming Namespace
actionUndone Event

CompatConfig Class

Contains API compatibility settings.

Syntax

```
JavaScript
// class
CompatConfig.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.CompatConfig

See Also

CompatConfig Members
MindFusion.Diagramming Namespace

CompatConfig Members

Contains API compatibility settings.

The following tables list the members exposed by the CompatConfig type.

Public Fields

	Name	Description
	gridLayout	Indicates whether to use CSS grid layout for diagram DOM.
	pixelThickness	Indicates whether stroke thickness values are specified in pixels or diagram units.

Public Properties

	Name	Description
	propFunctions	Gets or sets whether the control should generate get/set functionsfor properties to keep compatibility with older versions of the diagramming API.

See Also

CompatConfig Class
MindFusion.Diagramming Namespace

CompatConfig Fields

CompatConfig.gridLayout Field

Indicates whether to use CSS grid layout for diagram DOM.

Namespace: MindFusion.Diagramming
File: CompatConfig.js

Syntax

JavaScript

```
CompatConfig.gridLayout = true;
```

Remarks

This is a static field. Set it to **false** in order to use VirtualScroll mode in older browsers that don't support grid layout. Note that **false** value applies absolute positioning to the diagram's parent div, which could entail further layout changes in the html page.

See Also

CompatConfig Members
CompatConfig Class
MindFusion.Diagramming Namespace

CompatConfig.pixelThickness Field

Indicates whether stroke thickness values are specified in pixels or diagram units.

Namespace: MindFusion.Diagramming
File: CompatConfig.js

Syntax

JavaScript

```
CompatConfig.pixelThickness = false;
```

Remarks

This is a static field. Set it to **true** to make StrokeThickness of DiagramItem and Style objects regardes as pixels instead of MeasureUnit values.

StrokeThickness was changed from pixels to MeasureUnit in version 3.5.1 of the library.

See Also

CompatConfig Members
CompatConfig Class
MindFusion.Diagramming Namespace

CompatConfig Properties

CompatConfig.propFunctions Property

Gets or sets whether the control should generate get/set functionsfor properties to keep compatibility with older versions of the diagramming API.

Namespace: MindFusion.Diagramming
File: CompatConfig.js

Syntax

```
JavaScript
get propFunctions() {}
set propFunctions(value) {}
```

Property Value

Boolean

true to generate get/set function pairs, or false otherwise.

See Also

CompatConfig Members
CompatConfig Class
MindFusion.Diagramming Namespace

ComponentEventArgs Class

Contains the arguments passed to handlers of component-related events.

Syntax

```
JavaScript
// class
ComponentEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
  MindFusion.Diagramming.ItemEventArgs

MindFusion.Diagramming.NodeEventArgs
MindFusion.Diagramming.ComponentEventArgs

See Also

ComponentEventArgs Members
MindFusion.Diagramming Namespace
Events Class

ComponentEventArgs Members

Contains the arguments passed to handlers of component-related events.

The following tables list the members exposed by the ComponentEventArgs type.

Public Constructors

	Name	Description
	ComponentEventArgs	Initializes a new instance of the **ComponentEventArgs** class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user. (Inherited from ItemEventArgs.)
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	component	Gets the component for which the event was raised.
	container	Gets the container a node has been added to or removed from. (Inherited from NodeEventArgs.)
	context	Gets the canvas rendering context. (Inherited from ItemEventArgs.)
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	hyperlink	Gets the hyperlink that was clicked inside a node. (Inherited from NodeEventArgs.)
	item	Gets the diagram item for which the event is raised. (Inherited from ItemEventArgs.)
	label	Gets the node label for which the event is raised. (Inherited from NodeEventArgs.)
	mouseButton	Gets which mouse button has been pressed. (Inherited from ItemEventArgs.)
	mousePosition	Gets the current mouse pointer position. (Inherited from ItemEventArgs.)
	newText	Gets the text that has just been entered by the user. (Inherited from ItemEventArgs.)

	node	Gets the node for which the event is raised. (Inherited from NodeEventArgs.)
	oldText	Gets the text that had been displayed in the node before the user edited it. (Inherited from ItemEventArgs.)

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the creation or modification of the node. (Inherited from ItemEventArgs.)

See Also

ComponentEventArgs Class
MindFusion.Diagramming Namespace

ComponentEventArgs Constructor

Initializes a new instance of the ComponentEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function ComponentEventArgs (args)
```

Parameters

args

See Also

ComponentEventArgs Class
MindFusion.Diagramming Namespace

ComponentEventArgs Properties

ComponentEventArgs.component Property

Gets the component for which the event was raised.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get component() {}
```

Property Value

Component

An instance of the Component class.

See Also

ComponentEventArgs Members
ComponentEventArgs Class
MindFusion.Diagramming Namespace

CompositeCommand Class

Represents a set of Command instances as a single operation. Undoing or redoing the composite action, respectively undoes or redoes all its constituent subactions at the same time.

Syntax

JavaScript

```
// class
CompositeCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
MindFusion.Diagramming.CompositeCommand

See Also

CompositeCommand Members
MindFusion.Diagramming Namespace

CompositeCommand Members

Represents a set of Command instances as a single operation. Undoing or redoing the composite action, respectively undoes or redoes all its constituent subactions at the same time.

The following tables list the members exposed by the CompositeCommand type.

Public Constructors

	Name	Description
	CompositeCommand	Initializes a new instance of the **CompositeCommand** class.

Public Methods

	Name	Description
	execute	Command.execute override.
	getCommands	Gets a list containing the child **Command** objects.

	redo	Command.redo override.
	undo	Command.undo override.

See Also

CompositeCommand Class
MindFusion.Diagramming Namespace

CompositeCommand Constructor

Initializes a new instance of the CompositeCommand class.

Namespace: MindFusion.Diagramming
File: CompositeCommand.js

Syntax

JavaScript

```
function CompositeCommand (diagram)
```

Parameters

diagram
The Diagram modified by this command.

See Also

CompositeCommand Class
MindFusion.Diagramming Namespace

CompositeCommand Methods

CompositeCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: CompositeCommand.js

Syntax

JavaScript

```
function execute ()
```

See Also

CompositeCommand Members
CompositeCommand Class
MindFusion.Diagramming Namespace

CompositeCommand.getCommands Method

Gets a list containing the child Command objects.

Namespace: MindFusion.Diagramming
File: CompositeCommand.js

Syntax

```
JavaScript
function getCommands ()
```

Return Value

The list of child Command objects.

See Also

CompositeCommand Members
CompositeCommand Class
MindFusion.Diagramming Namespace

CompositeCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: CompositeCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

CompositeCommand Members
CompositeCommand Class
MindFusion.Diagramming Namespace

CompositeCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: CompositeCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

CompositeCommand Members
CompositeCommand Class
MindFusion.Diagramming Namespace

CompositeNode Class

CompositeNode implements nodes whose appearance can be defined via composition of components and layout containers.

Syntax

JavaScript

```
// class
CompositeNode.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.CompositeNode

See Also

CompositeNode Members
MindFusion.Diagramming Namespace
Composite Nodes

CompositeNode Members

CompositeNode implements nodes whose appearance can be defined via composition of components and layout containers.

The following tables list the members exposed by the **CompositeNode** type.

Public Constructors

	Name	Description
	CompositeNode	Initializes a new instance of the **CompositeNode** class with default values supplied from the specified Diagram.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)

	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	clipToOutline	Gets or sets whether contents should clip to node's outline, when outline component is set.
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)

	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)

	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the **Diagram** to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)

	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
S	classFromTemplate	Generates a node class whose appearance and properties are defined using JSON template.
	clone	DiagramNode.clone override.
	containsPoint	**DiagramItem.containsPoint** override. (Inherited from DiagramNode.)

	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	fromJson	DiagramNode.fromJson override.
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getComponent	Returns the component created for the template element with specified name.
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets the bounding rectangle of this node. (Inherited from DiagramNode.)
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding **rectangle** of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramItem.loadFromXml override. Loads the node's content from an XML element. (Inherited from DiagramNode.)
	loadTemplate	Load a JSON template defining node's appearance.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)

	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	DiagramNode.restoreState override.
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramNode.saveState override.
	saveToXml	Saves the item content into an XML element. (Inherited from DiagramNode.)
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

See Also

CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode Constructor

Initializes a new instance of the CompositeNode class with default values supplied from the specified Diagram.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

JavaScript

```
function CompositeNode ([parent])
```

Parameters

parent
Optional.
The Diagram from which to obtain default values for the node properties.

See Also

CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode Methods

CompositeNode.classFromTemplate Method

Generates a node class whose appearance and properties are defined using JSON template.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

```
JavaScript
function classFromTemplate (className, jsonTemplate, [init])
```

Parameters

className
String. Fully qualified name of the class to generate.

jsonTemplate
Object. A JavaScript object literal containing template of node's component hierarchy.

init
Optional.
function. Optional.

Return Value

CompositeNode. A constructor function that creates instances of the new class.

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.getComponent Method

Returns the component created for the template element with specified name.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

JavaScript

```
function getComponent (name)
```

Parameters

name
String. A string specifying the value of the name attribute in node's template.

Return Value

Component. The component created for specified element in the template.

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.loadTemplate Method

Load a JSON template defining node's appearance.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

JavaScript

```
function loadTemplate (jsonTemplate)
```

Parameters

jsonTemplate
Object. A JavaScript object literal containing template of node's component hierarchy.

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.restoreState Method

DiagramNode.restoreState override.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.saveState Method

DiagramNode.saveState override.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

JavaScript

```
function saveState ()
```

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeNode Properties

CompositeNode.clipToOutline Property

Gets or sets whether contents should clip to node's outline, when outline component is set.

Namespace: MindFusion.Diagramming
File: CompositeNode.js

Syntax

```
JavaScript
get clipToOutline() {}
set clipToOutline(value) {}
```

Property Value

Boolean

true to clip contents, or **false** otherwise.

Remarks

To specify the clip path, mark a Shape component in the template with **isOutline** attribute.

See Also

CompositeNode Members
CompositeNode Class
MindFusion.Diagramming Namespace

CompositeRouter Class

Implements a chain of link routers so that a link passes through the sequence until it gets routed successfully.

Syntax

```javascript
// class
CompositeRouter.prototype = {}
```

Remarks

By default diagram's linkRouter is set to a **CompositeRouter** instance that chains PatternRouter and Router. You can insert additional link routers into the routers list, including custom implementations of **Router** if target diagram should favor some specific link paths.

Inheritance Hierarchy

MindFusion.Diagramming.Router
  MindFusion.Diagramming.CompositeRouter

See Also

CompositeRouter Members
MindFusion.Diagramming Namespace

CompositeRouter Members

Implements a chain of link routers so that a link passes through the sequence until it gets routed successfully.

The following tables list the members exposed by the CompositeRouter type.

Public Constructors

	Name	Description
	CompositeRouter	Initializes a new instance of the **CompositeRouter** class.

Public Properties

	Name	Description
	avoidOverlaps	Gets or sets a value indicating whether to avoid overlapping links. (Inherited from Router.)
	diagram	Gets or sets the diagram associated with this router and all its sub-routers.
	routers	A list of router objects forming a chain through whichlinks pass until they are routed successfully.

Public Methods

	Name	Description
	route	Overrides Router.route.
	routeInternal	Overrides Router.routeInternal.

	routeLinks	Overrides Router.routeLinks.

See Also

CompositeRouter Class
MindFusion.Diagramming Namespace

CompositeRouter Constructor

Initializes a new instance of the CompositeRouter class.

Namespace: MindFusion.Diagramming
File: CompositeRouter.js

Syntax

JavaScript

```
function CompositeRouter ([diagram])
```

Parameters

diagram
Optional.
Diagram. The diagram this router is associated with.

See Also

CompositeRouter Class
MindFusion.Diagramming Namespace

CompositeRouter Methods

CompositeRouter.route Method

Overrides Router.route.

Namespace: MindFusion.Diagramming
File: CompositeRouter.js

Syntax

JavaScript

```
function route (link)
```

Parameters

link
DiagramLink. The link to route.

See Also

CompositeRouter Members
CompositeRouter Class

CompositeRouter.routeInternal Method

Overrides Router.routeInternal.

Namespace: MindFusion.Diagramming
File: CompositeRouter.js

Syntax

JavaScript

```
function routeInternal (link)
```

Parameters

link
DiagramLink. The link to route.

Return Value

Array.<Point>. An array of points representing the link's new shape, or null if routing fails.

See Also

CompositeRouter Members
CompositeRouter Class
MindFusion.Diagramming Namespace

CompositeRouter.routeLinks Method

Overrides Router.routeLinks.

Namespace: MindFusion.Diagramming
File: CompositeRouter.js

Syntax

JavaScript

```
function routeLinks (links)
```

Parameters

links
Array.<DiagramLink>. A list of links to route.

See Also

CompositeRouter Members
CompositeRouter Class
MindFusion.Diagramming Namespace

CompositeRouter Properties

CompositeRouter.diagram Property

Gets or sets the diagram associated with this router and all its sub-routers.

Namespace: MindFusion.Diagramming
File: CompositeRouter.js

Syntax

```
JavaScript
get diagram() {}
set diagram(value) {}
```

Property Value

Diagram

An instance of the Diagram class.

See Also

CompositeRouter Members
CompositeRouter Class
MindFusion.Diagramming Namespace

CompositeRouter.routers Property

A list of router objects forming a chain through whichlinks pass until they are routed successfully.

Namespace: MindFusion.Diagramming
File: CompositeRouter.js

Syntax

```
JavaScript
get routers() {}
set routers(value) {}
```

Property Value

Array

An array of Router -derived objects.

Remarks

By default the list contains PatternRouter and Router objects

See Also

CompositeRouter Members
CompositeRouter Class
MindFusion.Diagramming Namespace

ConnectionPoint Class

A class that represents a connection between an edge and a node in a graph. It decouples the DiagramLink class from the DiagramNode class, allowing edges to connect different types of nodes without knowing anything of their implementation details. Each **DiagramLink** object contains two **ConnectionPoint** instances describing the connections at both ends of the link.

Syntax

JavaScript

```
// class
ConnectionPoint.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ConnectionPoint
MindFusion.Diagramming.TableConnectionPoint
MindFusion.Diagramming.TreeViewConnectionPoint

See Also

ConnectionPoint Members
MindFusion.Diagramming Namespace

ConnectionPoint Members

A class that represents a connection between an edge and a node in a graph. It decouples the DiagramLink class from the DiagramNode class, allowing edges to connect different types of nodes without knowing anything of their implementation details. Each **DiagramLink** object contains two ConnectionPoint instances describing the connections at both ends of the link.

The following tables list the members exposed by the **ConnectionPoint** type.

Public Constructors

	Name	Description
	ConnectionPoint	Initializes a new instance of the **ConnectionPoint** class between the specified node and link.

Public Properties

	Name	Description
	endPoint	Calculates the coordinates of this connection point expressed in document coordinates.

Public Methods

	Name	Description
	addLinkToNode	Adds the **DiagramLink** associated with this connection point to the associated **DiagramNode**.

	getInitialPoint	Returns a Point (usually the center of the bounding rectangle) contained within the associated node.
	getIntersection	Calculates the intersection point between the specified line segment and the contour of the node associated with this connection point.
	getNearestAnchorPoint	Returns the nearest anchor **Point** pertinent to this connection.
	getNearestBorderPoint	Returns the nearest point lying on the **DiagramNode** border.
	getNodeRect	Returns the bounding rectangle of the node associated with this connection point.
	linkChanges	Checks whether the link would change if attached to the specified node at the specified point.
	nodesIntersect	Checks whether the node associated with the current connection point and the node associated with the specified connection point intersect.
	removeLinkFromNode	Removes the link associated with this connection point from the associated node.
	saveEndRelative	Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node.

See Also

ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint Constructor

Initializes a new instance of the ConnectionPoint class between the specified node and link.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript

```
function ConnectionPoint ([node, [link, [incoming]]])
```

Parameters

node
Optional. The node whose connection to a link is managed by this ConnectionPoint.

link
Optional. The link whose connection to a node is managed by this **ConnectionPoint**.

incoming
Optional. Specifies which end of the link is represented by this **ConnectionPoint**.

See Also

ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint Methods

ConnectionPoint.addLinkToNode Method

Adds the DiagramLink associated with this connection point to the associated DiagramNode.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript

```
function addLinkToNode ()
```

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.getInitialPoint Method

Returns a Point (usually the center of the bounding rectangle) contained within the associated node.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript

```
function getInitialPoint ()
```

Return Value

A Point contained within the node.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.getIntersection Method

Calculates the intersection point between the specified line segment and the contour of the node associated with this connection point.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

```
JavaScript
function getIntersection (point1, point2)
```

Parameters

point1
The first Point of the segment.

point2
The second **point** of the segment.

Return Value

A **Point** object representing the intersection point.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.getNearestAnchorPoint Method

Returns the nearest anchor Point pertinent to this connection.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

```
JavaScript
function getNearestAnchorPoint (point)
```

Parameters

point
A Point instance specifying location in the diagram.

Return Value

AnchorPointDetails. An AnchorPointDetails specifying the anchor point properties.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.getNearestBorderPoint Method

Returns the nearest point lying on the DiagramNode border.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript
`function getNearestBorderPoint (point)`

Parameters

point
A Point instance specifying location in the diagram.

Return Value

AnchorPointDetails. An AnchorPointDetails specifying the anchor point properties.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.getNodeRect Method

Returns the bounding rectangle of the node associated with this connection point.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript
`function getNodeRect ()`

Return Value

The bounding rectangle of the node.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.linkChanges Method

Checks whether the link would change if attached to the specified node at the specified point.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript
`function linkChanges (item, point)`

Parameters

item
A DiagramItem under the mouse cursor.

point
A point where the link will be connected.

Return Value

Boolean. true if the link would connect to a different entity if dropped at the specified location; otherwise, false.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.nodesIntersect Method

Checks whether the node associated with the current connection point and the node associated with the specified connection point intersect.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

JavaScript

```
function nodesIntersect (connectionPoint)
```

Parameters

connectionPoint
A ConnectionPoint instance.

Return Value

Boolean. true if the node associated with the current connection point and the node associated with the specified connection point intersect; otherwise, false.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.removeLinkFromNode Method

Removes the link associated with this connection point from the associated node.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

```
JavaScript
function removeLinkFromNode ()
```

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint.saveEndRelative Method

Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

```
JavaScript
function saveEndRelative ([ctrRelative])
```

Parameters

ctrRelative
Optional.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionPoint Properties

ConnectionPoint.endPoint Property

Calculates the coordinates of this connection point expressed in document coordinates.

Namespace: MindFusion.Diagramming
File: ConnectionPoint.js

Syntax

```
JavaScript
get endPoint() {}
```

Property Value

Point

A Point that represents the coordinate.

See Also

ConnectionPoint Members
ConnectionPoint Class
MindFusion.Diagramming Namespace

ConnectionStyle Enumeration

Specifies whether links should connect to a node or its rows.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
ConnectionStyle = {}
```

Members

	Member name	Description
	Node	Links should connect to the table node when drawn interactively.
	Rows	Links should connect to one of the table's rows when drawn interactively.

Remarks

Members of this enumeration can be assigned to the connectionStyle property of TableNode or TreeViewNode.

See Also

MindFusion.Diagramming Namespace

ContainerNode Class

ContainerNode instances are diagram nodes that can contain other nodes.

Syntax

JavaScript

```
// class
ContainerNode.prototype = {}
```

Remarks

Child node can be added to or removed from containers interactively when the allowAddChildren and allowRemoveChildren properties are enabled. Use the add method to add a node to container

programmatically. Use the remove method to remove a node from the container.

A container node can be folded and unfolded, either interactively or programmatically, as specified through the foldable and folded properties. When a container is folded, it occupies a minimal area and its child nodes are hidden.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.ContainerNode
      MindFusion.Diagramming.PagedContainerNode

See Also

ContainerNode Members
MindFusion.Diagramming Namespace

ContainerNode Members

ContainerNode instances are diagram nodes that can contain other nodes.

The following tables list the members exposed by the ContainerNode type.

Public Constructors

	Name	Description
	ContainerNode	Initializes a new instance of the ContainerNode class with default values supplied from the specified Diagram.

Public Properties

	Name	Description
	allowAddChildren	Gets or sets whether users are allowed to add child nodes to the container.
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	allowRemoveChildren	Gets or sets whether users are allowed to remove child nodes from the container.
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	autoDeleteChildren	Gets or sets whether whether deleting this container should also remove its child items from the diagram.
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)

	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	captionBackBrush	Gets or sets an object specifying how to fill the caption bar.
	captionFont	Gets or sets the font used to render the node's caption text. (Inherited from CaptionedNode.)
	captionHeight	Gets or sets the height of the node's caption bar.
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	children	Gets the child nodes of this container.
	clipChildren	Gets or sets whether child items should be clipped by container boundaries.
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent **ContainerNode** containing this node. (Inherited from DiagramItem.)
	contentPath	Creates a path corresponding to the content area rectangle, excluding caption bar.
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	dividerStroke	Gets or sets the color used to stroke the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	dividerStrokeDashArray	Gets or sets an array of values specifying the dash pattern of the node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from CaptionedNode.)
	dividerStrokeDashStyle	Gets or sets the line dash pattern applied when stroking node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeThickness	Gets or sets the width of the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveDividerStrokeThickness	Gets the effective line thickness that should be applied when stroking this node's divider line. (Inherited from CaptionedNode.)

	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	enableStyledText	Gets or sets a value indicating whether styled caption text rendering is enabled.
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	foldable	Gets or sets a value indicating whether users are allowed to fold the container.

	folded	Gets or sets a value indicating whether the container is folded.
	foldedSize	Gets or sets the size of the container in folded state.
	foldIconSize	Gets or sets a value indicating the size of the fold/unfold icon.
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)

	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the shape of the node's outline.
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	unfoldedSize	Gets the size of the container in unfolded state.

	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	add	Adds a node to this container.
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	arrange	Arranges the container's children with the given layout.
	arrangeAnimated	Arranges the container's children with the given layout, and animates items moving to their new positions.
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	Checks if the **ContainerNode** contains the specified point.
	containsRecursively	Determines whether the specified node is contained within this container, either directly or inside child containers.
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramNode.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)

	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets an array of points that define the outline of this node shape.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point.
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramNode.LoadFromXml override. Loads the node's content from an XML element.
	onRemove	DiagramItem.onRemove override.
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	remove	Removes a node from this container.
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	resizeToFitChildren	Resizes the container making it big enough to contain its child nodes.
	resizeToFitText	Makes the container node caption big enough to display its text without clipping.
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	DiagramNode.restoreState override.
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramNode.saveState override.
	saveToXml	Saves the node's content into an XML element.

	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)
	visitNodes	Visits the diagram nodes in reverse Z order.

See Also

ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode Constructor

Initializes a new instance of the ContainerNode class with default values supplied from the specified Diagram.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function ContainerNode ([parent])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the node properties.

See Also

ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode Methods

ContainerNode.add Method

Adds a node to this container.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript
`function add (node)`

Parameters

node
DiagramNode. The node that should be added to the container.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
remove Method
children Property

ContainerNode.arrange Method

Arranges the container's children with the given layout.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript
`function arrange (layout)`

Parameters

layout
Object. The Layout algorithm to apply.

Remarks

Available layout algorithms are SpringLayout, TreeLayout, BorderedTreeLayout, LayeredLayout, TreeMapLayout, FractalLayout, FlowchartLayout, TopologicalLayout, OrthogonalLayout, RadialTreeLayout, GridLayout.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
TreeLayout Class
LayeredLayout Class
SpringLayout Class

ContainerNode.arrangeAnimated Method

Arranges the container's children with the given layout, and animates items moving to their new positions.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function arrangeAnimated (layout, [duration, [animationType, [easingType]]])
```

Parameters

layout
Object. The Layout algorithm to apply.

duration
Optional.
Number. An integer, specifying the duration of the animation in milliseconds.

animationType
Optional.
A member of the AnimationType enumeration, specifying the animation type to use.

easingType
Optional.
A member of the EasingType enumeration, specifying the type of easing function to apply to the animation.

Remarks

Available layout algorithms are SpringLayout, TreeLayout, BorderedTreeLayout, LayeredLayout, TreeMapLayout, FractalLayout, FlowchartLayout, TopologicalLayout, OrthogonalLayout, RadialTreeLayout, GridLayout.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.containsPoint Method

Checks if the ContainerNode contains the specified point.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function containsPoint (point)
```

Parameters

point
The Point to check.

Return Value

true if the point is within the bounds of this node; otherwise, **false**.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.containsRecursively Method

Determines whether the specified node is contained within this container, either directly or inside child containers.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function containsRecursively (node)
```

Parameters

node

Return Value

Boolean. **true** if the specified node is inside the container, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.getOutline Method

Gets an array of points that define the outline of this node shape.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function getOutline ()
```

Return Value

An array containing the outline points.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.hitTest Method

Returns the diagram item that contains the specified point.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function hitTest (point, [threshold, onlyNodes, onlyInteractive])
```

Parameters

point
Point. A Point value specifying the logical coordinates of a diagram point.

threshold
Optional.
Number. Specifies the size of the area around the item outlines where the point is still considered inside the item.

onlyNodes
Boolean. true to exclude child links from the search.

onlyInteractive
Boolean. true to exclude locked items from the search.

Return Value

DiagramItem. The item that contains the specified point.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.loadFromXml Method

DiagramNode.LoadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.onRemove Method

DiagramItem.onRemove override.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function onRemove ()
```

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.remove Method

Removes a node from this container.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function remove (node)
```

Parameters

node
DiagramNode. The node that should be removed from the container.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
add Method
children Property

ContainerNode.resizeToFitChildren Method

Resizes the container making it big enough to contain its child nodes.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
function resizeToFitChildren (allowShrink, margin)
```

Parameters

allowShrink
true to allow setting smaller size than current size of the container, or **false** otherwise.

margin

Specifies the margin space to leave between container's borders and child nodes.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
children Property

ContainerNode.resizeToFitText Method

Makes the container node caption big enough to display its text without clipping.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function resizeToFitText (fit, [fixPosition, [foldBtnWidth]])
```

Parameters

fit
FitSize. A member of the FitSize enumeration.

fixPosition
Optional.
Boolean. Fix the position of the container to the original location. The default is false.

foldBtnWidth
Optional.
Number. The width of the folder button, if it's not provided, the value from foldIconSize is used.

Return Value

Boolean. **true** if the caption is resized successfully, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.restoreState Method

DiagramNode.restoreState override.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.saveState Method

DiagramNode.saveState override.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function saveState ()
```

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function toJson ()
```

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.visitNodes Method

Visits the diagram nodes in reverse Z order.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
function visitNodes (point, visitNode, relatedItem)
```

Parameters

point

visitNode

relatedItem

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode Properties

ContainerNode.allowAddChildren Property

Gets or sets whether users are allowed to add child nodes to the container.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get allowAddChildren() {}
set allowAddChildren(value) {}
```

Property Value

Boolean

true if child nodes can be added interactively, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.allowRemoveChildren Property

Gets or sets whether users are allowed to remove child nodes from the container.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get allowRemoveChildren() {}
set allowRemoveChildren(value) {}
```

Property Value

Boolean

true if child nodes can be removed interactively, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.autoDeleteChildren Property

Gets or sets whether whether deleting this container should also remove its child items from the diagram.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get autoDeleteChildren() {}
set autoDeleteChildren(value) {}
```

Property Value

Boolean

true to automatically delete child items, or **false** otherwise. The default value is **false**.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
children Property

ContainerNode.captionBackBrush Property

Gets or sets an object specifying how to fill the caption bar.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get captionBackBrush() {}
set captionBackBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
brush Property
captionHeight Property

ContainerNode.captionHeight Property

Gets or sets the height of the node's caption bar.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get captionHeight() {}
set captionHeight(value) {}
```

Property Value

Number

A number specifying the height of the caption bar. The default value is 6.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.children Property

Gets the child nodes of this container.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get children() {}
```

Property Value

Array

An array containing the child nodes of this container.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
add Method
remove Method
autoDeleteChildren Property
resizeToFitChildren Method

ContainerNode.clipChildren Property

Gets or sets whether child items should be clipped by container boundaries.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get clipChildren() {}
set clipChildren(value) {}
```

Property Value

Boolean

true if child items are clipped, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
shape Property

ContainerNode.contentPath Property

Creates a path corresponding to the content area rectangle, excluding caption bar.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
get contentPath() {}
```

Property Value

Path

A Path instance.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.enableStyledText Property

Gets or sets a value indicating whether styled caption text rendering is enabled.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
get enableStyledText() {}
set enableStyledText(value) {}
```

Property Value

Boolean

true to parse the node's caption text in order to find formatting tags, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace

ContainerNode.foldable Property

Gets or sets a value indicating whether users are allowed to fold the container.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get foldable() {}
set foldable(value) {}
```

Property Value

Boolean

true if the container can be folded, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
containerFolded Event
containerUnfolded Event
folded Property
foldIconSize Property
foldedSize Property

ContainerNode.folded Property

Gets or sets a value indicating whether the container is folded.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get folded() {}
set folded(value) {}
```

Property Value

Boolean

true if the container is folded, or **false** otherwise.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
foldable Property
foldedSize Property

ContainerNode.foldedSize Property

Gets or sets the size of the container in folded state.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get foldedSize() {}
set foldedSize(value) {}
```

Property Value

Size

A Size instance specifying the folded size of a container.

Remarks

If this property is not set, the width of the folded container is set to its minimumSize.width, and the height to captionHeight.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
unfoldedSize Property

ContainerNode.foldIconSize Property

Gets or sets a value indicating the size of the fold/unfold icon.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

JavaScript

```
get foldIconSize() {}
set foldIconSize(value) {}
```

Property Value

Number

A number specifying icon size.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
foldable Property

ContainerNode.shape Property

Gets or sets the shape of the node's outline.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
get shape() {}
set shape(value) {}
```

Property Value

SimpleShape

A member of the SimpleShape enumeration.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
cornerRadius Property

ContainerNode.unfoldedSize Property

Gets the size of the container in unfolded state.

Namespace: MindFusion.Diagramming
File: ContainerNode.js

Syntax

```
JavaScript
get unfoldedSize() {}
```

Property Value

Size

The size of the container in unfolded state.

See Also

ContainerNode Members
ContainerNode Class
MindFusion.Diagramming Namespace
foldedSize Property

ContainerPage Class

Represents a page in a PagedContainerNode.

Syntax

JavaScript
`// class` `ContainerPage.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.ContainerPage

See Also

ContainerPage Members
MindFusion.Diagramming Namespace

ContainerPage Members

Represents a page in a PagedContainerNode.

The following tables list the members exposed by the ContainerPage type.

Public Constructors

	Name	Description
	ContainerPage	Initializes a new instance of the ContainerPage class.

Public Properties

	Name	Description
	links	Gets the links on this page.
	nodes	Gets the nodes on this page.
	title	Gets or sets the title of this page.

Public Methods

	Name	Description
	fromJson	Deserializes the page from a JSON object.
	loadFromXml	Loads the page's content from an XML element.
	saveToXml	Saves the page's content into an XML element.
	toJson	Serializes the page to a JSON object.

See Also

ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage Constructor

Initializes a new instance of the ContainerPage class.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
function ContainerPage ([prototype])
```

Parameters

prototype
Optional.
ContainerPage. A prototype to copy.

See Also

ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage Methods

ContainerPage.fromJson Method

Deserializes the page from a JSON object.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
function fromJson (json, context)
```

Parameters

json

context

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage.loadFromXml Method

Loads the page's content from an XML element.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process.

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage.saveToXml Method

Saves the page's content into an XML element.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process.

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage.toJson Method

Serializes the page to a JSON object.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
```

```
function toJson (context)
```

Parameters

context

Return Value

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage Properties

ContainerPage.links Property

Gets the links on this page.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

JavaScript

```
get links() {}
set links(value) {}
```

Property Value

Array

An array of DiagramLink objects.

Remarks

Unlike nodes, the control does not add links to this collection automatically, as ownership could be ambiguous (when considering nested containers, cross-container links, etc). This list lets you associate links with a page yourself when you want it hidden along with the page. The container automatically hides links between direct child nodes of non-current page.

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage.nodes Property

Gets the nodes on this page.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
get nodes() {}
set nodes(value) {}
```

Property Value

Array

An array of DiagramNode objects.

Remarks

PagedContainerNode automatically adds child nodes to this list when users drag them into currentPage.

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPage.title Property

Gets or sets the title of this page.

Namespace: MindFusion.Diagramming
File: ContainerPage.js

Syntax

```
JavaScript
get title() {}
set title(value) {}
```

Property Value

String

A string containing title text.

Remarks

PagedContainerNode set its text to the title of currentPage.

See Also

ContainerPage Members
ContainerPage Class
MindFusion.Diagramming Namespace

ContainerPagesCommand Class

Tracks changes of the pages collection of a PagedContainerNode.

Syntax

JavaScript

```
// class
ContainerPagesCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.ContainerPagesCommand

See Also

ContainerPagesCommand Members
MindFusion.Diagramming Namespace

ContainerPagesCommand Members

Tracks changes of the pages collection of a PagedContainerNode.

The following tables list the members exposed by the ContainerPagesCommand type.

Public Constructors

	Name	Description
	ContainerPagesCommand	Initializes a new instance of the ContainerPagesCommand class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

ContainerPagesCommand Class
MindFusion.Diagramming Namespace

ContainerPagesCommand Constructor

Initializes a new instance of the ContainerPagesCommand class.

Namespace: MindFusion.Diagramming
File: ContainerPagesCommand.js

Syntax

JavaScript

```
function ContainerPagesCommand (container, name)
```

Parameters

container
PagedContainerNode. The PagedContainerNode modified by this command.

name
string. The name of the command.

See Also

ContainerPagesCommand Class
MindFusion.Diagramming Namespace

ContainerPagesCommand Methods

ContainerPagesCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: ContainerPagesCommand.js

Syntax

JavaScript

```
function execute ()
```

See Also

ContainerPagesCommand Members
ContainerPagesCommand Class
MindFusion.Diagramming Namespace

ContainerPagesCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: ContainerPagesCommand.js

Syntax

JavaScript

```
function redo ()
```

See Also

ContainerPagesCommand Members
ContainerPagesCommand Class
MindFusion.Diagramming Namespace

ContainerPagesCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: ContainerPagesCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

ContainerPagesCommand Members
ContainerPagesCommand Class
MindFusion.Diagramming Namespace

ControlNode Class

ControlNode instances are diagram nodes that render HTML content.

Syntax

```
JavaScript
// class
ControlNode.prototype = {}
```

Remarks

ControlNode objects display custom HTML content, specified via their template property. A node's DOM structure is created inside the Content div element, which is then rendered on top of the diagram canvas. Note that this prevents **ControlNode**s drawing in diagram' usual Z order, and they will always appear on top of other diagram items.

By default the diagram intercepts mouse events of all hosted html elements to enable moving nodes or drawing links. You can specify that interactive elements, such as buttons or text boxes, should receive input instead by setting their **data-interactive** attribute to **true**.

You can register event handlers for elements in the template by assigning function names to **data-event** attributes in the form data-event-'eventName'='handlerName'. Alternatively, you can attach event listeners using DOM API once nodeDomCreated is raised to signify that **content** is available. E.g. access child elements by calling args.getNode().getContent().querySelector(...) and attach handlers via elements' **addEventListener** method.

ControlNodes create DOM elements from their template only in the main diagram view, and render images instead when shown in Overview or NodeListView. Note that in this case image elements in the template will render only if specified in base64 format. The Utils.toDataUrl method helps you convert an image URL to base64 data URL.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.ControlNode

See Also

ControlNode Members
MindFusion.Diagramming Namespace

ControlNode Members

ControlNode instances are diagram nodes that render HTML content.

The following tables list the members exposed by the **ControlNode** type.

Public Constructors

	Name	Description
	ControlNode	Initializes a new instance of the **ControlNode** class with default values supplied from the specified Diagram.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)

	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)

	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the **Diagram** to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)

	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	template	Gets or sets HTML string representing the DOM content rendered by this node.
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	DiagramItem.visible override. Gets or sets whether this node is visible.
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	**DiagramItem.containsPoint** override. (Inherited from DiagramNode.)
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	createImage	Creates the background svg image.
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	fromJson	DiagramNode.fromJson override.
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getContent	Gets the content element of this node.
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets the bounding rectangle of this node. (Inherited from DiagramNode.)
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)

	getRotatedBounds	Gets the bounding **rectangle** of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	init	Returns a Builder object used to configure this ControlNode instance.
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramNode.loadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	onUpdateBounds	DiagramNode.onUpdateBounds override.
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramNode.removeCanvasElements override.
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	restoreState	DiagramItem.restoreState override. (Inherited from DiagramNode.)
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramItem.saveState override. (Inherited from DiagramNode.)
	saveToXml	Saves the node's content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

	With	Returns a Builder object used to configure and create new **ControlNode** instances.

See Also

ControlNode Class
MindFusion.Diagramming Namespace

ControlNode Constructor

Initializes a new instance of the ControlNode class with default values supplied from the specified Diagram.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function ControlNode ([diagramView])
```

Parameters

diagramView
Optional.
DiagramView. The DiagramView from which to obtain default values for the node properties.

See Also

ControlNode Class
MindFusion.Diagramming Namespace

ControlNode Methods

ControlNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function clone ()
```

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.createImage Method

Creates the background svg image.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

```
JavaScript
function createImage ()
```

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.getContent Method

Gets the content element of this node.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

```
JavaScript
function getContent ()
```

Return Value

The **div** element hosting HTML content generated from template.

Remarks

The diagram raises nodeDomCreated event once DOM elements have been created from template.

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.init Method

Returns a Builder object used to configure this ControlNode instance.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function init ()
```

Return Value

A Builder instance.

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.loadFromXml Method

DiagramNode.loadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

ControlNode Members
ControlNode Class

MindFusion.Diagramming Namespace

ControlNode.onUpdateBounds Method

DiagramNode.onUpdateBounds override.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function onUpdateBounds ()
```

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.removeCanvasElements Method

DiagramNode.removeCanvasElements override.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function removeCanvasElements ()
```

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function toJson ()
```

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.With Method

Returns a Builder object used to configure and create new ControlNode instances.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

JavaScript

```
function With (diagram)
```

Parameters

diagram

Return Value

A Builder instance.

See Also

ControlNode Members
ControlNode Class

ControlNode Properties

ControlNode.template Property

Gets or sets HTML string representing the DOM content rendered by this node.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

```
JavaScript
get template() {}
set template(value) {}
```

Property Value

String

A string containing HTML markup.

Remarks

The diagram raises nodeDomCreated event once DOM elements have been created from this template.

See Also

ControlNode Members
ControlNode Class
MindFusion.Diagramming Namespace

ControlNode.visible Property

DiagramItem.visible override. Gets or sets whether this node is visible.

Namespace: MindFusion.Diagramming
File: ControlNode.js

Syntax

```
JavaScript
get visible() {}
set visible(value) {}
```

Property Value

Boolean

true if this node should be displayed, or **false** otherwise.

See Also

CreateLinkController Class

A controller used to draw new links in the diagram.

Syntax

JavaScript

```
// class
CreateLinkController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
  MindFusion.Diagramming.CreateLinkController

See Also

CreateLinkController Members
MindFusion.Diagramming Namespace

CreateLinkController Members

A controller used to draw new links in the diagram.

The following tables list the members exposed by the CreateLinkController type.

Public Constructors

	Name	Description
	CreateLinkController	Initializes a new instance of the CreateLinkController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	SinglePointerController.commit override.
	drawInteraction	SinglePointerController.drawInteraction override.
	move	SinglePointerController.move override.
	start	SinglePointerController.start override.
	validate	SinglePointerController.validate override.

See Also

CreateLinkController Class
MindFusion.Diagramming Namespace

CreateLinkController Constructor

Initializes a new instance of the CreateLinkController class.

Namespace: MindFusion.Diagramming
File: CreateLinkController.js

Syntax

JavaScript

```
function CreateLinkController (diagram, link)
```

Parameters

diagram
Diagram. The parent diagram of the new link.

link
DiagramLink. The link that will be drawn by this controller.

See Also

CreateLinkController Class
MindFusion.Diagramming Namespace

CreateLinkController Methods

CreateLinkController.commit Method

SinglePointerController.commit override.

Namespace: MindFusion.Diagramming
File: CreateLinkController.js

Syntax

JavaScript

```
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateLinkController Members
CreateLinkController Class
MindFusion.Diagramming Namespace

CreateLinkController.drawInteraction Method

SinglePointerController.drawInteraction override.

Namespace: MindFusion.Diagramming
File: CreateLinkController.js

Syntax

```
JavaScript
function drawInteraction (context)
```

Parameters

context
DrawingContext2D. The drawing context for diagram's underlying canvas.

See Also

CreateLinkController Members
CreateLinkController Class
MindFusion.Diagramming Namespace

CreateLinkController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: CreateLinkController.js

Syntax

```
JavaScript
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateLinkController Members
CreateLinkController Class
MindFusion.Diagramming Namespace

CreateLinkController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: CreateLinkController.js

Syntax

```
JavaScript
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateLinkController Members
CreateLinkController Class
MindFusion.Diagramming Namespace

CreateLinkController.validate Method

SinglePointerController.validate override.

Namespace: MindFusion.Diagramming
File: CreateLinkController.js

Syntax

```
JavaScript
function validate (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

Return Value

Boolean. true to accept the operation, or false otherwise.

See Also

CreateLinkController Members
CreateLinkController Class
MindFusion.Diagramming Namespace

CreateNodeController Class

A controller used to draw new nodes in the diagram.

Syntax

```
JavaScript
```

```
// class
CreateNodeController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
MindFusion.Diagramming.CreateNodeController

See Also

CreateNodeController Members
MindFusion.Diagramming Namespace

CreateNodeController Members

A controller used to draw new nodes in the diagram.

The following tables list the members exposed by the CreateNodeController type.

Public Constructors

	Name	Description
	CreateNodeController	Initializes a new instance of the CreateNodeController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	SinglePointerController.commit override.
	drawInteraction	SinglePointerController.drawInteraction override.
	move	SinglePointerController.move override.
	start	SinglePointerController.start override.
	validate	SinglePointerController.validate override.

See Also

CreateNodeController Class
MindFusion.Diagramming Namespace

CreateNodeController Constructor

Initializes a new instance of the CreateNodeController class.

Namespace: MindFusion.Diagramming
File: CreateNodeController.js

Syntax

JavaScript

```
function CreateNodeController (diagram, node)
```

Parameters

diagram
Diagram. The parent diagram of the new node.

node
DiagramNode. The node that will be drawn by this controller.

See Also

CreateNodeController Class
MindFusion.Diagramming Namespace

CreateNodeController Methods

CreateNodeController.commit Method

SinglePointerController.commit override.

Namespace: MindFusion.Diagramming
File: CreateNodeController.js

Syntax

JavaScript

```
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateNodeController Members
CreateNodeController Class
MindFusion.Diagramming Namespace

CreateNodeController.drawInteraction Method

SinglePointerController.drawInteraction override.

Namespace: MindFusion.Diagramming
File: CreateNodeController.js

Syntax

JavaScript

```
function drawInteraction (context)
```

Parameters

context
DrawingContext2D. The drawing context for diagram's underlying canvas.

See Also

CreateNodeController Members
CreateNodeController Class
MindFusion.Diagramming Namespace

CreateNodeController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: CreateNodeController.js

Syntax

JavaScript

```
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateNodeController Members
CreateNodeController Class
MindFusion.Diagramming Namespace

CreateNodeController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: CreateNodeController.js

Syntax

JavaScript

```
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateNodeController Members
CreateNodeController Class
MindFusion.Diagramming Namespace

CreateNodeController.validate Method

SinglePointerController.validate override.

Namespace: MindFusion.Diagramming
File: CreateNodeController.js

Syntax

```
JavaScript
function validate (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

Return Value

Boolean. true to accept the operation, or false otherwise.

See Also

CreateNodeController Members
CreateNodeController Class
MindFusion.Diagramming Namespace

CreateSelectionController Class

A controller used to draw selection rectangle in the diagram.

Syntax

```
JavaScript
// class
CreateSelectionController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
MindFusion.Diagramming.CreateSelectionController

See Also

CreateSelectionController Members
MindFusion.Diagramming Namespace

CreateSelectionController Members

A controller used to draw selection rectangle in the diagram.

The following tables list the members exposed by the CreateSelectionController type.

Public Constructors

	Name	Description
	CreateSelectionController	Initializes a new instance of the CreateSelectionController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	SinglePointerController.commit override.
	drawInteraction	SinglePointerController.drawInteraction override.
	move	SinglePointerController.move override.
	start	SinglePointerController.start override.
	validate	Called to validate the operation at current pointer position. (Inherited from SinglePointerController.)

See Also

CreateSelectionController Class
MindFusion.Diagramming Namespace

CreateSelectionController Constructor

Initializes a new instance of the CreateSelectionController class.

Namespace: MindFusion.Diagramming
File: CreateSelectionController.js

Syntax

JavaScript

```
function CreateSelectionController (selection)
```

Parameters

selection
Selection. The selection that will be drawn by this controller.

See Also

CreateSelectionController Methods

CreateSelectionController.commit Method

SinglePointerController.commit override.

Namespace: MindFusion.Diagramming
File: CreateSelectionController.js

Syntax

```
JavaScript
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateSelectionController Members
CreateSelectionController Class
MindFusion.Diagramming Namespace

CreateSelectionController.drawInteraction Method

SinglePointerController.drawInteraction override.

Namespace: MindFusion.Diagramming
File: CreateSelectionController.js

Syntax

```
JavaScript
function drawInteraction (context)
```

Parameters

context
DrawingContext2D. The drawing context for diagram's underlying canvas.

See Also

CreateSelectionController Members
CreateSelectionController Class
MindFusion.Diagramming Namespace

CreateSelectionController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: CreateSelectionController.js

Syntax

```
JavaScript
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateSelectionController Members
CreateSelectionController Class
MindFusion.Diagramming Namespace

CreateSelectionController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: CreateSelectionController.js

Syntax

```
JavaScript
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

CreateSelectionController Members
CreateSelectionController Class
MindFusion.Diagramming Namespace

DelKeyAction Enumeration

Specifies the type of action performed when the user presses the DEL key.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript
`// enum` `DelKeyAction = {}`

Members

	Member name	Description
	DeleteActiveItem	Indicates that only the activeItem is deleted.
	DeleteSelectedItems	Indicates that all selected items are deleted.
	None	Indicates that nothing happens.

Remarks

Members of this enumeration can be assigned to the delKeyAction property of DiagramView.

See Also

MindFusion.Diagramming Namespace

Diagram Class

The **Diagram** class represents a flow diagram.

Syntax

JavaScript
`// class` `Diagram.prototype = {}`

Remarks

A diagram can contain connected nodes and links. Nodes are instances of DiagramNode-derived classes. Nodes are stored in the nodes collection. Links in MindFusion.Diagramming for JavaScript are instances of the DiagramLink class and are stored in the links collection.

Managing items

Items can be created by using the **new** operator and added to the diagram by means of the addItem method. Items can be deleted by calling the removeItem method. There are shortcut methods provided by the diagram's Factory object that can be used to create an item and add it to the diagram with just one method call, for example createShapeNode and createDiagramLink.

Controlling Users' Actions

Users actions can be validated by handling various events, fired while items are being created, modified or deleted. The allowInplaceEdit property controls whether users are allowed to edit text by double clicking on items.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramBase
MindFusion.Diagramming.Diagram
MindFusion.Diagramming.DiagramPage

See Also

Diagram Members
MindFusion.Diagramming Namespace

Diagram Members

The Diagram class represents a flow diagram.

The following tables list the members exposed by the **Diagram** type.

Public Constructors

	Name	Description
	Diagram	Initializes a new instance of the Diagram class.

Public Properties

	Name	Description
	activeItem	Gets the active diagram item.
	activeItemHandlesStyle	Gets or sets the visual style of adjustment handles of the active item.
	adjustmentHandlesSize	Gets or sets the size of adjustment handles that appear around selected items.
	alignToGrid	Gets or sets a value indicating whether the diagram items should be aligned to a grid.
	allowLinksRepeat	Gets or sets a value indicating whether more than one links can connect the same origin and destination diagram nodes.
	allowMultipleResize	Gets or sets a value indicating whether multiple selected nodes can be resized simultaneously.
	allowSelfLoops	Gets or sets a value indicating whether users are allowed to draw reflexive links.
	allowSplitLinks	Gets or sets a value indicating whether link segments can be added and removed interactively.
	allowUnanchoredLinks	Gets or sets a value indicating whether users can link nodes that do not have designated anchor points.
	allowUnconnectedLinks	Gets or sets a value indicating whether users are allowed to draw links that are not connected to any node.
	autoArrangeAvoidContainers	Gets or sets a value indicating whether auto-arranged link labels should avoid containers.
	autoArrangeAvoidSegments	Gets or sets a value indicating whether auto-arranged link labels should avoid link segments.

	autoCloseDistance	Gets or sets the maximum distance between first and last points of a FreeFormNode for which the node's outline is closed automatically.
	autoDeleteLinks	Gets or sets whether links are deleted automatically when their origin or destination is deleted.
	autoResize	Gets or sets a value indicating whether the document area should be resized automatically so it fits the diagram contents.
	backBrush	Gets or sets an object specifying how to fill the diagram background.
	backgroundImageAlign	Gets or sets a value indicating how to align the background image.
	backgroundImageContent	Gets or sets a Base64-encoded image to display as diagram background.
	backgroundImageUrl	Gets or sets URL of the image to paint as diagram's background.
	crossingRadius	Gets or sets the radius length of decorations displayed at link intersection points.
	customLinkType	Gets or sets the type of DiagramLink objects to create when users starts drawing.
	customNodeType	Gets or sets the type of DiagramNode objects to create when users starts drawing.
	defaultShape	Gets or sets the default shape assigned to new ShapeNode objects.
	dirty	Gets or sets a value indicating whether the diagram has changed since loading it.
	disabledHandlesStyle	Gets or sets the visual style of disabled adjustment handles.
	dynamicLinks	Gets or sets initial value of the dynamic property of new links.
	enableSpatialIndex	Gets or sets whether to index item positions for faster hit-testing and viewport clipping queries.
	expandButtonAction	Gets or sets how the control responds to users clickingthe +/- button displayed by expandable nodes.
	expandOnIncoming	Gets or sets a value indicating the link direction in which tree branches are expanded.
	factory	Gets a Factory object that lets you programmatically add new items to the diagram.
	font	Gets or sets the default font of text displayed by diagram items.
	freeFormAttractDistance	Gets or sets the radius around dragged free-form adjustment handle in which other points of **FreeFormNode** are modified too.

	freeFormTargets	Gets or sets an array of Shape objects used to replace **FreeFormNode** instances with ShapeNode ones when behavior is set to LinkFreeShapes or **DrawFreeShapes**.
	gridColor	Gets or sets the color of grid points.
	gridOffsetX	Gets or sets the horizontal offset of the first point of the alignment grid.
	gridOffsetY	Gets or sets the vertical offset of the first point of the alignment grid.
	gridPointSize	Gets or sets the size of shapes used to represent grid points.
	gridSizeX	Gets or sets the horizontal distance between adjacent grid points.
	gridSizeY	Gets or sets the vertical distance between adjacent grid points.
	gridStyle	Gets or sets the visual style of the alignment grid.
	items	Returns an array of all items in this diagram.
	laneGrid	Gets the lane grid.
	linkBaseShape	Gets or sets the shape displayed at the base of new links.
	linkBaseShapeSize	Gets or sets the default size of arrowheads at base of new links.
	linkBrush	Gets or sets the default fill of link arrowheads.
	linkCrossings	Gets or sets a value indicating how to render the intersection points of links.
	linkHeadShape	Gets or sets the shape displayed as head of new links.
	linkHeadShapeSize	Gets or sets the default size of arrowheads.
	linkHitDistance	Gets or sets a value specifying how far from a link a click is still considered a hit.
	linkPen	Gets or sets the default stroke of links.
	linkRouter	Gets or sets the object used to find paths for auto-routed links when diagram nodes are added or their positions change.
	links	Returns an array of all links in this diagram.
	linkSegments	Gets or sets the default number of link segments.
	linkShape	Gets or sets the initial shape assigned to new links.
	linkTextStyle	Gets or sets the default text placement and orientation of new links.
	mergeThreshold	Gets or sets the maximum distance between adjacent control points of a link at which the respective segments can be merged.
	nodeEffects	Gets the global effects applied to nodes in this diagram.
	nodes	Returns an array of all nodes in this diagram.

	nodesExpandable	Gets or sets a value indicating whether newly created nodes are expandable.
	roundedLinks	Gets or sets a value indicating whether the joints between segments of Polyline and **Cascading** links should be rounded.
	roundedLinksRadius	Gets or sets the radius of joint arcs of rounded links' segments.
	routeLinks	Gets or sets initial value of the autoRoute property of new links.
	routeMargin	Gets or sets the minimal distance between nodes and auto-routed links.
	selectedItemHandlesStyle	Gets or sets the visual style of adjustment handles of selected items.
	selection	Gets a Selection instance that represents the selection of items in this diagram.
	shadowOffsetX	Gets or sets the horizontal offset of items' shadows.
	shadowOffsetY	Gets or sets the vertical offset of items' shadows.
	shadowsStyle	Gets or sets the style of shadows.
	shapeBrush	Gets or sets the default fill of **ShapeNode** objects.
	shapeLibraryLocation	Gets or sets the location of a shape library file containing custom shape definitions. It can be set to a full URL of the file, or a relative file path.
	shapePen	Gets or sets the default stroke of **ShapeNode** objects.
	showAnchors	Gets or sets a value indicating when to display anchor point marks.
	showDisabledHandles	Gets or sets a value indicating whether adjustment handles are painted when they are disabled.
	showGrid	Gets or sets a value indicating whether the alignment grid is visible.
	showLaneGrid	Gets or sets a value indicating whether the lane grid is visible.
	style	Gets or sets the style associated with this diagram.
	tag	Gets or sets custom data associated with this diagram.
	theme	Gets or sets the theme associated with this diagram.
	undoEnabled	Gets or sets a value indicating whether saving action records for later undo is enabled.
	validityChecks	Gets or sets a value indicating whether validity checks are performed each time an item is added to the diagram.
	version	Gets a string identifying current version of the library.

Public Methods

	Name	Description
	addItem	Adds an item to the **Diagram**.
	alignPointToGrid	Returns a point of the alignment grid nearest to the one passed as an argument.
	arrange	Arranges the **diagram** with the given layout.
	arrangeAnimated	Arranges the diagram using the given layout, and animates items moving to their new positions.
	arrangeLinkLabels	Rearranges link labels whose autoArrange property is enabled.
	cancelCompositeOperation	Stops recording undo/redo commands into a CompositeCommand and discards the records created so far.
	clearAll	Removes all items from the diagram.
	commitCompositeOperation	Stops recording undo/redo commands into a **CompositeCommand** and saves the composite in the history queue.
	executeCommand	Executes the specified command on this diagram.
	fromJson	Deserializes the diagram from a JSON string.
	fromXmlDocument	Loads diagram contents from specified XML Document.
	getContentBounds	Returns the smallest rectangle that bounds all diagram items.
	getItemAt	Finds and returns the topmost DiagramItem found at the specified location.
	getItemsAt	Finds and returns an array of the **diagram items** found at the specified location.
	getLinkAt	Finds and returns the topmost **DiagramLink** found at the specified location.
	getLinksAt	Finds and returns an array of the diagram **DiagramLink**s found at the specified location.
	getNearestNode	Returns the node nearest to the specified point.
	getNodeAt	Finds and returns the topmost **DiagramNode** found at the specified location.
	getNodesAt	Finds and returns an array of the **DiagramNode**s found at the specified location.
	isItemInteractive	Checks if the specified item can be modified interactively.
	isItemVisible	Checks if the specified item is currently visible.
	loadFromString	Loads diagram contents from specified XML or JSON string.
	loadFromXml	Loads the diagram from an XML file.
	raiseClicked	Raises an event which notifies that a Diagram item has been clicked.

	raiseCreated	Raises an event which notifies that a Diagram item has been created.
	raiseDeleted	Raises an event which notifies that a Diagram item has been deleted.
	raiseDeselected	Raises an event which notifies that a Diagram item has been deselected.
	raiseDoubleClicked	Raises an event which notifies that a Diagram item has been double-clicked.
	raiseModified	Raises an event which notifies that a Diagram item has been modified.
	raisePasted	Raises an event which notifies that a Diagram item has been pasted from clipboard.
	raiseSelected	Raises an event which notifies that a Diagram item has been selected.
	raiseSelecting	Raises an event which notifies that a Diagram item will be selected.
	raiseSelectingAsync	Raises an event which notifies that a Diagram item will be selected. (Inherited from DiagramBase.)
	raiseSelectionChanged	Raises an event which notifies that the selection has changed.
	redo	Executes again an action saved in the command history.
S	registerClass	Registers a diagram item class.
S	registerComponent	Registers a component class.
S	registerItemClass	Registers a diagram item class for XML serialization support.
	removeItem	Removes the specified **item** from the **Diagram**.
	resizeToFitItem	Resize the document extents so that the specified item is contained within.
	resizeToFitItems	Resizes the document scrollable area so that it fits all diagram items.
	routeAllLinks	Routes all links in the diagram in such a way they do not cross nodes.
	saveToString	Encodes the diagram contents into an XML or JSON string.
	saveToXml	Saves the diagram to an XML file.
	startCompositeOperation	Starts recording undo/redo commands into a **CompositeCommand**.
	toJson	Serializes the diagram into a JSON string.
	toXmlDocument	Saves the diagram into an XML Document.
	undo	Undoes an action saved in the command history.

Public Events

	Name	Description
	actionRedone	Raised when an action is redone. (Inherited from DiagramBase.)
	actionUndone	Raised when an action is undone. (Inherited from DiagramBase.)
	animatedLayoutCompleted	Raised when an animated layout completes. (Inherited from DiagramBase.)
	cellTextEdited	Raised when the user edits the text of a table cell. (Inherited from DiagramBase.)
	cellTextEditing	Raised when the user starts to edit the text of a table cell. (Inherited from DiagramBase.)
	clicked	Raised when the user clicks the diagram at a point where no items are located. (Inherited from DiagramBase.)
	containerChildAdded	Raised when the user adds a node to a ContainerNode. (Inherited from DiagramBase.)
	containerChildAdding	A validation event raised to let you prevent users from adding child nodes to a container. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	containerChildRemoved	Raised when the user removes a node from a **ContainerNode**. (Inherited from DiagramBase.)
	containerChildRemoving	Raised when the user removes a node from a **ContainerNode**. (Inherited from DiagramBase.)
	containerFolded	Raised when the user folds a container by clicking the arrow-up button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	containerPageChanged	Raised when the user changes current page of a PagedContainerNode. (Inherited from DiagramBase.)
	containerUnfolded	Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	dataLoaded	Raised when the diagram data is loaded. (Inherited from DiagramBase.)
	deserializeTag	Raised when the Tag or Id of a diagram item must be deserialized from XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	drawAdjustmentHandles	Raised to draw custom adjustment handles. (Inherited from DiagramBase.)
	drawAnchorPoint	Raised when an anchor point must be custom-drawn. (Inherited from DiagramBase.)
	expandButtonClicked	Raised if **expandButtonAction** is set to RaiseEvents and the [+/-]button of an expandable node is clicked. (Inherited from DiagramBase.)

	headerResized	Raised when the user finished resizing a header. (Inherited from DiagramBase.)
	headerStartResizing	Raised when the user starts resizing a header. (Inherited from DiagramBase.)
	hitTestAdjustmentHandles	Raised to hit-test custom adjustment handles. (Inherited from DiagramBase.)
	hyperlinkClicked	Raised when the user clicks a hyperlink inside a node. (Inherited from DiagramBase.)
	initializeLink	Raised when the user starts drawing a new link, just after the DiagramLink instance is created. (Inherited from DiagramBase.)
	initializeNode	Raised when the user starts drawing a new node, just after the DiagramNode instance is created. (Inherited from DiagramBase.)
	itemAdded	Raised when an item is added to the diagram. (Inherited from DiagramBase.)
	itemRemoved	Raised when an item is removed from the diagram. (Inherited from DiagramBase.)
	laneGridCellTextEdited	Raised when the user edits the text of a grid cell. (Inherited from DiagramBase.)
	linkClicked	Raised when the user clicks a link. (Inherited from DiagramBase.)
	linkCreated	Raised when the user draws a new link. (Inherited from DiagramBase.)
	linkCreating	A validation event raised while the user is drawing a new link. (Inherited from DiagramBase.)
	linkDeleted	Raised when a link is deleted, either programmatically or by the user. (Inherited from DiagramBase.)
	linkDeleting	Raised when the user tries to delete a link. This event lets you cancel the operation. (Inherited from DiagramBase.)
	linkDeselected	Raised when a link is deselected, either programmatically or by the user. (Inherited from DiagramBase.)
	linkDoubleClicked	Raised when the user double-clicks a link. (Inherited from DiagramBase.)
	linkHovered	Raised when the mouse pointer hovers over a link. (Inherited from DiagramBase.)
	linkModified	Raised when the user moves any control point of a link. (Inherited from DiagramBase.)
	linkModifying	A validation event raised while the user is modifying a link. (Inherited from DiagramBase.)
	linkPasted	Raised when a link has been pasted from the clipboard. (Inherited from DiagramBase.)
	linkPointerEnter	Raised when the mouse pointer enters the boundaries of a link. (Inherited from DiagramBase.)

	linkPointerLeave	Raised when the mouse pointer leaves the boundaries of a link. (Inherited from DiagramBase.)
	linkRouted	Raised when a link is routed by the auto-routing algorithm. (Inherited from DiagramBase.)
	linkSelected	Raised when a link is selected, either programmatically or by the user. (Inherited from DiagramBase.)
	linkSelecting	Raised when the user tries to select a link. This event lets you cancel the operation. (Inherited from DiagramBase.)
	linkSplit	Raised when the user splits a link interactively. (Inherited from DiagramBase.)
	linkTextEdited	Raised when the user edits the text of a link. (Inherited from DiagramBase.)
	linkTextEditing	Raised when the user starts to edit the text of a link. (Inherited from DiagramBase.)
	nodeClicked	Raised when the user clicks a node. (Inherited from DiagramBase.)
	nodeCreated	Raised when the user draws a new node. (Inherited from DiagramBase.)
	nodeCreating	A validation event raised while the user is drawing a new node. (Inherited from DiagramBase.)
	nodeDeleted	Raised when a node is deleted, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeDeleting	Raised when the user tries to delete a node. This event lets you cancel the operation. (Inherited from DiagramBase.)
	nodeDeselected	Raised when a node is deselected, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeDomCreated	Raised when the content of a ControlNode is added to the page DOM. (Inherited from DiagramBase.)
	nodeDoubleClicked	Raised when the user double-clicks a node. (Inherited from DiagramBase.)
	nodeHovered	Raised when the mouse pointer hovers over a node. (Inherited from DiagramBase.)
	nodeModified	Raised when the user moves or resizes a node. (Inherited from DiagramBase.)
	nodeModifying	A validation event raised while the user is moving or resizing a node. (Inherited from DiagramBase.)
	nodePasted	Raised when a node has been pasted from the clipboard. (Inherited from DiagramBase.)
	nodePointerEnter	Raised when the mouse pointer enters the boundaries of a node. (Inherited from DiagramBase.)
	nodePointerLeave	Raised when the mouse pointer leaves the boundaries of a node. (Inherited from DiagramBase.)

	nodeSelected	Raised when a node is selected, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeSelecting	Raised when the user tries to select a node. This event lets you cancel the operation. (Inherited from DiagramBase.)
	nodeTextEdited	Raised when the user edits the text of a node. (Inherited from DiagramBase.)
	nodeTextEditing	Raised when the user starts to edit the text of a node. (Inherited from DiagramBase.)
	queryView	Raised to enumerate diagram views. (Inherited from DiagramBase.)
	selectionChanged	Raised when the diagram selection has changed. (Inherited from DiagramBase.)
	selectionModifying	A validation event raised while the user is moving or resizing multiple items. (Inherited from DiagramBase.)
	serializeTag	Raised when the Tag or Id of a diagram item must be serialized in XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from **shapeLibraryLocation**. (Inherited from DiagramBase.)
	sizeChanged	Raised when the size of the diagram canvas has changed. (Inherited from DiagramBase.)
	tableColumnResized	Raised when the user finishes resizing a column in a table. (Inherited from DiagramBase.)
	tableColumnResizing	Raised when the user resizes a column in a table interactively. (Inherited from DiagramBase.)
	tableRowResized	Raised when the user finishes resizing a row in a table. (Inherited from DiagramBase.)
	tableRowResizing	Raised when the user resizes a row in a table interactively. (Inherited from DiagramBase.)
	tableSectionCollapsed	Raised when a table section is collapsed. (Inherited from DiagramBase.)
	tableSectionExpanded	Raised when a table section is expanded. (Inherited from DiagramBase.)
	treeCollapsed	Raised when a user collapses a tree branch by clicking the [-] button of an **expandable** node. (Inherited from DiagramBase.)
	treeExpanded	Raised when a user expands a tree branch by clicking the [+] button of an **expandable** node. (Inherited from DiagramBase.)
	treeItemTextEdited	Raised when the user edits the text of a tree view item. (Inherited from DiagramBase.)
	treeItemTextEditing	Raised when the user starts to edit the text of a tree view item. (Inherited from DiagramBase.)

	validateAnchorPoint	A validation event that lets you approve or reject connecting a link to an anchor point. (Inherited from DiagramBase.)
	validateLinkCrossing	A validation event that lets you approve or reject displaying a decoration at the intersection point of links. (Inherited from DiagramBase.)

See Also

Diagram Class
MindFusion.Diagramming Namespace

Diagram Constructor

Initializes a new instance of the Diagram class.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function Diagram ()
```

See Also

Diagram Class
MindFusion.Diagramming Namespace

Diagram Methods

Diagram.addItem Method

Adds an item to the Diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function addItem (item)
```

Parameters

item
The DiagramItem to add.

Remarks

This method is called internally by methods of the Factory class. You should call it explicitly only if instantiating items using the **new** operator.

Example

JavaScript

```
var node1 = new ShapeNode(diagram);
node1.bounds = new Drawing.Rect(20, 20, 30, 30);
diagram.addItem(node1);

var node2 = diagram.factory.createShapeNode(60, 20, 30, 30);
```

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
items Property
nodes Property
links Property
validityChecks Property

Diagram.alignPointToGrid Method

Returns a point of the alignment grid nearest to the one passed as an argument.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function alignPointToGrid (point)
```

Parameters

point
Point. A point specified in diagram coordinates.

Return Value

Point. A point from the alignment grid.

Remarks

The control aligns items to the alignment grid only during user interaction. This method lets you align coordinates of programmatically created or modified items.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
showGrid Property
alignToGrid Property

Diagram.arrange Method

Arranges the diagram with the given layout.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function arrange (layout)
```

Parameters

layout
Object. The Layout algorithm to apply.

Remarks

Available layout algorithms are SpringLayout, TreeLayout, BorderedTreeLayout, LayeredLayout, TreeMapLayout, FractalLayout, FlowchartLayout, TopologicalLayout, OrthogonalLayout, RadialTreeLayout, GridLayout.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.arrangeAnimated Method

Arranges the diagram using the given layout, and animates items moving to their new positions.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function arrangeAnimated (layout, [duration, [animationType, [easingType]]])
```

Parameters

layout
Object. The Layout algorithm to apply.

duration
Optional.
Number. An integer, specifying the duration of the animation in milliseconds.

animationType
Optional.
A member of the AnimationType enumeration, specifying the animation type to use.

easingType
Optional.
A member of the EasingType enumeration, specifying the type of easing function to apply to the animation.

Remarks

Available layout algorithms are SpringLayout, TreeLayout, BorderedTreeLayout, LayeredLayout, TreeMapLayout, FractalLayout, FlowchartLayout, TopologicalLayout, OrthogonalLayout, RadialTreeLayout, GridLayout.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.arrangeLinkLabels Method

Rearranges link labels whose autoArrange property is enabled.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function arrangeLinkLabels ()
```

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
LinkLabel Class

Diagram.cancelCompositeOperation Method

Stops recording undo/redo commands into a CompositeCommand and discards the records created so far.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function cancelCompositeOperation ()
```

Remarks

You must call either commitCompositeOperation or this method when the diagram modifications are done after a startCompositeOperation call.

See Also

Diagram Members
Diagram Class

MindFusion.Diagramming Namespace
undoEnabled Property

Diagram.clearAll Method

Removes all items from the diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function clearAll ()
```

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
addItem Method
items Property

Diagram.commitCompositeOperation Method

Stops recording undo/redo commands into a CompositeCommand and saves the composite in the history queue.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function commitCompositeOperation ()
```

Remarks

You must call either cancelCompositeOperation or this method when the diagram modifications are done after a startCompositeOperation call.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
undoEnabled Property

Diagram.executeCommand Method

Executes the specified command on this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function executeCommand (command)
```

Parameters

command
The Command to execute.

Remarks

If undo/redo is enabled, the command is saved in the undo history or added to the active composite command.

AddItemCommand and RemoveItemCommand objects are created automatically by the control when adding or removing items, both programmatically or interactively. Calling this method explicitly could be necessary in order to save item's state for later undo using a ChangeItemCommand, or to run a custom operation represented by a class derived from Command.

Example

The following example shows the pattern that should be used to save the state of an item for later undo.

JavaScript

```
var ChangeItemCommand = MindFusion.Diagramming.ChangeItemCommand;
var DiagramNode = MindFusion.Diagramming.DiagramNode;

var diagram = DiagramView.find("diagram").diagram;
var node = diagram.activeItem;
if (node instanceof DiagramNode)
{
    var change = new ChangeItemCommand(diagram, node);
    node.text = "new text";
    node.brush = "red";
    diagram.executeCommand(change);
}
```

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
undoEnabled Property
undo Method
startCompositeOperation Method

Diagram.fromJson Method

Deserializes the diagram from a JSON string.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function fromJson (json)
```

Parameters

json
A string created by the toJson method.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.fromXmlDocument Method

Loads diagram contents from specified XML Document.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function fromXmlDocument (document)
```

Parameters

document
Document. A DOM Document object containing XML-serialized diagram contents.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
toXmlDocument Method

Diagram.getContentBounds Method

Returns the smallest rectangle that bounds all diagram items.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getContentBounds ([onlyVisible, [repaintBounds]])
```

Parameters

onlyVisible
Optional.
Boolean. Specifies whether only visible items should be considered.

repaintBounds
Optional.
Boolean. Specifies whether to consider the items' repaint bounds.

Return Value

Rect. A Rect instance specifying the smallest rectangle that bounds all diagram items.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getItemAt Method

Finds and returns the topmost DiagramItem found at the specified location.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getItemAt (point, [onlyInteractive])
```

Parameters

point
A Point instance identifying a diagram location.

onlyInteractive
Optional.
true to return only items that can be interacted with; otherwise, **false**.

Return Value

The topmost DiagramItem found at the given location or **null** if no **DiagramItem** has been found.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getItemsAt Method

Finds and returns an array of the diagram items found at the specified location.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getItemsAt (point)
```

Parameters

point
Point. instance identifying a location in the diagram.

Return Value

Array of the DiagramItems found at the given location or **null** if no items have been found. The items are arranged according to their Z-index.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getLinkAt Method

Finds and returns the topmost DiagramLink found at the specified location.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getLinkAt (point)
```

Parameters

point
A Point instance identifying a diagram location.

Return Value

The topmost DiagramLink found at the given location or **null** if no **DiagramLink** has been found.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getLinksAt Method

Finds and returns an array of the diagram DiagramLinks found at the specified location.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getLinksAt (point)
```

Parameters

point
Point. instance identifying a diagram location.

Return Value

Array of the DiagramLinks found at the given location or **null** if no **DiagramLink**s have been found. The links are arranged according to their Z-index.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getNearestNode Method

Returns the node nearest to the specified point.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getNearestNode (point, maxDistance, ignored)
```

Parameters

point
Point. A point in the diagram.

maxDistance
Number. The maximum distance from the point at which to look for nodes.

ignored
DiagramNode. A node that should be ignored.

Return Value

The DiagramNode nearest to the specified point.

Remarks

The last argument is useful when calling this method from nodeCreated and nodeModified handlers and the affected node should not be included in the search.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getNodeAt Method

Finds and returns the topmost DiagramNode found at the specified location.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getNodeAt (point, [onlyInteractive])
```

Parameters

point
A Point instance identifying a diagram location.

onlyInteractive
Optional.
true to return only items that can be interacted with; otherwise, **false**.

Return Value

The topmost DiagramNode found at the given location or **null** if no **DiagramNode** has been found.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.getNodesAt Method

Finds and returns an array of the DiagramNodes found at the specified location.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function getNodesAt (point)
```

Parameters

point
Point. instance identifying a diagram location.

Return Value

Array of the DiagramNodes found at the given location or null if no **DiagramNode**s have been found. The nodes are arranged according to their Z-index.

See Also

Diagram Members
Diagram Class

Diagram.isItemInteractive Method

Checks if the specified item can be modified interactively.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function isItemInteractive (item)
```

Parameters

item
DiagramItem. The DiagramItem to check.

Return Value

Boolean. **true**, if the item is currently visible and unlocked and it is not associated with a hidden or locked layer, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.isItemVisible Method

Checks if the specified item is currently visible.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function isItemVisible (item)
```

Parameters

item
DiagramItem. The DiagramItem to check.

Return Value

Boolean. **true** if the item's visible property is set to **true** and the item is not associated with a hidden layer, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.loadFromString Method

Loads diagram contents from specified XML or JSON string.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function loadFromString (str)
```

Parameters

str
String. A string containing the serialized diagram contents.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
saveToString Method

Diagram.loadFromXml Method

Loads the diagram from an XML file.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function loadFromXml (fileUrl, [onLoad, [onError]])
```

Parameters

fileUrl
String. The URL of an XML file where the data should be read from.

onLoad
Optional.
function. A callback that should be invoked if the file is loaded successfully.

onError
Optional.
function. A callback that should be invoked if the file could not be downloaded.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseClicked Method

Raises an event which notifies that a Diagram item has been clicked.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raiseClicked (item, mousePosition, mouseButton)
```

Parameters

item
DiagramItem. The Diagram item.

mousePosition
Point. The position of the mouse.

mouseButton
MouseButton. The mouse button, with which the item has been clicked.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseCreated Method

Raises an event which notifies that a Diagram item has been created.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raiseCreated (item)
```

Parameters

item
DiagramNode | DiagramLink. The diagram item that has been created.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseDeleted Method

Raises an event which notifies that a Diagram item has been deleted.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raiseDeleted (item)
```

Parameters

item
DiagramItem. The DiagramItem that has been deleted.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseDeselected Method

Raises an event which notifies that a Diagram item has been deselected.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raiseDeselected (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseDoubleClicked Method

Raises an event which notifies that a Diagram item has been double-clicked.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function raiseDoubleClicked (item, mousePosition, mouseButton)
```

Parameters

item
DiagramItem. The Diagram item.

mousePosition
Point. The position of the mouse.

mouseButton
MouseButton. The mouse button, with which the item has been clicked.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseModified Method

Raises an event which notifies that a Diagram item has been modified.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function raiseModified (item, mousePosition, adjustmentHandle)
```

Parameters

item
DiagramItem. The DiagramItem that has been modified.

mousePosition
Point. The current mouse pointer position.

adjustmentHandle
Number. Identifies the handle used to modify the item.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raisePasted Method

Raises an event which notifies that a Diagram item has been pasted from clipboard.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raisePasted (item)
```

Parameters

item
DiagramItem. The DiagramItem that has been pasted.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseSelected Method

Raises an event which notifies that a Diagram item has been selected.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raiseSelected (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseSelecting Method

Raises an event which notifies that a Diagram item will be selected.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function raiseSelecting (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.raiseSelectionChanged Method

Raises an event which notifies that the selection has changed.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function raiseSelectionChanged (oldItems, newItems)
```

Parameters

oldItems
Array. The selected items before the change.

newItems
Array. The selected items after the change.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.redo Method

Executes again an action saved in the command history.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function redo ()
```

Remarks

Calling this method has no effect before undo has been called at least once. Any command that has been undone can later be redone by **redo**. The method processes the commands in the same order they were initially executed, which is opposite to the order in which they are undone.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

undoEnabled Property
executeCommand Method
actionRedone Event

Diagram.registerClass Method

Registers a diagram item class.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function registerClass (itemClass, className, [classId, [classVersion, [baseType]]])
```

Parameters

itemClass
Object. An object identifying the item class.

className
String. A class identifier to use when saving and loading items of the specified type to/from JSON.

classId
Optional.
String. A class identifier to use when saving and loading items of the specified type to/from XML.

classVersion
Optional.
Number. A revision number of the item's class serialization format.

baseType
Optional.
Object. The base class of the item class.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.registerComponent Method

Registers a component class.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function registerComponent (componentClass, className)
```

Parameters

componentClass

Object. The component class.

className
String. The name that refers to the component class in CompositeNode templates.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.registerItemClass Method

Registers a diagram item class for XML serialization support.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function registerItemClass (itemClass, classId, classVersion)
```

Parameters

itemClass
Object. An object identifying the item class.

classId
String. A class identifier to use when saving and loading items of the specified type.

classVersion
Number. A revision number of the item's class serialization format.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.removeItem Method

Removes the specified item from the Diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function removeItem (item)
```

Parameters

item
The DiagramItem to be removed.

Remarks

The Diagram object calls this method automatically when users delete items. Classes that inherit **Diagram** can override it to implement some additional processing, but in any case they must call the base method.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
addItem Method
delKeyAction Property
showDeleteButton Property

Diagram.resizeToFitItem Method

Resize the document extents so that the specified item is contained within.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function resizeToFitItem (item)
```

Parameters

item
DiagramItem. The item to be contained.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
bounds Property

Diagram.resizeToFitItems Method

Resizes the document scrollable area so that it fits all diagram items.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function resizeToFitItems ([margin, [onlyVisible, [repaintBounds]]])
```

Parameters

margin

Optional.

onlyVisible
Optional.
Boolean. Specifies whether only visible items should be considered.

repaintBounds
Optional.
Boolean. Specifies whether to consider the items' repaint bounds.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
bounds Property

Diagram.routeAllLinks Method

Routes all links in the diagram in such a way they do not cross nodes.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function routeAllLinks ()
```

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
autoRoute Property
routeLinks Property
obstacle Property
linkRouted Event

Diagram.saveToString Method

Encodes the diagram contents into an XML or JSON string.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function saveToString (format)
```

Parameters

format
A member of the SaveToStringFormat enumeration.

Return Value

String. A string containing the diagram's serialized representation.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.saveToXml Method

Saves the diagram to an XML file.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function saveToXml (url)
```

Parameters

url
String. A URL specifying where the diagram's XML should be posted to.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
loadFromXml Method

Diagram.startCompositeOperation Method

Starts recording undo/redo commands into a CompositeCommand.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function startCompositeOperation ()
```

Remarks

After the diagram modifications are done, call commitCompositeOperation or cancelCompositeOperation.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
undoEnabled Property

Diagram.toJson Method

Serializes the diagram into a JSON string.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function toJson ()
```

Return Value

String. A string containing the diagram's JSON representation.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
fromJson Method

Diagram.toXmlDocument Method

Saves the diagram into an XML Document.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function toXmlDocument ()
```

Return Value

Document. A DOM Document object containing XML-serialized diagram contents.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
fromXmlDocument Method

Diagram.undo Method

Undoes an action saved in the command history.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
function undo ()
```

Remarks

The control supports unlimited undo/redo depth. This method moves the "current command" pointer one position back, and calling it again will undo the previous command. The commands can later be redone by calling redo.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
undoEnabled Property
actionUndone Event

Diagram Properties

Diagram.activeItem Property

Gets the active diagram item.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get activeItem() {}
```

Property Value

DiagramItem

The active item, or **null** if no item is active.

Remarks

activeItem is set to the last selected item, and has a distinct appearance from other items as set in activeItemHandlesStyle. Some possible uses for it are as target position when aligning multiple selected items, or as a source of property values when copying styles.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
selection Property

Diagram.activeItemHandlesStyle Property

Gets or sets the visual style of adjustment handles of the active item.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get activeItemHandlesStyle() {}
set activeItemHandlesStyle(value) {}
```

Property Value

HandlesVisualStyle

A HandlesVisualStyle instance describing handles' appearance.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
activeItem Property
selectedItemHandlesStyle Property
disabledHandlesStyle Property

Diagram.adjustmentHandlesSize Property

Gets or sets the size of adjustment handles that appear around selected items.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get adjustmentHandlesSize() {}
set adjustmentHandlesSize(value) {}
```

Property Value

Number

A number specifying the size of selection handles. The default value is 2.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.alignToGrid Property

Gets or sets a value indicating whether the diagram items should be aligned to a grid.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get alignToGrid() {}
set alignToGrid(value) {}
```

Property Value

Boolean

true if the diagram items should be aligned to a grid, or **false** otherwise.

Remarks

When the property is set to true, the sides of every newly created diagram item are aligned to the nearest grid points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
showGrid Property
gridStyle Property
gridSizeX Property
gridSizeX Property

Diagram.allowLinksRepeat Property

Gets or sets a value indicating whether more than one links can connect the same origin and destination diagram nodes.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get allowLinksRepeat() {}
set allowLinksRepeat(value) {}
```

Property Value

Boolean

true to let links connect to the same nodes, or **false** otherwise. The default is **true**.

Remarks

Specifies whether more than one link can connect the same origin and destination diagram nodes. If set to **false**, two diagram nodes can be connected with at most one link in a single direction.

See Also

Diagram.allowMultipleResize Property

Gets or sets a value indicating whether multiple selected nodes can be resized simultaneously.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get allowMultipleResize() {}
set allowMultipleResize(value) {}
```

Property Value

Boolean

true to let the user resize multiple selected nodes, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.allowSelfLoops Property

Gets or sets a value indicating whether users are allowed to draw reflexive links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get allowSelfLoops() {}
set allowSelfLoops(value) {}
```

Property Value

Boolean

true to allow reflexive links, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
linkCreating Event

allowUnconnectedLinks Property
allowLinksRepeat Property

Diagram.allowSplitLinks Property

Gets or sets a value indicating whether link segments can be added and removed interactively.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get allowSplitLinks() {}
set allowSplitLinks(value) {}
```

Property Value

Boolean

true if interactive adding/removing of link segments is allowed, or **false** otherwise.

Remarks

You can insert new link segments interactively by clicking and dragging a point of a selected link's segment. For polyline-style links this creates one additional segment by splitting the existing one into two parts. For cascading links, there are two segments inserted, keeping intact the alternating pattern of horizontal and vertical segments, each orthogonal to its neighbors.

Segments can be merged by moving their joining point until the angle between reaches certain threshold value. In case of cascading and Bezier links, segments can be merged by placing adjacent control points close to each other, as set in mergeThreshold.

This property also lets users change the orientation of cascading links' end segments, by dragging second or next-to-last control point diagonally.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.allowUnanchoredLinks Property

Gets or sets a value indicating whether users can link nodes that do not have designated anchor points.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get allowUnanchoredLinks() {}
set allowUnanchoredLinks(value) {}
```

Property Value

Boolean

true to let links connect to nodes without anchor points, or **false** otherwise. The default is **true**.

Remarks

allowUnanchoredLinks applies to nodes whose anchorPattern is not set or does not contain incoming or outgoing anchor points available. If the property is enabled, users can connect links to any point of such nodes. If disabled, users cannot connect links to such nodes.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
AnchorPoint Class
AnchorPattern Class

Diagram.allowUnconnectedLinks Property

Gets or sets a value indicating whether users are allowed to draw links that are not connected to any node.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get allowUnconnectedLinks() {}
set allowUnconnectedLinks(value) {}
```

Property Value

Boolean

true to let users draw unconnected links, or **false** otherwise.

Remarks

If this property is set to **true**, users will be able to draw links from or to any point of the document. To draw links that do not start from a node, you must also set behavior to DrawLinks.

Depending on which end of a link is unconnected, its origin or destination nodes will be set to a DummyNode instance.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
linkCreating Event
allowSelfLoops Property
allowLinksRepeat Property

Diagram.autoArrangeAvoidContainers Property

Gets or sets a value indicating whether auto-arranged link labels should avoid containers.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get autoArrangeAvoidContainers() {}
set autoArrangeAvoidContainers(value) {}
```

Property Value

Boolean

true to avoid containers, or **false** otherwise. The default value is **true**.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.autoArrangeAvoidSegments Property

Gets or sets a value indicating whether auto-arranged link labels should avoid link segments.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get autoArrangeAvoidSegments() {}
set autoArrangeAvoidSegments(value) {}
```

Property Value

Boolean

true to avoid link segments, or **false** otherwise. The default value is **true**.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.autoCloseDistance Property

Gets or sets the maximum distance between first and last points of a FreeFormNode for which the node's outline is closed automatically.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get autoCloseDistance() {}
set autoCloseDistance(value) {}
```

Property Value

Number

A number specifying auto-close distance.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.autoDeleteLinks Property

Gets or sets whether links are deleted automatically when their origin or destination is deleted.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get autoDeleteLinks() {}
set autoDeleteLinks(value) {}
```

Property Value

Boolean

true to delete links along with the nodes they connect, or **false** otherwise. The default value is **true**.

Remarks

If this property is set to **false**, deleting a node will leave its links in the diagram as unconnected links.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
delKeyAction Property

Diagram.autoResize Property

Gets or sets a value indicating whether the document area should be resized automatically so it fits the diagram contents.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get autoResize() {}
set autoResize(value) {}
```

Property Value

AutoResize

A member of the AutoResize enumeration indicating the auto resize method. The default is **RightAndDown**.

Remarks

If automatic resizing is enabled, the size of the document scrollable area is updated each time an item is created, deleted or moved by the end-user. To resize the diagram after programmatic changes, call the resizeToFitItems method.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
bounds Property
autoScroll Property

Diagram.backBrush Property

Gets or sets an object specifying how to fill the diagram background.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get backBrush() {}
set backBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.backgroundImageAlign Property

Gets or sets a value indicating how to align the background image.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get backgroundImageAlign() {}
set backgroundImageAlign(value) {}
```

Property Value

ImageAlign

A member of the ImageAlign enumeration.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
backgroundImageUrl Property

Diagram.backgroundImageContent Property

Gets or sets a Base64-encoded image to display as diagram background.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get backgroundImageContent() {}
set backgroundImageContent(value) {}
```

Property Value

String

A Base64-encoded representation of an image.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
backgroundImageAlign Property

Diagram.backgroundImageUrl Property

Gets or sets URL of the image to paint as diagram's background.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
```

```
get backgroundImageUrl() {}
set backgroundImageUrl(value) {}
```

Property Value

String

A string containing image URL.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
backBrush Property
backgroundImageAlign Property

Diagram.crossingRadius Property

Gets or sets the radius length of decorations displayed at link intersection points.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get crossingRadius() {}
set crossingRadius(value) {}
```

Property Value

Number

A number specifying the radius of the arcs drawn at link intersection points. The default value is 1.5.

Remarks

If linkCrossings is set to Cut, link segments are cut at intersection points and this property defines the length of the break left between adjacent cut points. If **linkCrossings** is set to **Arcs**, this property defines the radius of the arcs drawn over intersection points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
roundedLinksRadius Property

Diagram.customLinkType Property

Gets or sets the type of DiagramLink objects to create when users starts drawing.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get customLinkType() {}
set customLinkType(value) {}
```

Property Value

Object

An object identifying a constructor function or a class.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.customNodeType Property

Gets or sets the type of DiagramNode objects to create when users starts drawing.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get customNodeType() {}
set customNodeType(value) {}
```

Property Value

Object

An object identifying a constructor function or a class.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.defaultShape Property

Gets or sets the default shape assigned to new ShapeNode objects.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get defaultShape() {}
set defaultShape(value) {}
```

Property Value

Shape | *String*

A Shape object or identifier.

Remarks

The shape property of newly created ShapeNode objects is initialized with the value of this property. It defines a series of lines, arcs and curves composed in a complex shape. There are predefined shapes exposed as static members of the Shapes class. You can also define your own shapes as shown below.

Example

JavaScript

```
var shape = new MindFusion.Diagramming.Shape({
   id: "DoubleEllipse",
   outline: "M100,50 A50,50,0,1,1,0,50 A50,50,0,1,1,100,50",
   decoration: "M98,50 A48,48,0,1,1,2,50 A48,48,0,1,1,98,50",
   fillMode: "nonzero"});

diagram.defaultShape = shape;
```

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.dirty Property

Gets or sets a value indicating whether the diagram has changed since loading it.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get dirty() {}
set dirty(value) {}
```

Property Value

Boolean

true if the diagram has changed, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.disabledHandlesStyle Property

Gets or sets the visual style of disabled adjustment handles.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get disabledHandlesStyle() {}
set disabledHandlesStyle(value) {}
```

Property Value

HandlesVisualStyle

A HandlesVisualStyle instance describing handles' appearance.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
enabledHandles Property
activeItemHandlesStyle Property
selectedItemHandlesStyle Property

Diagram.dynamicLinks Property

Gets or sets initial value of the dynamic property of new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get dynamicLinks() {}
set dynamicLinks(value) {}
```

Property Value

Boolean

true if dynamic links are enabled, or **false** otherwise. The default value is **false**.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.enableSpatialIndex Property

Gets or sets whether to index item positions for faster hit-testing and viewport clipping queries.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get enableSpatialIndex() {}
set enableSpatialIndex(value) {}
```

Property Value

Boolean

true to enable spatial indexing, or **false** otherwise. The default value is **false**.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.expandButtonAction Property

Gets or sets how the control responds to users clickingthe +/- button displayed by expandable nodes.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get expandButtonAction() {}
set expandButtonAction(value) {}
```

Property Value

ExpandButtonAction

A member of the ExpandButtonAction enumeration.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.expandOnIncoming Property

Gets or sets a value indicating the link direction in which tree branches are expanded.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get expandOnIncoming() {}
set expandOnIncoming(value) {}
```

Property Value

Boolean

true if trees are expanded in the direction of incoming links, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.factory Property

Gets a Factory object that lets you programmatically add new items to the diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get factory() {}
```

Property Value

Factory

An instance of the Factory class.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.font Property

Gets or sets the default font of text displayed by diagram items.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get font() {}
set font(value) {}
```

Property Value

Font

A font used to render the text of diagram items.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.freeFormAttractDistance Property

Gets or sets the radius around dragged free-form adjustment handle in which other points of FreeFormNode are modified too.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get freeFormAttractDistance() {}
set freeFormAttractDistance(value) {}
```

Property Value

Number

A number specifying point attraction radius.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.freeFormTargets Property

Gets or sets an array of Shape objects used to replace FreeFormNode instances with ShapeNode ones when behavior is set to LinkFreeShapes or **DrawFreeShapes**.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get freeFormTargets() {}
set freeFormTargets(value) {}
```

Property Value

Array

An array of shape identifiers or Shape instances.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridColor Property

Gets or sets the color of grid points.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get gridColor() {}
set gridColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridOffsetX Property

Gets or sets the horizontal offset of the first point of the alignment grid.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get gridOffsetX() {}
set gridOffsetX(value) {}
```

Property Value

Number

The horizontal offset of the first point of the alignment grid.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridOffsetY Property

Gets or sets the vertical offset of the first point of the alignment grid.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get gridOffsetY() {}
set gridOffsetY(value) {}
```

Property Value

Number

The vertical offset of the first point of the alignment grid.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridPointSize Property

Gets or sets the size of shapes used to represent grid points.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get gridPointSize() {}
set gridPointSize(value) {}
```

Property Value

Number

The size of shapes used to represent grid points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridSizeX Property

Gets or sets the horizontal distance between adjacent grid points.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get gridSizeX() {}
set gridSizeX(value) {}
```

Property Value

Number

The horizontal distance between adjacent grid points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridSizeY Property

Gets or sets the vertical distance between adjacent grid points.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get gridSizeY() {}
set gridSizeY(value) {}
```

Property Value

Number

The vertical distance between adjacent grid points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.gridStyle Property

Gets or sets the visual style of the alignment grid.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get gridStyle() {}
set gridStyle(value) {}
```

Property Value

GridStyle

A member of the GridStyle enumeration.

See Also

Diagram.items Property

Returns an array of all items in this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get items() {}
```

Property Value

Array

Array of DiagramItem objects.

Remarks

This collection gives you access to all items contained in this diagram. You cannot add or remove items directly through it, but must use the addItem and removeItem methods instead.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
enableSpatialIndex Property

Diagram.laneGrid Property

Gets the lane grid.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get laneGrid() {}
```

Property Value

Lanes.Grid

An instance of the Lanes.Grid class.

See Also

Diagram Members
Diagram Class

Diagram.linkBaseShape Property

Gets or sets the shape displayed at the base of new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkBaseShape() {}
set linkBaseShape(value) {}
```

Property Value

Shape | *String*

A Shape instance or identifier that represents an arrowhead shape.

Remarks

This property specifies the default shape displayed as arrowhead at the links' origin points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
ArrowHeads Class
baseShape Property

Diagram.linkBaseShapeSize Property

Gets or sets the default size of arrowheads at base of new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkBaseShapeSize() {}
set linkBaseShapeSize(value) {}
```

Property Value

Number

A number specifying arrowhead size. The default value is 5.

See Also

Diagram Members
Diagram Class

Diagram.linkBrush Property

Gets or sets the default fill of link arrowheads.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkBrush() {}
set linkBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

The property provides a default value for links' brush, used to fill the interior of arrowheads.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.linkCrossings Property

Gets or sets a value indicating how to render the intersection points of links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkCrossings() {}
set linkCrossings(value) {}
```

Property Value

LinkCrossings

A member of the LinkCrossings enumeration. The default value is **Straight**.

Remarks

Intersection points where links cross each other's paths can be rendered as arcs or cuts, depending on the value of this property. The diagram z-order defines which links jump over the others or which links are cut by the others. If **linkCrossings** is set to Arcs, links with higher zIndex jump over those with lower

zIndex. If set to **Cut**, segments of links at lower z-position are cut by links at higher z-position. Radius of arcs and cuts can be set via the crossingRadius property.

> **Performance Note**
>
> Calculating coordinates for crossing point decorations is CPU intensive operation and is triggered each time link's control points change. If populating large diagrams from data, best enable this property only after creating all links.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
routeLinks Property
roundedLinks Property
validateLinkCrossing Event

Diagram.linkHeadShape Property

Gets or sets the shape displayed as head of new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkHeadShape() {}
set linkHeadShape(value) {}
```

Property Value

Shape | *String*

A Shape instance or identifier that represents an arrowhead shape. The default is Arrow.

Remarks

This property specifies the default shape displayed as arrowhead at the links' end points.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
ArrowHeads Class
headShape Property

Diagram.linkHeadShapeSize Property

Gets or sets the default size of arrowheads.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get linkHeadShapeSize() {}
set linkHeadShapeSize(value) {}
```

Property Value

Number

A number specifying arrowhead size. The default value is 5.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.linkHitDistance Property

Gets or sets a value specifying how far from a link a click is still considered a hit.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get linkHitDistance() {}
set linkHitDistance(value) {}
```

Property Value

Number

A value specifying link hit distance.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.linkPen Property

Gets or sets the default stroke of links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get linkPen() {}
set linkPen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

Remarks

This property specifies the default value for the pen property of new links.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.linkRouter Property

Gets or sets the object used to find paths for auto-routed links when diagram nodes are added or their positions change.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkRouter() {}
set linkRouter(value) {}
```

Property Value

Router

An instance of Router-derived class.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
linkRouted Event

Diagram.links Property

Returns an array of all links in this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get links() {}
set links(value) {}
```

Property Value

Array

Array of DiagramLink objects.

Remarks

You cannot add or remove links directly through this array, but must use the addItem and removeItem methods instead.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.linkSegments Property

Gets or sets the default number of link segments.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkSegments() {}
set linkSegments(value) {}
```

Property Value

Number

The default number of link segments.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.linkShape Property

Gets or sets the initial shape assigned to new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkShape() {}
set linkShape(value) {}
```

Property Value

LinkShape

A member of the LinkShape enumeration. The default is **Polyline**.

Remarks

The specified value is assigned to the shape property of new links.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
roundedLinks Property

Diagram.linkTextStyle Property

Gets or sets the default text placement and orientation of new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get linkTextStyle() {}
set linkTextStyle(value) {}
```

Property Value

LinkTextStyle

A member of the LinkTextStyle enumeration. The default value is **Center**.

Remarks

The value of this property is used to initialize the textStyle property of new links.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.mergeThreshold Property

Gets or sets the maximum distance between adjacent control points of a link at which the respective segments can be merged.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get mergeThreshold() {}
set mergeThreshold(value) {}
```

Property Value

Number

A number specifying the maximum distance. The default is 0.

Remarks

A user may add or remove links' segments interactively when the allowSplitLinks property is enabled. They can drag a link's control point over the next one in order to merge two adjacent segments. Usually the two points must be placed exactly at the same coordinates in order to merge the respective segments of the link. Assign a larger value to this property if you wish to make it easier for users to merge segments by just moving link's control point close to each other.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.nodeEffects Property

Gets the global effects applied to nodes in this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get nodeEffects() {}
```

Property Value

Array

An array of NodeEffect objects.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.nodes Property

Returns an array of all nodes in this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get nodes() {}
```

Property Value

Array

Array of DiagramNode objects.

Remarks

You cannot add or remove nodes directly through this array, but must use the addItem and removeItem methods instead.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.nodesExpandable Property

Gets or sets a value indicating whether newly created nodes are expandable.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get nodesExpandable() {}
set nodesExpandable(value) {}
```

Property Value

Boolean

A boolean value assigned to the expandable property of new nodes.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.roundedLinks Property

Gets or sets a value indicating whether the joints between segments of Polyline and **Cascading** links should be rounded.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get roundedLinks() {}
set roundedLinks(value) {}
```

Property Value

Boolean

true to draw rounded segment joints, or **false** otherwise. The default value is **false**.

Remarks

The value of this property is considered when drawing links whose shape is set to Polyline or **Cascading**. If the property is enabled, the joins between segments of all such links are drawn as arcs, whose radius equals the value of roundedLinksRadius.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
linkCrossings Property

Diagram.roundedLinksRadius Property

Gets or sets the radius of joint arcs of rounded links' segments.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get roundedLinksRadius() {}
set roundedLinksRadius(value) {}
```

Property Value

Number

A number specifying the radius length. The default value is 2.

Remarks

The value of this property is used when roundedLinks is enabled. It specifies the radius of the arcs drawn as joins between link segments.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
crossingRadius Property

Diagram.routeLinks Property

Gets or sets initial value of the autoRoute property of new links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get routeLinks() {}
set routeLinks(value) {}
```

Property Value

Boolean

true to route links, or **false** otherwise. The default value is **false**.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
linkRouter Property
linkCrossings Property
linkRouted Event

Diagram.routeMargin Property

Gets or sets the minimal distance between nodes and auto-routed links.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get routeMargin() {}
set routeMargin(value) {}
```

Property Value

Number

A value specifying distance in the current measureUnit. The default value is 8.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.selectedItemHandlesStyle Property

Gets or sets the visual style of adjustment handles of selected items.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get selectedItemHandlesStyle() {}
set selectedItemHandlesStyle(value) {}
```

Property Value

HandlesVisualStyle

A HandlesVisualStyle instance describing handles' appearance.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
selection Property
activeItemHandlesStyle Property
disabledHandlesStyle Property

Diagram.selection Property

Gets a Selection instance that represents the selection of items in this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get selection() {}
```

Property Value

Selection

An instance of the Selection class representing the current selection.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
activeItem property
allowMultipleSelection Property

Diagram.shadowOffsetX Property

Gets or sets the horizontal offset of items' shadows.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
```

```
get shadowOffsetX() {}
set shadowOffsetX(value) {}
```

Property Value

Number

A number specifying shadows' horizontal offset.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.shadowOffsetY Property

Gets or sets the vertical offset of items' shadows.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get shadowOffsetY() {}
set shadowOffsetY(value) {}
```

Property Value

Number

A number specifying shadows' vertical offset.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.shadowsStyle Property

Gets or sets the style of shadows.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get shadowsStyle() {}
set shadowsStyle(value) {}
```

Property Value

ShadowsStyle

A member of the ShadowsStyle enumeration.

Remarks

This property specifies how the shadows of diagram items are drawn. If its value is None, there are no shadows painted. **OneLevel** makes all shadows paint on the lowest z-level, not covering any items. Thus the diagram items will seem to appear on the same z-level. **ZOrder** makes the shadow paint immediately under their items, possibly covering other items. This style gives more 3D-like outlook to the diagram.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.shapeBrush Property

Gets or sets the default fill of ShapeNode objects.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get shapeBrush() {}
set shapeBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

This property provides a default value for the brush used to fill ShapeNode objects.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.shapeLibraryLocation Property

Gets or sets the location of a shape library file containing custom shape definitions. It can be set to a full URL of the file, or a relative file path.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get shapeLibraryLocation() {}
set shapeLibraryLocation(value) {}
```

Property Value

String

The location of a shape library file containing custom shape definitions.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.shapePen Property

Gets or sets the default stroke of ShapeNode objects.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get shapePen() {}
set shapePen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

Remarks

This property provides a default value for the pen used to stroke ShapeNode border lines.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.showAnchors Property

Gets or sets a value indicating when to display anchor point marks.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get showAnchors() {}
set showAnchors(value) {}
```

Property Value

ShowAnchors

A member of the ShowAnchors enumeration.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.showDisabledHandles Property

Gets or sets a value indicating whether adjustment handles are painted when they are disabled.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get showDisabledHandles() {}
set showDisabledHandles(value) {}
```

Property Value

Boolean

true to display disabled handles, or **false** otherwise. The default value is **true**.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.showGrid Property

Gets or sets a value indicating whether the alignment grid is visible.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get showGrid() {}
set showGrid(value) {}
```

Property Value

Boolean

true if the alignment grid is visible, or **false** otherwise.

Remarks

If it is visible, the grid is painted as either a matrix of points or crossing horizontal and vertical lines. That depends on the value of gridStyle.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
alignToGrid Property
gridStyle Property
gridSizeX Property
gridSizeY Property

Diagram.showLaneGrid Property

Gets or sets a value indicating whether the lane grid is visible.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get showLaneGrid() {}
set showLaneGrid(value) {}
```

Property Value

Boolean

true if the lane grid is visible, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace
laneGrid Property

Diagram.style Property

Gets or sets the style associated with this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get style() {}
set style(value) {}
```

Property Value

Style

An instance of the Style class.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.tag Property

Gets or sets custom data associated with this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get tag() {}
set tag(value) {}
```

Property Value

Object

An object containing custom data.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.theme Property

Gets or sets the theme associated with this diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get theme() {}
set theme(value) {}
```

Property Value

Theme

An instance of the Theme class.

See Also

Diagram Members
Diagram Class

Diagram.undoEnabled Property

Gets or sets a value indicating whether saving action records for later undo is enabled.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get undoEnabled() {}
set undoEnabled(value) {}
```

Property Value

Boolean

true if undo is enabled, or **false** otherwise.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram.validityChecks Property

Gets or sets a value indicating whether validity checks are performed each time an item is added to the diagram.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
get validityChecks() {}
set validityChecks(value) {}
```

Property Value

Boolean

true to check validity when adding items, or **false** otherwise. The default is **true**.

Remarks

If enabled, validity checks are performed each time an item is added to the diagram. That involves enumerating the items collection and can slow up the process considerably for large diagrams. You can disable this property in order to skip the validity checks, however be sure that you don't add an item twice to the diagram and that links are created only between items in the same diagram.

See Also

Diagram.version Property

Gets a string identifying current version of the library.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get version() {}
set version(value) {}
```

Property Value

String

A string in "major.minor.revision" format.

See Also

Diagram Members
Diagram Class
MindFusion.Diagramming Namespace

Diagram Events

DiagramBase.actionRedone Event

Raised when an action is redone.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
diagram.actionRedone.addEventListener(onActionRedone);

function onActionRedone(sender, args)
{
}
```

Event Data

Event handlers receive an UndoEventArgs instance containing data about this event.

Remarks

This event is raised during calls to redo. It is raised both for top-level commands and for the constituent actions of composite commands.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
actionUndone Event

DiagramBase.actionUndone Event

Raised when an action is undone.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.actionUndone.addEventListener(onActionUndone);

function onActionUndone(sender, args)
{
}
```

Event Data

Event handlers receive an UndoEventArgs instance containing data about this event.

Remarks

This event is raised during calls to undo. It is raised both for top-level commands and for the constituent actions of composite commands.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
actionRedone Event

DiagramBase.animatedLayoutCompleted Event

Raised when an animated layout completes.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.animatedLayoutCompleted.addEventListener(onAnimatedLayoutCompleted);

function onAnimatedLayoutCompleted(sender, args)
```

```
{
}
```

Event Data

When layout is applied on a ContainerNode, event handlers receive a NodeEventArgs instance as argument.

Example

This example handles the **animatedLayoutCompleted** event to resize the diagram.

JavaScript

```
diagram.animatedLayoutCompleted.addEventListener(
    (sender, args) =>
    {
        sender.resizeToFitItems(10);
    });
diagram.arrangeAnimated(...);
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
arrangeAnimated Method

DiagramBase.cellTextEdited Event

Raised when the user edits the text of a table cell.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a CellEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramBase.cellTextEditing Event

Raised when the user starts to edit the text of a table cell.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a CellEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramBase.clicked Event

Raised when the user clicks the diagram at a point where no items are located.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a DiagramEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
linkClicked Event
nodeClicked Event

DiagramBase.containerChildAdded Event

Raised when the user adds a node to a ContainerNode.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.containerChildAdding Event

A validation event raised to let you prevent users from adding child nodes to a container. Event handlers receive a NodeEventArgs instance that contains data about this event.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.containerChildRemoved Event

Raised when the user removes a node from a ContainerNode.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.containerChildRemoving Event

Raised when the user removes a node from a ContainerNode.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.containerFolded Event

Raised when the user folds a container by clicking the arrow-up button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
ContainerNode.foldable Property

DiagramBase.containerPageChanged Event

Raised when the user changes current page of a PagedContainerNode.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.containerUnfolded Event

Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
ContainerNode.foldable Property

DiagramBase.dataLoaded Event

Raised when the diagram data is loaded.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
diagram.dataLoaded.addEventListener(onDataLoaded);

function onDataLoaded(sender, args)
{
}
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.deserializeTag Event

Raised when the Tag or Id of a diagram item must be deserialized from XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a SerializeTagEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.drawAdjustmentHandles Event

Raised to draw custom adjustment handles.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
diagram.drawAdjustmentHandles.addEventListener(onDrawAdjustmentHandles);

function onDrawAdjustmentHandles(sender, args)
{
}
```

Event Data

Event handlers receive a NodeEventArgs or LinkEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
DiagramNode.handlesStyle Property

DiagramBase.drawAnchorPoint Event

Raised when an anchor point must be custom-drawn.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.drawAnchorPoint.addEventListener(onDrawAnchorPoint);

function onDrawAnchorPoint(sender, args)
{
}
```

Event Data

Event handlers receive a DrawAnchorPointEventArgs instance that contains data about this event.

Remarks

Set the markStyle property of AnchorPoint to Custom to get this event raised.

Example

JavaScript

```
function onDrawAnchorPoint(sender, e)
{
    var p = e.location;
    var ctx = e.context;
    ctx.fillStyle = "red";
    ctx.fillRect(p.x - 1, p.y - 1, 2, 2);
}
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.expandButtonClicked Event

Raised if expandButtonAction is set to RaiseEvents and the [+/-]button of an expandable node is clicked.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
diagram.expandButtonClicked.addEventListener(onExpandButtonClicked);

function onExpandButtonClicked(sender, args)
{
}
```

Event Data

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.headerResized Event

Raised when the user finished resizing a header.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a HeaderResizeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.headerStartResizing Event

Raised when the user starts resizing a header.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a HeaderResizeEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.hitTestAdjustmentHandles Event

Raised to hit-test custom adjustment handles.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.hitTestAdjustmentHandles.addEventListener(onHitTestAdjustmentHandles);

function onHitTestAdjustmentHandles(sender, args)
{
}
```

Event Data

Event handlers receive a NodeEventArgs or LinkEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
DiagramNode.handlesStyle Property

DiagramBase.hyperlinkClicked Event

Raised when the user clicks a hyperlink inside a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.hyperlinkClicked.addEventListener(onHyperlinkClicked);

function onHyperlinkClicked(sender, args)
{
}
```

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.initializeLink Event

Raised when the user starts drawing a new link, just after the DiagramLink instance is created.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.initializeNode Event

Raised when the user starts drawing a new node, just after the DiagramNode instance is created.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.initializeNode.addEventListener(onInitializeNode);

function onInitializeNode(sender, args)
{
}
```

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.itemAdded Event

Raised when an item is added to the diagram.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.itemAdded.addEventListener(onItemAdded);

function onItemAdded(sender, args)
{
}
```

Event Data

Event handlers receive an ItemEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.itemRemoved Event

Raised when an item is removed from the diagram.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.itemRemoved.addEventListener(onItemRemoved);

function onItemRemoved(sender, args)
{
}
```

Event Data

Event handlers receive an ItemEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.laneGridCellTextEdited Event

Raised when the user edits the text of a grid cell.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a CellEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkClicked Event

Raised when the user clicks a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
linkHovered Event
linkDoubleClicked Event

DiagramBase.linkCreated Event

Raised when the user draws a new link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkCreating Event

A validation event raised while the user is drawing a new link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

Example

This example shows how to handle the **linkCreating** event to prevent cycles in the graph.

JavaScript

```
// assuming namespaced scripts
var PathFinder = MindFusion.Diagramming.PathFinder;

diagram.linkCreating.addEventListener(onLinkCreating);

function onLinkCreating(diagram, args)
{
    if (args.destination == null)
    {
        // not pointing to a node yet
        return;
    }

    var pathFinder = new PathFinder(diagram);
    var path = pathFinder.findShortestPath(
        args.destination, args.origin);

    if (path != null)
    {
        // adding this new link would create a cycle
        // [origin]--[dest]--[path internal nodes]--[origin]

        args.cancel = true;
    }
}
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowSelfLoops Property
allowUnconnectedLinks Property

DiagramBase.linkDeleted Event

Raised when a link is deleted, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkDeleting Event

Raised when the user tries to delete a link. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkDeselected Event

Raised when a link is deselected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkDoubleClicked Event

Raised when the user double-clicks a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkHovered Event

Raised when the mouse pointer hovers over a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

Remarks

This event fires when the mouse pointer rests over a link for hoverDelay milliseconds.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
linkClicked Event
linkPointerEnter Event
linkPointerLeave Event

DiagramBase.linkModified Event

Raised when the user moves any control point of a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkModifying Event

A validation event raised while the user is modifying a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkPasted Event

Raised when a link has been pasted from the clipboard.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.linkPasted.addEventListener(onLinkPasted);

function onLinkPasted(sender, args)
{
}
```

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkPointerEnter Event

Raised when the mouse pointer enters the boundaries of a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
linkHovered Event
linkPointerLeave Event

DiagramBase.linkPointerLeave Event

Raised when the mouse pointer leaves the boundaries of a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
linkHovered Event
linkPointerEnter Event

DiagramBase.linkRouted Event

Raised when a link is routed by the auto-routing algorithm.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.linkRouted.addEventListener(onLinkRouted);

function onLinkRouted(sender, args)
{
}
```

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkSelected Event

Raised when a link is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkSelecting Event

Raised when the user tries to select a link. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.linkSplit Event

Raised when the user splits a link interactively.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.linkSplit.addEventListener(onLinkSplit);

function onLinkSplit(sender, args)
{
}
```

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowSplitLinks Property

DiagramBase.linkTextEdited Event

Raised when the user edits the text of a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramBase.linkTextEditing Event

Raised when the user starts to edit the text of a link.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a LinkEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeClicked Event

Raised when the user clicks a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
nodeDoubleClicked Event
nodeHovered Event

DiagramBase.nodeCreated Event

Raised when the user draws a new node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

Example

This example handles the **nodeCreated** event to set new node's anchorPattern.

JavaScript

```
diagram.nodeCreated.addEventListener(
    (sender, args) =>
    {
        if (sender.defaultShape.id == "Rectangle")
            args.node.anchorPattern = pattern1;
        else
            args.node.anchorPattern = pattern2;
    });
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeCreating Event

A validation event raised while the user is drawing a new node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeDeleted Event

Raised when a node is deleted, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeDeleting Event

Raised when the user tries to delete a node. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeDeselected Event

Raised when a node is deselected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeDomCreated Event

Raised when the content of a ControlNode is added to the page DOM.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeDoubleClicked Event

Raised when the user double-clicks a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeHovered Event

Raised when the mouse pointer hovers over a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

Remarks

This event fires when the mouse pointer rests over a node for hoverDelay milliseconds.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeModified Event

Raised when the user moves or resizes a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
nodeClicked Event
nodePointerEnter Event
nodePointerLeave Event

DiagramBase.nodeModifying Event

A validation event raised while the user is moving or resizing a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodePasted Event

Raised when a node has been pasted from the clipboard.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.nodePasted.addEventListener(onNodePasted);

function onNodePasted(sender, args)
{
}
```

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodePointerEnter Event

Raised when the mouse pointer enters the boundaries of a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
nodeHovered Event
nodePointerLeave Event

DiagramBase.nodePointerLeave Event

Raised when the mouse pointer leaves the boundaries of a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeSelected Event

Raised when a node is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeSelecting Event

Raised when the user tries to select a node. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.nodeTextEdited Event

Raised when the user edits the text of a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramBase.nodeTextEditing Event

Raised when the user starts to edit the text of a node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.queryView Event

Raised to enumerate diagram views.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.queryView.addEventListener(onQueryView);

function onQueryView(sender, args)
{
}
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.selectionChanged Event

Raised when the diagram selection has changed.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a SelectionChangedEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.selectionModifying Event

A validation event raised while the user is moving or resizing multiple items.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a SelectionEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.serializeTag Event

Raised when the Tag or Id of a diagram item must be serialized in XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a SerializeTagEventArgs instance that contains data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.shapeLibraryLoaded Event

Raised when a ShapeLibrary is loaded from shapeLibraryLocation.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.sizeChanged Event

Raised when the size of the diagram canvas has changed.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.sizeChanged.addEventListener(onSizeChanged);

function onSizeChanged(sender, args)
{
}
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.tableColumnResized Event

Raised when the user finishes resizing a column in a table.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.tableColumnResized.addEventListener(onTableColumnResized);
```

```
function onTableColumnResized(sender, args)
{
}
```

Event Data

Event handlers receive a CellEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowResizeColumns Property

DiagramBase.tableColumnResizing Event

Raised when the user resizes a column in a table interactively.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.tableColumnResizing.addEventListener(onTableColumnResizing);

function onTableColumnResizing(sender, args)
{
}
```

Event Data

Event handlers receive a CellEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowResizeColumns Property

DiagramBase.tableRowResized Event

Raised when the user finishes resizing a row in a table.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.tableRowResized.addEventListener(onTableRowResized);

function onTableRowResized(sender, args)
{
}
```

Event Data

Event handlers receive a CellEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowResizeRows Property

DiagramBase.tableRowResizing Event

Raised when the user resizes a row in a table interactively.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.tableRowResizing.addEventListener(onTableRowResizing);

function onTableRowResizing(sender, args)
{
}
```

Event Data

Event handlers receive a CellEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowResizeRows Property

DiagramBase.tableSectionCollapsed Event

Raised when a table section is collapsed.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.tableSectionCollapsed.addEventListener(onTableSectionCollapsed);

function onTableSectionCollapsed(sender, args)
{
}
```

Event Data

Event handlers receive a CellEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
TableRow.header Property

DiagramBase.tableSectionExpanded Event

Raised when a table section is expanded.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.tableSectionExpanded.addEventListener(onTableSectionExpanded);

function onTableSectionExpanded(sender, args)
{
}
```

Event Data

Event handlers receive a CellEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
TableRow.header Property

DiagramBase.treeCollapsed Event

Raised when a user collapses a tree branch by clicking the [-] button of an expandable node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.treeExpanded Event

Raised when a user expands a tree branch by clicking the [+] button of an expandable node.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a NodeEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.treeItemTextEdited Event

Raised when the user edits the text of a tree view item.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a TreeItemEventArgs instance containing data about this event.

Example

This example handles the **treeItemTextEdited** event to remove items without text.

JavaScript

```
diagram.treeItemTextEdited.addEventListener(
    (sender, args) =>
    {
        if (!args.newText)
            args.node.removeItem(args.item);
    });
```

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramBase.treeItemTextEditing Event

Raised when the user starts to edit the text of a tree view item.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

Event Data

Event handlers receive a TreeItemEventArgs instance containing data about this event.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramBase.validateAnchorPoint Event

A validation event that lets you approve or reject connecting a link to an anchor point.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.validateAnchorPoint.addEventListener(onValidateAnchorPoint);

function onValidateAnchorPoint(sender, args)
{
}
```

Event Data

Event handlers receive a LinkValidationEventArgs instance containing data about this event.

Remarks

This event is raised for each point of an AnchorPattern while a link's end point is being moved over a node. If none of the anchor points are validated, the mouse cursor icon changes to *NotAllowed*, indicating that the link cannot be connected to that node.

When the mouse button is released over a node, the event is raised again for the node's anchor points. From all validated anchor points, the one nearest to the dragged link end point is used to dock the link.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.validateLinkCrossing Event

A validation event that lets you approve or reject displaying a decoration at the intersection point of links.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
diagram.validateLinkCrossing.addEventListener(onValidateLinkCrossing);

function onValidateLinkCrossing(sender, args)
{
}
```

Event Data

Event handlers receive a LinkCrossingEventArgs instance containing data about this event.

Remarks

This event is raised for each intersection point of links when linkCrossings property is enabled.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase Class

The base class of the Diagram and DiagramDocument classes.

Syntax

JavaScript

```
// class
DiagramBase.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.DiagramBase
MindFusion.Diagramming.Diagram
MindFusion.Diagramming.DiagramDocument

See Also

DiagramBase Members
MindFusion.Diagramming Namespace

DiagramBase Members

The base class of the Diagram and DiagramDocument classes.

The following tables list the members exposed by the DiagramBase type.

Public Constructors

	Name	Description
	DiagramBase	Initializes a new instance of the DiagramBase class.

Public Methods

	Name	Description
	raiseClicked	Raises an event which notifies that a Diagram item has been clicked.
	raiseCreated	Raises an event which notifies that a Diagram item has been created.
	raiseDeleted	Raises an event which notifies that a Diagram item has been deleted.
	raiseDeselected	Raises an event which notifies that a Diagram item has been deselected.
	raiseDoubleClicked	Raises an event which notifies that a Diagram item has been double-clicked.
	raiseModified	Raises an event which notifies that a Diagram item has been modified.
	raisePasted	Raises an event which notifies that a Diagram item has been pasted from clipboard.
	raiseSelected	Raises an event which notifies that a Diagram item has been selected.
	raiseSelecting	Raises an event which notifies that a Diagram item will be selected.
	raiseSelectingAsync	Raises an event which notifies that a Diagram item will be selected.
	raiseSelectionChanged	Raises an event which notifies that the selection has changed.

Public Events

	Name	Description
	actionRedone	Raised when an action is redone.
	actionUndone	Raised when an action is undone.
	animatedLayoutCompleted	Raised when an animated layout completes.
	cellTextEdited	Raised when the user edits the text of a table cell.
	cellTextEditing	Raised when the user starts to edit the text of a table cell.
	clicked	Raised when the user clicks the diagram at a point where no items are located.
	containerChildAdded	Raised when the user adds a node to a ContainerNode.
	containerChildAdding	A validation event raised to let you prevent users from adding child nodes to a container. Event handlers receive a NodeEventArgs instance that contains data about this event.
	containerChildRemoved	Raised when the user removes a node from a **ContainerNode**.

	containerChildRemoving	Raised when the user removes a node from a **ContainerNode**.
	containerFolded	Raised when the user folds a container by clicking the arrow-up button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event.
	containerPageChanged	Raised when the user changes current page of a PagedContainerNode.
	containerUnfolded	Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event.
	dataLoaded	Raised when the diagram data is loaded.
	deserializeTag	Raised when the Tag or Id of a diagram item must be deserialized from XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event.
	drawAdjustmentHandles	Raised to draw custom adjustment handles.
	drawAnchorPoint	Raised when an anchor point must be custom-drawn.
	expandButtonClicked	Raised if expandButtonAction is set to RaiseEvents and the [+/-]button of an expandable node is clicked.
	headerResized	Raised when the user finished resizing a header.
	headerStartResizing	Raised when the user starts resizing a header.
	hitTestAdjustmentHandles	Raised to hit-test custom adjustment handles.
	hyperlinkClicked	Raised when the user clicks a hyperlink inside a node.
	initializeLink	Raised when the user starts drawing a new link, just after the DiagramLink instance is created.
	initializeNode	Raised when the user starts drawing a new node, just after the DiagramNode instance is created.
	itemAdded	Raised when an item is added to the diagram.
	itemRemoved	Raised when an item is removed from the diagram.
	laneGridCellTextEdited	Raised when the user edits the text of a grid cell.
	linkClicked	Raised when the user clicks a link.
	linkCreated	Raised when the user draws a new link.
	linkCreating	A validation event raised while the user is drawing a new link.
	linkDeleted	Raised when a link is deleted, either programmatically or by the user.
	linkDeleting	Raised when the user tries to delete a link. This event lets you cancel the operation.
	linkDeselected	Raised when a link is deselected, either programmatically or by the user.

	linkDoubleClicked	Raised when the user double-clicks a link.
	linkHovered	Raised when the mouse pointer hovers over a link.
	linkModified	Raised when the user moves any control point of a link.
	linkModifying	A validation event raised while the user is modifying a link.
	linkPasted	Raised when a link has been pasted from the clipboard.
	linkPointerEnter	Raised when the mouse pointer enters the boundaries of a link.
	linkPointerLeave	Raised when the mouse pointer leaves the boundaries of a link.
	linkRouted	Raised when a link is routed by the auto-routing algorithm.
	linkSelected	Raised when a link is selected, either programmatically or by the user.
	linkSelecting	Raised when the user tries to select a link. This event lets you cancel the operation.
	linkSplit	Raised when the user splits a link interactively.
	linkTextEdited	Raised when the user edits the text of a link.
	linkTextEditing	Raised when the user starts to edit the text of a link.
	nodeClicked	Raised when the user clicks a node.
	nodeCreated	Raised when the user draws a new node.
	nodeCreating	A validation event raised while the user is drawing a new node.
	nodeDeleted	Raised when a node is deleted, either programmatically or by the user.
	nodeDeleting	Raised when the user tries to delete a node. This event lets you cancel the operation.
	nodeDeselected	Raised when a node is deselected, either programmatically or by the user.
	nodeDomCreated	Raised when the content of a ControlNode is added to the page DOM.
	nodeDoubleClicked	Raised when the user double-clicks a node.
	nodeHovered	Raised when the mouse pointer hovers over a node.
	nodeModified	Raised when the user moves or resizes a node.
	nodeModifying	A validation event raised while the user is moving or resizing a node.
	nodePasted	Raised when a node has been pasted from the clipboard.
	nodePointerEnter	Raised when the mouse pointer enters the boundaries of a node.
	nodePointerLeave	Raised when the mouse pointer leaves the boundaries of a node.

	nodeSelected	Raised when a node is selected, either programmatically or by the user.
	nodeSelecting	Raised when the user tries to select a node. This event lets you cancel the operation.
	nodeTextEdited	Raised when the user edits the text of a node.
	nodeTextEditing	Raised when the user starts to edit the text of a node.
	queryView	Raised to enumerate diagram views.
	selectionChanged	Raised when the diagram selection has changed.
	selectionModifying	A validation event raised while the user is moving or resizing multiple items.
	serializeTag	Raised when the Tag or Id of a diagram item must be serialized in XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event.
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from shapeLibraryLocation.
	sizeChanged	Raised when the size of the diagram canvas has changed.
	tableColumnResized	Raised when the user finishes resizing a column in a table.
	tableColumnResizing	Raised when the user resizes a column in a table interactively.
	tableRowResized	Raised when the user finishes resizing a row in a table.
	tableRowResizing	Raised when the user resizes a row in a table interactively.
	tableSectionCollapsed	Raised when a table section is collapsed.
	tableSectionExpanded	Raised when a table section is expanded.
	treeCollapsed	Raised when a user collapses a tree branch by clicking the [-] button of an **expandable** node.
	treeExpanded	Raised when a user expands a tree branch by clicking the [+] button of an **expandable** node.
	treeItemTextEdited	Raised when the user edits the text of a tree view item.
	treeItemTextEditing	Raised when the user starts to edit the text of a tree view item.
	validateAnchorPoint	A validation event that lets you approve or reject connecting a link to an anchor point.
	validateLinkCrossing	A validation event that lets you approve or reject displaying a decoration at the intersection point of links.

See Also

DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase Constructor

Initializes a new instance of the DiagramBase class.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
function DiagramBase ()
```

See Also

DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase Methods

DiagramBase.raiseClicked Method

Raises an event which notifies that a Diagram item has been clicked.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
function raiseClicked (item, mousePosition, mouseButton)
```

Parameters

item
DiagramItem. The Diagram item.

mousePosition
Point. The position of the mouse.

mouseButton
MouseButton. The mouse button, with which the item has been clicked.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseCreated Method

Raises an event which notifies that a Diagram item has been created.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
function raiseCreated (item)
```

Parameters

item
DiagramNode | DiagramLink. The diagram item that has been created.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseDeleted Method

Raises an event which notifies that a Diagram item has been deleted.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
function raiseDeleted (item)
```

Parameters

item
DiagramItem. The DiagramItem that has been deleted.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseDeselected Method

Raises an event which notifies that a Diagram item has been deselected.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
function raiseDeselected (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseDoubleClicked Method

Raises an event which notifies that a Diagram item has been double-clicked.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
function raiseDoubleClicked (item, mousePosition, mouseButton)
```

Parameters

item
DiagramItem. The Diagram item.

mousePosition
Point. The position of the mouse.

mouseButton
MouseButton. The mouse button, with which the item has been clicked.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseModified Method

Raises an event which notifies that a Diagram item has been modified.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
function raiseModified (item, mousePosition, adjustmentHandle)
```

Parameters

item
DiagramItem. The DiagramItem that has been modified.

mousePosition

Point. The current mouse pointer position.

adjustmentHandle
Number. Identifies the handle used to modify the item.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raisePasted Method

Raises an event which notifies that a Diagram item has been pasted from clipboard.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
function raisePasted (item)
```

Parameters

item
DiagramItem. The DiagramItem that has been pasted.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseSelected Method

Raises an event which notifies that a Diagram item has been selected.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

JavaScript

```
function raiseSelected (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

DiagramBase Members
DiagramBase Class

DiagramBase.raiseSelecting Method

Raises an event which notifies that a Diagram item will be selected.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
function raiseSelecting (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseSelectingAsync Method

Raises an event which notifies that a Diagram item will be selected.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
function raiseSelectingAsync (item)
```

Parameters

item
DiagramItem. The Diagram item.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase.raiseSelectionChanged Method

Raises an event which notifies that the selection has changed.

Namespace: MindFusion.Diagramming
File: DiagramBase.js

Syntax

```
JavaScript
function raiseSelectionChanged (oldItems, newItems)
```

Parameters

oldItems
Array. The selected items before the change.

newItems
Array. The selected items after the change.

See Also

DiagramBase Members
DiagramBase Class
MindFusion.Diagramming Namespace

DiagramBase Events

DiagramDocument Class

Represents a collection of diagram pages.

Syntax

```
JavaScript
// class
DiagramDocument.prototype = {}
```

Remarks

The **DiagramDocument** class represents a collection of diagram pages or sheets, represented by DiagramPage objects. New pages can be added to the document and existing pages can be removed and reordered by using the addPage and removePage methods respectively. The toJson, saveToXml and saveToString methods of **DiagramDocument** save all pages in a single file, and respectively the fromJson, loadFromXml and loadFromString methods load all pages from the file. **DiagramDocument** can also load files created by the serialization methods of Diagram, and will show them in a single page.

DiagramPage is derived from **Diagram** and only adds a few properties, so any code that processes **Diagram** objects will work with **DiagramPage** objects too. A diagram document forwards each event raised by its pages, giving you a chance to use the same handler for events raised by different pages. If a page should implement distinct user interaction or validation logic, it is also possible to attach handlers to the **Diagram** events inherited by **DiagramPage**.

A single **DiagramPage** could be assigned to the diagram property of DiagramView, and that lets you create your own user interface for presenting a list of pages to the user and selecting a current page. The package also includes a TabbedDiagramView that presents all pages of a document in a tabbed user interface where each tab corresponds to a page. The page displayed in the active tab is exposed by the selectedPage property of **TabbedDiagramView**. The active page can be changed interactively by activating its associated tab. New pages can be added and removed to / from the document through buttons in the tab strip and the pages can be rearranged by dragging their associated tabs with the mouse.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramBase
MindFusion.Diagramming.DiagramDocument

See Also

DiagramDocument Members
MindFusion.Diagramming Namespace

DiagramDocument Members

Represents a collection of diagram pages.

The following tables list the members exposed by the DiagramDocument type.

Public Constructors

	Name	Description
	DiagramDocument	Initializes a new instance of the DiagramDocument class.

Public Properties

	Name	Description
	dirty	Gets or sets a value indicating whether the diagram document has changed since loading it.
	pages	Gets a list of the diagram pages contained in this document.

Public Methods

	Name	Description
	addPage	Adds a page to the document.
	fromJson	Deserializes the document from a JSON string.
	fromXmlDocument	Loads document contents from specified XML Document.
	loadFromString	Loads document contents from specified XML or JSON string.
	loadFromXml	Loads the document from an XML file.
	raiseClicked	Raises an event which notifies that a Diagram item has been clicked. (Inherited from DiagramBase.)
	raiseCreated	Raises an event which notifies that a Diagram item has been created. (Inherited from DiagramBase.)
	raiseDeleted	Raises an event which notifies that a Diagram item has been deleted. (Inherited from DiagramBase.)
	raiseDeselected	Raises an event which notifies that a Diagram item has been deselected. (Inherited from DiagramBase.)

	raiseDoubleClicked	Raises an event which notifies that a Diagram item has been double-clicked. (Inherited from DiagramBase.)
	raiseModified	Raises an event which notifies that a Diagram item has been modified. (Inherited from DiagramBase.)
	raisePasted	Raises an event which notifies that a Diagram item has been pasted from clipboard. (Inherited from DiagramBase.)
	raiseSelected	Raises an event which notifies that a Diagram item has been selected. (Inherited from DiagramBase.)
	raiseSelecting	Raises an event which notifies that a Diagram item will be selected. (Inherited from DiagramBase.)
	raiseSelectingAsync	Raises an event which notifies that a Diagram item will be selected. (Inherited from DiagramBase.)
	raiseSelectionChanged	Raises an event which notifies that the selection has changed. (Inherited from DiagramBase.)
	removePage	Removes the specified page from the document.
	saveToString	Encodes the document contents into an XML or JSON string.
	saveToXml	Saves the document to an XML file.
	toJson	Serializes the document into a JSON string.
	toXmlDocument	Saves the document into an XML Document.

Public Events

	Name	Description
	actionRedone	Raised when an action is redone. (Inherited from DiagramBase.)
	actionUndone	Raised when an action is undone. (Inherited from DiagramBase.)
	animatedLayoutCompleted	Raised when an animated layout completes. (Inherited from DiagramBase.)
	cellTextEdited	Raised when the user edits the text of a table cell. (Inherited from DiagramBase.)
	cellTextEditing	Raised when the user starts to edit the text of a table cell. (Inherited from DiagramBase.)
	clicked	Raised when the user clicks the diagram at a point where no items are located. (Inherited from DiagramBase.)
	containerChildAdded	Raised when the user adds a node to a ContainerNode. (Inherited from DiagramBase.)
	containerChildAdding	A validation event raised to let you prevent users from adding child nodes to a container. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	containerChildRemoved	Raised when the user removes a node from a **ContainerNode**. (Inherited from DiagramBase.)

	containerChildRemoving	Raised when the user removes a node from a **ContainerNode**. (Inherited from DiagramBase.)
	containerFolded	Raised when the user folds a container by clicking the arrow-up button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	containerPageChanged	Raised when the user changes current page of a PagedContainerNode. (Inherited from DiagramBase.)
	containerUnfolded	Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	dataLoaded	Raised when the diagram data is loaded. (Inherited from DiagramBase.)
	deserializeTag	Raised when the Tag or Id of a diagram item must be deserialized from XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	drawAdjustmentHandles	Raised to draw custom adjustment handles. (Inherited from DiagramBase.)
	drawAnchorPoint	Raised when an anchor point must be custom-drawn. (Inherited from DiagramBase.)
	expandButtonClicked	Raised if expandButtonAction is set to RaiseEvents and the [+/-]button of an expandable node is clicked. (Inherited from DiagramBase.)
	headerResized	Raised when the user finished resizing a header. (Inherited from DiagramBase.)
	headerStartResizing	Raised when the user starts resizing a header. (Inherited from DiagramBase.)
	hitTestAdjustmentHandles	Raised to hit-test custom adjustment handles. (Inherited from DiagramBase.)
	hyperlinkClicked	Raised when the user clicks a hyperlink inside a node. (Inherited from DiagramBase.)
	initializeLink	Raised when the user starts drawing a new link, just after the DiagramLink instance is created. (Inherited from DiagramBase.)
	initializeNode	Raised when the user starts drawing a new node, just after the DiagramNode instance is created. (Inherited from DiagramBase.)
	itemAdded	Raised when an item is added to the diagram. (Inherited from DiagramBase.)
	itemRemoved	Raised when an item is removed from the diagram. (Inherited from DiagramBase.)
	laneGridCellTextEdited	Raised when the user edits the text of a grid cell. (Inherited from DiagramBase.)

	linkClicked	Raised when the user clicks a link. (Inherited from DiagramBase.)
	linkCreated	Raised when the user draws a new link. (Inherited from DiagramBase.)
	linkCreating	A validation event raised while the user is drawing a new link. (Inherited from DiagramBase.)
	linkDeleted	Raised when a link is deleted, either programmatically or by the user. (Inherited from DiagramBase.)
	linkDeleting	Raised when the user tries to delete a link. This event lets you cancel the operation. (Inherited from DiagramBase.)
	linkDeselected	Raised when a link is deselected, either programmatically or by the user. (Inherited from DiagramBase.)
	linkDoubleClicked	Raised when the user double-clicks a link. (Inherited from DiagramBase.)
	linkHovered	Raised when the mouse pointer hovers over a link. (Inherited from DiagramBase.)
	linkModified	Raised when the user moves any control point of a link. (Inherited from DiagramBase.)
	linkModifying	A validation event raised while the user is modifying a link. (Inherited from DiagramBase.)
	linkPasted	Raised when a link has been pasted from the clipboard. (Inherited from DiagramBase.)
	linkPointerEnter	Raised when the mouse pointer enters the boundaries of a link. (Inherited from DiagramBase.)
	linkPointerLeave	Raised when the mouse pointer leaves the boundaries of a link. (Inherited from DiagramBase.)
	linkRouted	Raised when a link is routed by the auto-routing algorithm. (Inherited from DiagramBase.)
	linkSelected	Raised when a link is selected, either programmatically or by the user. (Inherited from DiagramBase.)
	linkSelecting	Raised when the user tries to select a link. This event lets you cancel the operation. (Inherited from DiagramBase.)
	linkSplit	Raised when the user splits a link interactively. (Inherited from DiagramBase.)
	linkTextEdited	Raised when the user edits the text of a link. (Inherited from DiagramBase.)
	linkTextEditing	Raised when the user starts to edit the text of a link. (Inherited from DiagramBase.)
	nodeClicked	Raised when the user clicks a node. (Inherited from DiagramBase.)
	nodeCreated	Raised when the user draws a new node. (Inherited from DiagramBase.)

	nodeCreating	A validation event raised while the user is drawing a new node. (Inherited from DiagramBase.)
	nodeDeleted	Raised when a node is deleted, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeDeleting	Raised when the user tries to delete a node. This event lets you cancel the operation. (Inherited from DiagramBase.)
	nodeDeselected	Raised when a node is deselected, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeDomCreated	Raised when the content of a ControlNode is added to the page DOM. (Inherited from DiagramBase.)
	nodeDoubleClicked	Raised when the user double-clicks a node. (Inherited from DiagramBase.)
	nodeHovered	Raised when the mouse pointer hovers over a node. (Inherited from DiagramBase.)
	nodeModified	Raised when the user moves or resizes a node. (Inherited from DiagramBase.)
	nodeModifying	A validation event raised while the user is moving or resizing a node. (Inherited from DiagramBase.)
	nodePasted	Raised when a node has been pasted from the clipboard. (Inherited from DiagramBase.)
	nodePointerEnter	Raised when the mouse pointer enters the boundaries of a node. (Inherited from DiagramBase.)
	nodePointerLeave	Raised when the mouse pointer leaves the boundaries of a node. (Inherited from DiagramBase.)
	nodeSelected	Raised when a node is selected, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeSelecting	Raised when the user tries to select a node. This event lets you cancel the operation. (Inherited from DiagramBase.)
	nodeTextEdited	Raised when the user edits the text of a node. (Inherited from DiagramBase.)
	nodeTextEditing	Raised when the user starts to edit the text of a node. (Inherited from DiagramBase.)
	queryView	Raised to enumerate diagram views. (Inherited from DiagramBase.)
	selectionChanged	Raised when the diagram selection has changed. (Inherited from DiagramBase.)
	selectionModifying	A validation event raised while the user is moving or resizing multiple items. (Inherited from DiagramBase.)
	serializeTag	Raised when the Tag or Id of a diagram item must be serialized in XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from shapeLibraryLocation. (Inherited from DiagramBase.)

	sizeChanged	Raised when the size of the diagram canvas has changed. (Inherited from DiagramBase.)
	tableColumnResized	Raised when the user finishes resizing a column in a table. (Inherited from DiagramBase.)
	tableColumnResizing	Raised when the user resizes a column in a table interactively. (Inherited from DiagramBase.)
	tableRowResized	Raised when the user finishes resizing a row in a table. (Inherited from DiagramBase.)
	tableRowResizing	Raised when the user resizes a row in a table interactively. (Inherited from DiagramBase.)
	tableSectionCollapsed	Raised when a table section is collapsed. (Inherited from DiagramBase.)
	tableSectionExpanded	Raised when a table section is expanded. (Inherited from DiagramBase.)
	treeCollapsed	Raised when a user collapses a tree branch by clicking the [-] button of an **expandable** node. (Inherited from DiagramBase.)
	treeExpanded	Raised when a user expands a tree branch by clicking the [+] button of an **expandable** node. (Inherited from DiagramBase.)
	treeItemTextEdited	Raised when the user edits the text of a tree view item. (Inherited from DiagramBase.)
	treeItemTextEditing	Raised when the user starts to edit the text of a tree view item. (Inherited from DiagramBase.)
	validateAnchorPoint	A validation event that lets you approve or reject connecting a link to an anchor point. (Inherited from DiagramBase.)
	validateLinkCrossing	A validation event that lets you approve or reject displaying a decoration at the intersection point of links. (Inherited from DiagramBase.)

See Also

DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument Constructor

Initializes a new instance of the DiagramDocument class.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function DiagramDocument ()
```

See Also

DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument Methods

DiagramDocument.addPage Method

Adds a page to the document.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

```
JavaScript
function addPage (page)
```

Parameters

page
DiagramPage. The page to add.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.fromJson Method

Deserializes the document from a JSON string.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

```
JavaScript
function fromJson (json)
```

Parameters

json
String. A string created by the toJson method.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.fromXmlDocument Method

Loads document contents from specified XML Document.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

```
JavaScript
function fromXmlDocument (document)
```

Parameters

document
Document. A DOM Document object containing XML-serialized DiagramDocument contents.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.loadFromString Method

Loads document contents from specified XML or JSON string.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

```
JavaScript
function loadFromString (str)
```

Parameters

str
String. A string containing the serialized document contents.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.loadFromXml Method

Loads the document from an XML file.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function loadFromXml (fileUrl, [onLoad, [onError]])
```

Parameters

fileUrl
String. The URL of an XML file where the data should be read from.

onLoad
Optional.
function. A callback that should be invoked if the file is loaded successfully.

onError
Optional.
function. A callback that should be invoked if the file could not be downloaded.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.removePage Method

Removes the specified page from the document.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function removePage (page)
```

Parameters

page
DiagramPage. The page to be removed.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.saveToString Method

Encodes the document contents into an XML or JSON string.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function saveToString (format)
```

Parameters

format
SaveToStringFormat. A member of the SaveToStringFormat enumeration.

Return Value

String. A string containing the document's serialized representation.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.saveToXml Method

Saves the document to an XML file.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function saveToXml (url)
```

Parameters

url
String. A URL specifying where the document's XML should be posted to.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.toJson Method

Serializes the document into a JSON string.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function toJson ()
```

Return Value

String. A string containing the document's JSON representation.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument.toXmlDocument Method

Saves the document into an XML Document.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
function toXmlDocument ()
```

Return Value

Document. A DOM Document object containing XML-serialized DiagramDocument contents.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument Properties

DiagramDocument.dirty Property

Gets or sets a value indicating whether the diagram document has changed since loading it.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

JavaScript

```
get dirty() {}
set dirty(value) {}
```

Property Value

Boolean

true if the document has changed, or false otherwise.

See Also

DiagramDocument Members
DiagramDocument Class

DiagramDocument.pages Property

Gets a list of the diagram pages contained in this document.

Namespace: MindFusion.Diagramming
File: DiagramDocument.js

Syntax

```
JavaScript
get pages() {}
set pages(value) {}
```

Property Value

Array

Array of DiagramPage objects.

Remarks

Use the addPage and removePage methods to respectively add or remove pages.

See Also

DiagramDocument Members
DiagramDocument Class
MindFusion.Diagramming Namespace

DiagramDocument Events

DiagramEventArgs Class

Contains the arguments passed to handlers of global diagram events such as clicked.

Syntax

```
JavaScript
// class
DiagramEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.EventArgs
  MindFusion.Diagramming.DiagramEventArgs

See Also

DiagramEventArgs Members

Contains the arguments passed to handlers of global diagram events such as clicked.

The following tables list the members exposed by the DiagramEventArgs type.

Public Constructors

	Name	Description
	DiagramEventArgs	Initializes a new instance of the DiagramEventArgs class.

Public Properties

	Name	Description
	deltaX	Gets the horizontal scroll amount of a wheel event.
	deltaY	Gets the vertical scroll amount of a wheel event.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	mouseButton	Gets which mouse button has been pressed.
	mousePosition	Gets the current mouse pointer position.

See Also

DiagramEventArgs Class
MindFusion.Diagramming Namespace

DiagramEventArgs Constructor

Initializes a new instance of the DiagramEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function DiagramEventArgs ()
```

See Also

DiagramEventArgs Class
MindFusion.Diagramming Namespace

DiagramEventArgs Properties

DiagramEventArgs.deltaX Property

Gets the horizontal scroll amount of a wheel event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get deltaX() {}
set deltaX(value) {}
```

Property Value

Number

The horizontal scroll amount.

See Also

DiagramEventArgs Members
DiagramEventArgs Class
MindFusion.Diagramming Namespace

DiagramEventArgs.deltaY Property

Gets the vertical scroll amount of a wheel event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get deltaY() {}
set deltaY(value) {}
```

Property Value

Number

The vertical scroll amount.

See Also

DiagramEventArgs Members
DiagramEventArgs Class
MindFusion.Diagramming Namespace

DiagramEventArgs.mouseButton Property

Gets which mouse button has been pressed.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get mouseButton() {}
```

Property Value

MouseButton

A member of the MouseButton enumeration.

See Also

DiagramEventArgs Members
DiagramEventArgs Class
MindFusion.Diagramming Namespace

DiagramEventArgs.mousePosition Property

Gets the current mouse pointer position.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get mousePosition() {}
```

Property Value

Point

A Point instance containing mouse pointer coordinates.

See Also

DiagramEventArgs Members
DiagramEventArgs Class
MindFusion.Diagramming Namespace

DiagramItem Class

Represents an item in the diagram model. All classes representing diagram elements derive from **DiagramItem**.

Syntax

```
JavaScript
```

```
// class
DiagramItem.prototype = {}
```

Remarks

All items in the diagram document - links and nodes - share common base class called **DiagramItem**. Its main function is to provide interface for the most common methods and properties of every diagram element.

The main textual content of an item is set using the text property. Supplementary labels can be created by calling addLabel.

Appearance can be customized using a shared Style object assigned to the style property. Attributes of the shared style can be overridden by setting item's properties such as brush, stroke, font.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
MindFusion.Diagramming.DiagramLink
MindFusion.Diagramming.DiagramNode
MindFusion.Diagramming.Selection

See Also

DiagramItem Members
MindFusion.Diagramming Namespace

DiagramItem Members

Represents an item in the diagram model. All classes representing diagram elements derive from DiagramItem.

The following tables list the members exposed by the **DiagramItem** type.

Public Constructors

	Name	Description
	DiagramItem	Initializes a new instance of the **DiagramItem** class.

Public Properties

	Name	Description
	brush	Gets or sets an object specifying how to fill the interior of this item.
	container	Gets the parent ContainerNode containing this node.
	cssClass	Gets or sets the CSS class associated with this item.
	cssElementName	Gets the name of the CSS element associated with this item.
	effectiveBrush	Gets the effective brush that should be used to paint this item.
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar.

	effectiveFont	Gets the effective font that should be used to render this item's text.
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text.
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text.
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text.
	effectiveOpacity	Gets the effective opacity that should be used to paint this item.
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow.
	effectiveStroke	Gets the effective color that should be used to stroke this item.
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item.
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item.
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item.
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item.
	effectiveTextColor	Gets the effective text color that should be used to render this item's text.
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline.
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item.
	font	Gets or sets the font used to render item's text.
	graphicsContent	Gets an array of visual elements used to render this item on the canvas.
	hyperLink	Gets or sets the hyperlink associated with this diagram item.
	id	Gets or sets a custom identifier associated with this item.
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods.
	layoutTraits	Gets an object containing properties specific to individual layout algorithms.
	lineAlignment	Gets or sets the vertical alignment of item's text.
	locked	Gets or sets whether users are allowed to interact with this item.
	opacity	Gets or sets opacity of this item.
	parent	Gets the Diagram to which this item belongs.

	pen	Gets or sets the color used to stroke item's borders.
	selected	Gets or sets a value indicating whether a diagram item is selected.
	shadowColor	Gets or sets the color used to draw item's shadow.
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow.
	shadowOffsetY	Gets or sets the vertical offset of item's shadow.
	stroke	Gets or sets the color used to stroke item's frame.
	strokeDashArray	Gets or sets an array of values specifying the dash pattern.
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern.
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame.
	strokeThickness	Gets or sets the line width applied when stroking item's frame.
	style	Gets or sets the style associated with this item.
	tag	Gets or sets custom data tag associated with this item.
	text	Gets or sets the text label displayed by this item.
	textAlignment	Gets or sets the horizontal alignment of item's text.
	textColor	Gets or sets the color used to fill item's text.
	textComponent	Gets the Text object instance used to render item's text.
	textPadding	Gets or sets the padding distance between item's boundaries and text label.
	textStroke	Gets or sets the color used to stroke item's text.
	textStrokeThickness	Gets or sets the outline width of item's text.
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item.
	topLevel	Gets if this is a top-level item in the diagram.
	visible	Gets or sets a value indicating whether this item is visible.
	weight	Gets or sets a weight value used by layout and path-finding algorithms.
	zIndex	Gets the z-order position of this item.

Public Methods

	Name	Description
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection.
	addLabel	Adds a new label to this item.

	allowDrag	Invoked to validate the drag operation.
	animate	Animates the item.
	clone	Creates a copy of this item.
	containsPoint	Determines whether this item contains the specified point.
	endDrag	Invoked when the user stops dragging this item.
	fromJson	Deserializes this item from a JSON object.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow.
	hitTest	Returns the diagram item that contains the specified point.
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds.
	loadFromXml	Loads the item content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo.
	removeCanvasElements	Derived classes must implement this method by removing their graphics primitives from the canvas.elements collection.
	removeLabel	Removes the specified label from this item.
	restoreState	Loads the properties of this diagram item.
	saveLocationState	Saves the state of this **DiagramItem**.
	saveState	Saves the properties of this diagram item.
	saveToXml	Saves the item content into an XML element.
	startDrag	Invoked when the user starts dragging this item.
	toJson	Serializes this item into a JSON object.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements.
	updateDrag	Invoked while the user drags this item.

See Also

DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem Constructor

Initializes a new instance of the DiagramItem class.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function DiagramItem ([parent])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the item properties.

See Also

DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem Methods

DiagramItem.addCanvasElements Method

Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function addCanvasElements ()
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.addLabel Method

Adds a new label to this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function addLabel (text)
```

Parameters

text
String. The label's text.

Return Value

A LinkLabel or NodeLabel instance, depending on item's type.

Remarks

An item's main text content is specified via its text property. Call **addLabel** if you need to display additional text labels, such as identifiers or hints to the user.

Position of labels can be specified by calling methods such as setCornerPosition for nodes, or setControlPointPosition for links. You can let users move labels interactively by setting the behavior property of DiagramView to MoveLabels.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
ItemLabel Class
removeLabel Method

DiagramItem.allowDrag Method

Invoked to validate the drag operation.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function allowDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.animate Method

Animates the item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function animate (animation)
```

Parameters

animation
ItemAnimation. An **ItemAnimation** -derived object.

Remarks

Item's animated values are considered transient and do not change actual property values. If you want to assign last value from animation to actual property such as bounds or opacity, set the *keepLastValue* flag in animation parameters as in example below.

Example

JavaScript

```
node.animate(
    new PathAnimation(
        [
            new Point(30, 30),
            new Point(50, 30),
            new Point(75, 66),
            new Point(125, 88)
        ],
        {
            keepLastValue: true
        }));
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
Animations

DiagramItem.clone Method

Creates a copy of this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function clone ()
```

Return Value

Object. The copied item.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.containsPoint Method

Determines whether this item contains the specified point.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function containsPoint (point)
```

Parameters

point
The Point to check.

Return Value

Boolean. **true** if this item contains the specified point, or **false** otherwise.

Remarks

This method is used during hit-testing to see if the mouse is inside the item.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.endDrag Method

Invoked when the user stops dragging this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function endDrag (ist)
```

Parameters

ist
InteractionState. An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.fromJson Method

Deserializes this item from a JSON object.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function fromJson (json)
```

Parameters

json
Object. An object, containing the data for this item.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.getRepaintBounds Method

Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function getRepaintBounds ()
```

Return Value

A Rect specifying the region that should be repainted when this item is modified.

Remarks

This method is invoked when an item is modified by the user, or when an appearance property of the item changes. The returned region is invalidated and redrawn at the next paint operation.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
bounds Property

DiagramItem.hitTest Method

Returns the diagram item that contains the specified point.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function hitTest (point, [threshold, [onlyNodes, [onlyInteractive]]])
```

Parameters

point
Point. A Point value specifying the logical coordinates of a diagram point.

threshold
Optional.
Number. Specifies the size of the area around the item outlines where the point is still considered inside the item.

onlyNodes
Optional.
Boolean. true to exclude child links from the search.

onlyInteractive
Optional.
Boolean. true to exclude locked items from the search.

Return Value

DiagramItem. The item that contains the specified point.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.invalidate Method

Invalidates the region of the canvas, that matches the item's repaint bounds.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function invalidate ([dirty, [force]])
```

Parameters

dirty
Optional.
Boolean. Optional.

force
Optional.
Boolean. Optional.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.loadFromXml Method

Loads the item content from an XML element.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that contains the item's serialized content.

context
An XmlPersistContext object providing contextual information about the serialization process and some helper serialization methods.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
saveToXml Method

DiagramItem.onRemove Method

Called when an item is removed from a Diagram, either by the user, programmatically or via undo / redo.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function onRemove ()
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.removeCanvasElements Method

Derived classes must implement this method by removing their graphics primitives from the canvas.elements collection.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function removeCanvasElements ()
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.removeLabel Method

Removes the specified label from this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function removeLabel (label)
```

Parameters

label
An ItemLabel instance.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
addLabel Method

DiagramItem.restoreState Method

Loads the properties of this diagram item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function restoreState (state)
```

Parameters

state

Object. An object, containing the properties of this DiagramItem.

Remarks

This method is used by the undo/redo subsystem.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.saveLocationState Method

Saves the state of this DiagramItem.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function saveLocationState ()
```

Return Value

A DiagramItemState instance.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.saveState Method

Saves the properties of this diagram item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function saveState ()
```

Return Value

Object. An object, containing the properties of this DiagramItem.

Remarks

This method is used by the undo/redo subsystem.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.saveToXml Method

Saves the item content into an XML element.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
loadFromXml Method

DiagramItem.startDrag Method

Invoked when the user starts dragging this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
function startDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramItem Members
DiagramItem Class

DiagramItem.toJson Method

Serializes this item into a JSON object.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function toJson ()
```

Return Value

Object. An object, containing the data for this item.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.updateCanvasElements Method

Derived classes must implement this method to update the item's visual elements.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
function updateCanvasElements ()
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.updateDrag Method

Invoked while the user drags this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
```

```
function updateDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem Properties

DiagramItem.brush Property

Gets or sets an object specifying how to fill the interior of this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

This property can be set to a color string to specify solid fill:

JavaScript

```
node.brush = "white";
```

It also accepts objects that specify the type of brush and additional information:

JavaScript

```
node1.brush =
    { type: 'SolidBrush', color: '#e0e9e9' };
node2.brush =
    { type: 'LinearGradientBrush', color1: '#e0e9e9', color2: '#616a7f', angle: 90 };
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

stroke Property
opacity Property

DiagramItem.container Property

Gets the parent ContainerNode containing this node.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get container() {}
```

Property Value

ContainerNode

The parent ContainerNode of this node.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.cssClass Property

Gets or sets the CSS class associated with this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get cssClass() {}
set cssClass(value) {}
```

Property Value

String

A string containing the CSS class.

Remarks

You could set this value to associate CSS rules with a specific item using class selector.

Example

```
JavaScript
// JavaScript
node.text = "Attention";
```

```
node.cssClass = "urgent";

// CSS
mfd-shapenode.urgent {
    --brush: #ffcccc;
    --stroke: red;
}
```

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
style Property

DiagramItem.cssElementName Property

Gets the name of the CSS element associated with this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get cssElementName() {}
set cssElementName(value) {}
```

Property Value

String

A string containing the CSS element name for item's type.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveBrush Property

Gets the effective brush that should be used to paint this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveBrush() {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

This property returns the brush set locally for an item via its brush property, or looks up in the styling hierarchy for a brush set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
brush Property
style Property

DiagramItem.effectiveCaptionBackBrush Property

Gets the effective brush that should be used to paint node's caption bar.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveCaptionBackBrush() {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveFont Property

Gets the effective font that should be used to render this item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveFont() {}
```

Property Value

Font

An instance of the Font class.

Remarks

This property returns a font set locally for an item via its font property, or constructs a font from the results of effectiveFontName and effectiveFontSize properties.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
font Property
style Property

DiagramItem.effectiveFontName Property

Gets the name of the effective font that should be used to render this item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get effectiveFontName() {}
```

Property Value

String

A string specifying the font name.

Remarks

This property looks up in the styling hierarchy for font names set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveFontSize Property

Gets the size of the effective font that should be used to render this item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get effectiveFontSize() {}
```

Property Value

Number

A number specifying the font size.

Remarks

This property looks up in the styling hierarchy for font sizes set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveFontStyle Property

Gets the style of the effective font that should be used to render this item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveFontStyle() {}
```

Property Value

FontStyle

A combination of FontStyle enumeration members.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveOpacity Property

Gets the effective opacity that should be used to paint this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveOpacity() {}
set effectiveOpacity(value) {}
```

Property Value

Number

A value between 0 and 1 specifying the item's opacity.

Remarks

Returns opacity value or current OpacityAnimation progress.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveShadowColor Property

Gets the effective color that should be used to render this item's shadow.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveShadowColor() {}
```

Property Value

String

A string specifying the item's shadow color.

Remarks

This property returns the shadow color set locally for an item via shadowColor, or looks up in the styling hierarchy for a shadow color set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveStroke Property

Gets the effective color that should be used to stroke this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveStroke() {}
```

Property Value

String

A string specifying the item's stroke color.

Remarks

This property returns the stroke color set locally for an item via stroke, or looks up in the styling hierarchy for a stroke color set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveStrokeDashArray Property

Gets the effective line dash array that should be applied when stroking this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveStrokeDashArray() {}
set effectiveStrokeDashArray(value) {}
```

Property Value

Array

An array of numbers specifying the item's stroke dash array.

Remarks

This property returns the array set locally for an item via strokeDashArray, or looks up in the styling hierarchy for a stroke color set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveStrokeDashOffset Property

Gets the effective line dash offset that should be applied when stroking this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveStrokeDashOffset() {}
set effectiveStrokeDashOffset(value) {}
```

Property Value

Number

A number specifying the item's stroke dash offset.

Remarks

This property returns the offset set locally for an item via strokeDashOffset, or looks up in the styling hierarchy for a stroke color set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveStrokeDashStyle Property

Gets the effective line dash pattern that should be applied when stroking this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveStrokeDashStyle() {}
```

Property Value

DashStyle

A member of the DashStyle enumeration.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveStrokeThickness Property

Gets the effective line thickness that should be applied when stroking this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveStrokeThickness() {}
```

Property Value

Number

A number specifying the item's stroke thickness.

Remarks

This method returns the stroke thickness set locally for an item via strokeThickness, or looks up in the styling hierarchy for a thickness set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
strokeThickness Property
style Property

DiagramItem.effectiveTextColor Property

Gets the effective text color that should be used to render this item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveTextColor() {}
```

Property Value

String

A string specifying the item's text color.

Remarks

This method returns the text color set locally for an item via textColor, or looks up in the styling hierarchy for a text color set in the item's style, the diagram's style or the current theme.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveTextStroke Property

Gets the effective text color that should be used to render this item's text outline.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveTextStroke() {}
```

Property Value

String

A string specifying the item's text outline color.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.effectiveTextStrokeThickness Property

Gets the effective width of the text outline of this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get effectiveTextStrokeThickness() {}
```

Property Value

Number

The width of the text outline of this item.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.font Property

Gets or sets the font used to render item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get font() {}
set font(value) {}
```

Property Value

Font

An instance of the Font class.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
text Property
textColor Property

DiagramItem.graphicsContent Property

Gets an array of visual elements used to render this item on the canvas.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get graphicsContent() {}
```

Property Value

Array

An array containing objects from MindFusion.Drawing namespace.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.hyperLink Property

Gets or sets the hyperlink associated with this diagram item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get hyperLink() {}
set hyperLink(value) {}
```

Property Value

String

A string containing hyperlink URL.

Remarks

This is just a data property, and clicking an item whose **hyperlink** is set does not automatically open the target page. If that's required, you could call **window.open** fron nodeClicked or linkClicked event handlers.

Another way to associate hyperlinks with items is by embedding <a> elements in text when enableStyledText is set to **true**. The control raises hyperlinkClicked event when users click on such

embeded hyperlinks.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.id Property

Gets or sets a custom identifier associated with this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get id() {}
set id(value) {}
```

Property Value

Object

An Object containing custom identifier.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.ignoreLayout Property

Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get ignoreLayout() {}
set ignoreLayout(value) {}
```

Property Value

Boolean

true to ignore this item when arranging the diagram, or **false** otherwise.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.layoutTraits Property

Gets an object containing properties specific to individual layout algorithms.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get layoutTraits() {}
```

Property Value

Object

An object containing layout properties.

Remarks

At this time this property is used only by TreeLayout in JavaScript version of the library:

- **TreeLayout** looks for a **treeLayoutAssistant** value set to a member of the AssistantType enumeration.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.lineAlignment Property

Gets or sets the vertical alignment of item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get lineAlignment() {}
set lineAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.locked Property

Gets or sets whether users are allowed to interact with this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get locked() {}
set locked(value) {}
```

Property Value

Boolean

true to prevent interaction, or **false** otherwise. Default value is **false**.

Remarks

A locked item cannot be selected, moved or resized interactively. Additionally, its text cannot be edited in-place.

Locked nodes still can be moved or resized programmatically through their bounds property. Positions of locked links' control points can be changed programmatically too - through their controlPoints collections.

Locked items positions can be changed also by the automatic layout algorithms. To prevent automatic layout methods from changing positions of individual items, set the ignoreLayout property of DiagramItem to **true**.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
visible Property

DiagramItem.opacity Property

Gets or sets opacity of this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get opacity() {}
set opacity(value) {}
```

Property Value

Number

A value between 0 and 1 specifying the item's opacity.

Remarks

This value is applied as a factor multiplying the alpha channel of all colors in the item's graphics.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
brush Property
stroke Property

DiagramItem.parent Property

Gets the Diagram to which this item belongs.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get parent() {}
```

Property Value

Diagram

An instance of the Diagram class.

Remarks

Call diagram's addItem method to add items to it.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.pen Property

Gets or sets the color used to stroke item's borders.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get pen() {}
set pen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

Remarks

This is synonim for the stroke property, and will be removed in future versions of the library.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.selected Property

Gets or sets a value indicating whether a diagram item is selected.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get selected() {}
set selected(value) {}
```

Property Value

Boolean

true if the item is selected, or **false** otherwise.

Remarks

Use the property to check which items are selected by the user, or to programmatically select items if needed.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.shadowColor Property

Gets or sets the color used to draw item's shadow.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get shadowColor() {}
set shadowColor(value) {}
```

Property Value

String

A string specifying the shadow color.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.shadowOffsetX Property

Gets or sets the horizontal offset of item's shadow.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get shadowOffsetX() {}
set shadowOffsetX(value) {}
```

Property Value

Number

A number specifying the shadow's horizontal offset.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.shadowOffsetY Property

Gets or sets the vertical offset of item's shadow.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get shadowOffsetY() {}
set shadowOffsetY(value) {}
```

Property Value

Number

A number specifying the shadow's vertical offset.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.stroke Property

Gets or sets the color used to stroke item's frame.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get stroke() {}
set stroke(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
brush Property
opacity Property

DiagramItem.strokeDashArray Property

Gets or sets an array of values specifying the dash pattern.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get strokeDashArray() {}
set strokeDashArray(value) {}
```

Property Value

Array

An array of numbers specifying the dash pattern.

Remarks

This value is used when strokeDashStyle is set to Custom.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.strokeDashOffset Property

Gets or sets the the distance between the line start and the beginning of the dash pattern.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get strokeDashOffset() {}
set strokeDashOffset(value) {}
```

Property Value

Number

The dash offset.

Remarks

This value is used when strokeDashStyle is set to Custom.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.strokeDashStyle Property

Gets or sets the line dash pattern applied when stroking item's frame.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get strokeDashStyle() {}
set strokeDashStyle(value) {}
```

Property Value

DashStyle

A member of the DashStyle enumeration.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.strokeThickness Property

Gets or sets the line width applied when stroking item's frame.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get strokeThickness() {}
set strokeThickness(value) {}
```

Property Value

Number

A number specifying the stroke thickness.

Remarks

Setting this value to zero makes stroke one-pixel wide regardless of diagram's measureUnit.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.style Property

Gets or sets the style associated with this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get style() {}
set style(value) {}
```

Property Value

Style

An instance of the Style class.

Remarks

Individual properties of DiagramItem have higher priority than ones set through a Style object. If neither are set, values are inherited from Diagram's style or from current theme.

Note

As of version 4.9 of the library, items can also by styled via CSS, where a custom CSS property name inside a declaration matches respective Style property, but is hyphenated, and type selectors match item type name with added mfd- prefix. For example:

```
mfd-shapenode {
    --brush: lightblue;
    --stroke: blue;
    --stroke-thickness: 2px;
    --font-name: 'Verdana';
    --font-size: 12px;
}
```

You can also associate CSS rules with individual items by setting cssClass property and specifying it in class selectors.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.tag Property

Gets or sets custom data tag associated with this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get tag() {}
set tag(value) {}
```

Property Value

Object

An Object containing custom data.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.text Property

Gets or sets the text label displayed by this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get text() {}
set text(value) {}
```

Property Value

String

A string containing the item's text. The default value is empty string "".

Remarks

This property defines item's main text content. You can add more text labels by calling the addLabel method, and customize their position and appearance via the various members of returned ItemLabel object.

Users can double-click an item to edit its text interactively when the allowInplaceEdit property is enabled. You can trigger the edit operation from a different event handler by calling beginEdit.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
font Property
textAlignment Property
lineAlignment Property
textColor Property

DiagramItem.textAlignment Property

Gets or sets the horizontal alignment of item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get textAlignment() {}
set textAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

text Property
lineAlignment Property

DiagramItem.textColor Property

Gets or sets the color used to fill item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string value identifying the text fill color.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
text Property
font Property

DiagramItem.textComponent Property

Gets the Text object instance used to render item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get textComponent() {}
set textComponent(value) {}
```

Property Value

Text

The Text component.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.textPadding Property

Gets or sets the padding distance between item's boundaries and text label.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get textPadding() {}
set textPadding(value) {}
```

Property Value

Thickness

An instance of the Thickness class.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace
text Property

DiagramItem.textStroke Property

Gets or sets the color used to stroke item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get textStroke() {}
set textStroke(value) {}
```

Property Value

String

A string value identifying the text stroke color.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.textStrokeThickness Property

Gets or sets the outline width of item's text.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get textStrokeThickness() {}
set textStrokeThickness(value) {}
```

Property Value

Number

A number specifying the thickness of text outline.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.tooltip Property

Gets or sets tooltip text that should be displayed when the mouse hovers over this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get tooltip() {}
set tooltip(value) {}
```

Property Value

String

A string containing the tooltip text.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.topLevel Property

Gets if this is a top-level item in the diagram.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

```
JavaScript
get topLevel() {}
```

Property Value

Boolean

true if this is a top-level item in the diagram, or **false** if inside a container.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.visible Property

Gets or sets a value indicating whether this item is visible.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get visible() {}
set visible(value) {}
```

Property Value

Boolean

true if this item should be displayed, or **false** otherwise.

Remarks

Invisible items are not drawn at all. You might use the property to hide or show items temporarily, or to construct more complex groups of attached items in which intermediate helper items are invisible.

Invisible nodes are ignored by automatic layout classes. The position of such nodes won't change when the arrange method of Diagram is called. Visible items can be ignored too by enabling their ignoreLayout property.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.weight Property

Gets or sets a weight value used by layout and path-finding algorithms.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get weight() {}
set weight(value) {}
```

Property Value

Number

A number specifying the weight of this item. The default value is 1.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramItem.zIndex Property

Gets the z-order position of this item.

Namespace: MindFusion.Diagramming
File: DiagramItem.js

Syntax

JavaScript

```
get zIndex() {}
set zIndex(value) {}
```

Property Value

Number

A number specifying the position of this item within diagram's z-order.

Remarks

Use the method to arrange items in depth. Items with lower **zIndex** appear behind items with higher **zIndex**.

See Also

DiagramItem Members
DiagramItem Class
MindFusion.Diagramming Namespace

DiagramLink Class

Represents a link between two diagram nodes.

Syntax

JavaScript

```
// class
DiagramLink.prototype = {}
```

Remarks

Links between nodes in MindFusion.Diagramming for JavaScript are represented by instances of the **DiagramLink** class. Links connect at most two nodes, which can be accessed via the origin and destination properties. If connected to node anchor points, the point indices can be accessed using the originAnchor and destinationAnchor properties.

Links are built of smaller parts called segments, which can be straight lines, cascading orthogonal lines or Bezier curves. The type of segments is set through the shape property. The number of link's segments and their locations are specified by the segmentCount and controlPoints properties. If changing the coordinates of individual points in **controlPoints**, you should also call the updateFromPoints method to update link's internal state.

Typically, only the start and end segments of a link are updated when the nodes it connects are moved. Number, orientation and/or position of link segments can change dynamically in various ways depending on the values of autoRoute and dynamic properties.

Links can display distinct arrowhead shapes at their ends. These shapes are configured using the headShape and baseShape properties. The sizes of the base and arrowhead images are specified with the headShapeSize and baseShapeSize properties. Arrowhead shapes can also be displayed at the midpoint of each segment to help distinguish between different links. The type and size of these shapes are controlled by the intermediateShape and intermediateShapeSize properties. The outlines of the arrowhead shapes are rendered using the headStroke color.

The color of link segments is specified by the stroke property, while the fill color of the base and head shapes is controlled by the baseBrush and headBrush properties.

Links can have associated text, displayed according to the text, textColor, and textStyle properties. The addLabel method allows multiple labels to be displayed per link. Labels can automatically avoid overlapping nodes or other labels when their autoArrange property is enabled.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramLink

See Also

DiagramLink Members
MindFusion.Diagramming Namespace

DiagramLink Members

Represents a link between two diagram nodes.

The following tables list the members exposed by the DiagramLink type.

Public Constructors

	Name	Description
	DiagramLink	Initializes a new instance of the DiagramLink class between the specified nodes using the specified diagram link as a prototype.

Public Properties

	Name	Description

	autoRoute	Gets or sets whether this link should be automatically routed to avoid crossing nodes.
	baseBrush	Gets or sets an object that specifies how to fill link's baseShape.
	baseShape	Gets or sets the arrowhead shape displayed at the link' start point.
	baseShapeSize	Gets or sets the size of the arrowhead displayed at the link's start point.
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	controlPoints	Gets or sets the control points of this link.
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	destination	Gets or sets the destination node of this link.
	destinationAnchor	Gets or sets the anchor point to which the link connects at its destination.
	destinationIndex	Gets or sets the index of link's destination row when destination is a TableNode.
	dynamic	Gets or sets whether link's end segments are automatically adjustedin order to keep pointing to the centers of the nodes it connects.
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)

	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	endPoint	Gets or sets the last control point of this link.
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	headBrush	Gets or sets an object that specifies how to fill link's headShape.
	headShape	Gets or sets the arrowhead shape displayed at the link' end point.
	headShapeSize	Gets or sets the size of the arrowhead displayed at the link's end point.
	headStroke	Gets or sets the brush used to stroke the outline of link's arrowheads.
	headStrokeDashStyle	Gets or sets the dash pattern of arrowhead outlines.
	headStrokeThickness	Gets or sets the thickness of arrowhead outlines.
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	intermediateBrush	Gets or sets an object that specifies how to fill intermediateShape arrowheads.
	intermediateShape	Gets or sets the arrowhead shape displayed at the middle of each segment of the link.
	intermediateShapeSize	Gets or sets the size of the arrowhead displayed at the middle of each segment of the link.

	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	origin	Gets or sets the origin node of this link.
	originAnchor	Gets or sets the anchor point to which the link connects at its origin.
	originIndex	Gets or sets the index of link's origin row when origin is a **TableNode**.
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	segmentCount	Gets or sets the number of segments that form the link's geometry.
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the type of link segments and how they are oriented relatively to each other.
	startPoint	Gets or sets the first control point of this link.
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)

	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text displayed by this link.
	textAlignment	Gets or sets the alignment of link's text.
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	textStyle	Gets or sets the placement and orientation of link's text.
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this link.
	allowDrag	DiagramItem.allowDrag override.
	animate	Animates the item. (Inherited from DiagramItem.)
	clone	DiagramItem.clone override.
	containsPoint	DiagramItem.containsPoint override.
	endDrag	DiagramItem.endDrag override.
	fromJson	DiagramItem.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)

	getLength	Gets the length of the links
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	intersects	Checks whether the link contour intersects with the outline of the specified node.
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramItem.LoadFromXml override. Loads the link's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	pointAlongLength	Returns a point lying on a polyline that is positioned at a relative position along the polyline's length.
	putEndPointsAtNodeBorders	Places the link's end points on the outline of its origin and destination nodes.
	removeCanvasElements	Derived classes must implement this method by removing their graphics primitives from the canvas.elements collection. (Inherited from DiagramItem.)
	removeLabel	Removes the specified label from this link.
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	**DiagramItem.restoreLocationState** override.
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	restoreState	DiagramItem.restoreState override.
	route	Routes this link, regardless of whether the link's **autoRoute** property is enabled.
	saveLocationState	**DiagramItem.saveLocationState** override.
	saveState	DiagramItem.saveState override.
	saveToXml	Saves the item's content into an XML element.
	setDestination	Sets the destination node of this link.
	splitSegment	Splits the link segment with the specified index at the specified point.
	startDrag	DiagramItem.startDrag override.
	toJson	DiagramItem.toJson override.
	updateCanvasElements	Updates the elements on the HTML Canvas.
	updateDrag	DiagramItem.updateDrag override.

	updateFromPoints	Updates the link's internal state after the link's control points have been changed.

See Also

DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink Constructor

Initializes a new instance of the DiagramLink class between the specified nodes using the specified diagram link as a prototype.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function DiagramLink ([parent, [origin, [destination]]])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the link properties.

origin
Optional.
DiagramNode | Point. The origin node of the new link.

destination
Optional.
DiagramNode | Point. The destination node of the new link.

See Also

DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink Methods

DiagramLink.addLabel Method

Adds a new label to this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function addLabel (text)
```

Parameters

text
String. The label's text.

Return Value

A LinkLabel instance.

Remarks

A link's main text content is specified via its text property. Call **addLabel** if you need to display additional text labels, such as identifiers or hints to the user.

Position of labels can be specified by calling methods such as setControlPointPosition. You can let users move labels interactively by setting the behavior property of DiagramView to MoveLabels.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
removeLabel Method

DiagramLink.allowDrag Method

DiagramItem.allowDrag override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function allowDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.clone Method

DiagramItem.clone override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function clone ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.containsPoint Method

DiagramItem.containsPoint override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function containsPoint (point)
```

Parameters

point
The Point to check.

Return Value

Boolean. **true** if the link contains the point, or **false** otherwise.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.endDrag Method

DiagramItem.endDrag override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function endDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.fromJson Method

DiagramItem.fromJson override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.getLength Method

Gets the length of the links

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function getLength ([segments])
```

Parameters

segments
Optional.
Array. A list of points specifying the polyline segments.

Return Value

Number. The sum of the lengths of the line's segments.

Remarks

If the link's Shape is set to Bezier, an approximate length is calculated.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.intersects Method

Checks whether the link contour intersects with the outline of the specified node.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function intersects (rect)
```

Parameters

rect
Rect. The bounding rectangle of the specified node.

Return Value

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.loadFromXml Method

DiagramItem.LoadFromXml override. Loads the link's content from an XML element.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.pointAlongLength Method

Returns a point lying on a polyline that is positioned at a relative position along the polyline's length.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function pointAlongLength (lengthFactor, [segments])
```

Parameters

lengthFactor
Number. A value from 0 to 1 specifying a portion of link's length.

segments
Optional.
Array. A list of points specifying the polyline segments.

Return Value

Point. A Point instance.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.putEndPointsAtNodeBorders Method

Places the link's end points on the outline of its origin and destination nodes.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function putEndPointsAtNodeBorders (action, routing)
```

Parameters

action

routing

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.removeLabel Method

Removes the specified label from this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
function removeLabel (label)
```

Parameters

label
A LinkLabel instance.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
addLabel Method

DiagramLink.restoreLocationState Method

DiagramItem.restoreLocationState override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
function restoreLocationState (ist)
```

Parameters

ist
A **DiagramLinkState** instance.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.restoreState Method

DiagramItem.restoreState override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
```

```
function restoreState ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.route Method

Routes this link, regardless of whether the link's autoRoute property is enabled.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function route ([args])
```

Parameters

args
Optional.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.saveLocationState Method

DiagramItem.saveLocationState override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function saveLocationState ()
```

Return Value

DiagramLinkState. A DiagramLinkState instance.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.saveState Method

DiagramItem.saveState override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
function saveState ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.saveToXml Method

Saves the item's content into an XML element.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.setDestination Method

Sets the destination node of this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
```

```
function setDestination (destination)
```

Parameters

destination
The DiagramNode to set as destination for this link.

Remarks

For internal use.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
origin Property
destination Property

DiagramLink.splitSegment Method

Splits the link segment with the specified index at the specified point.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function splitSegment ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.startDrag Method

DiagramItem.startDrag override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function startDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.toJson Method

DiagramItem.toJson override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
function toJson ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.updateCanvasElements Method

Updates the elements on the HTML Canvas.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
function updateCanvasElements ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.updateDrag Method

DiagramItem.updateDrag override.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
```

```
function updateDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.updateFromPoints Method

Updates the link's internal state after the link's control points have been changed.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
function updateFromPoints ()
```

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink Properties

DiagramLink.autoRoute Property

Gets or sets whether this link should be automatically routed to avoid crossing nodes.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get autoRoute() {}
set autoRoute(value) {}
```

Property Value

Boolean

true to automatically route this link, or **false** otherwise.

Remarks

If **autoRoute** is enabled, a link automatically finds the shortest possible path between its origin and destination that doesn't intersect obstacle nodes. The number and orientation of the link's segments are updated when the nodes it links are moved around. The positions of end points of a routed link are not changed, unless the link is also set to dynamic.

Performance Note

Auto-routing is CPU intensive operation and is triggered each time a node is moved or resized. If populating large diagrams from data, best enable this property only after creating all nodes. Alternatively, call route or routeAllLinks method explicitly when appropriate for your application.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
dynamic Property

DiagramLink.baseBrush Property

Gets or sets an object that specifies how to fill link's baseShape.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get baseBrush() {}
set baseBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

If this property is not set, the arrowhead is filled using the link's brush.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.baseShape Property

Gets or sets the arrowhead shape displayed at the link' start point.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get baseShape() {}
set baseShape(value) {}
```

Property Value

Shape | *String*

A Shape instance that represents an arrowhead shape.

Remarks

This property specifies the shape that is displayed at the beginning of a link, that is, the point where it connects to its origin node. This shape is rotated simultaneously with the first segment of the link so that it follows the overall link direction.

The ArrowHeads class exposes predefined arrowhead shapes as static methods, each one returning Shape.fromId result for respective shape identifier.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
headShape Property
intermediateShape Property

DiagramLink.baseShapeSize Property

Gets or sets the size of the arrowhead displayed at the link's start point.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get baseShapeSize() {}
set baseShapeSize(value) {}
```

Property Value

Number

A number that represents the size of the arrowhead shape.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.controlPoints Property

Gets or sets the control points of this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get controlPoints() {}
set controlPoints(value) {}
```

Property Value

Array

An array of Point objects containing coordinates of link segments.

Remarks

This property lets you change coordinates of link points. The number of points per segment depends on link's shape. Links made of Bézier-splines have four control points for each segment. Links with other shape types have two control points per segment. Adjacent segments share a common control point. After setting new positions of control points, you must call the updateFromPoints method.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
segmentCount Property

DiagramLink.destination Property

Gets or sets the destination node of this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get destination() {}
set destination(value) {}
```

Property Value

DiagramNode

A reference to the destination DiagramNode.

Remarks

When handling validation events, this property keeps pointing to the original destination node. Use the destination property of LinkEventArgs to determine which node the link will connect to if operation is approved.

See Also

DiagramLink.destinationAnchor Property

Gets or sets the anchor point to which the link connects at its destination.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get destinationAnchor() {}
set destinationAnchor(value) {}
```

Property Value

Number

An integer value specifying the index of the destination anchor point.

Remarks

This property is valid when destination node's anchorPattern is set.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.destinationIndex Property

Gets or sets the index of link's destination row when destination is a TableNode.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get destinationIndex() {}
set destinationIndex(value) {}
```

Property Value

Number

An integer value specifying the index of destination row.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.dynamic Property

Gets or sets whether link's end segments are automatically adjustedin order to keep pointing to the centers of the nodes it connects.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get dynamic() {}
set dynamic(value) {}
```

Property Value

Boolean

true if the link automatically adjusts its end points, or false otherwise.

Remarks

If this property is enabled, moving link's adjacent nodes or next-to-last link control points will also update the position of link end points along the node border.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
autoRoute Property

DiagramLink.endPoint Property

Gets or sets the last control point of this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get endPoint() {}
set endPoint(value) {}
```

Property Value

Point

A Point instance containing the control point coordinates.

Remarks

This is a shortcut for accessing last element of controlPoints.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
startPoint Property

DiagramLink.headBrush Property

Gets or sets an object that specifies how to fill link's headShape.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get headBrush() {}
set headBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

If this property is not set, the arrowhead is filled using the link's brush.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.headShape Property

Gets or sets the arrowhead shape displayed at the link' end point.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get headShape() {}
set headShape(value) {}
```

Property Value

Shape | *String*

A Shape instance that represents an arrowhead shape.

Remarks

This property specifies the shape that is displayed at the end of a link, that is, the point where it connects to its destination node. This shape is rotated simultaneously with the last segment of the link so that it follows the overall link direction.

The ArrowHeads class exposes predefined arrowhead shapes as static methods, each one returning Shape.fromId result for respective shape identifier.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
baseShape Property
intermediateShape Property

DiagramLink.headShapeSize Property

Gets or sets the size of the arrowhead displayed at the link's end point.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get headShapeSize() {}
set headShapeSize(value) {}
```

Property Value

Number

A number that represents the size of the arrowhead shape.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
headShape Property

DiagramLink.headStroke Property

Gets or sets the brush used to stroke the outline of link's arrowheads.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get headStroke() {}
set headStroke(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
headShape Property
baseShape Property

DiagramLink.headStrokeDashStyle Property

Gets or sets the dash pattern of arrowhead outlines.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get headStrokeDashStyle() {}
set headStrokeDashStyle(value) {}
```

Property Value

DashStyle

A member of the DashStyle enumeration.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.headStrokeThickness Property

Gets or sets the thickness of arrowhead outlines.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get headStrokeThickness() {}
set headStrokeThickness(value) {}
```

Property Value

Number

A number specifying the line thickness.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.intermediateBrush Property

Gets or sets an object that specifies how to fill intermediateShape arrowheads.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get intermediateBrush() {}
set intermediateBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

If this property is not set, intermediate arrowheads are filled using the link's brush.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
intermediateShapeSize Property

DiagramLink.intermediateShape Property

Gets or sets the arrowhead shape displayed at the middle of each segment of the link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get intermediateShape() {}
set intermediateShape(value) {}
```

Property Value

Shape | String

A Shape instance that represents an arrowhead shape.

Remarks

The property specifies what shape to display at the middle of each segment of a link. This could help users more easily follow the path of long links.

The ArrowHeads class exposes predefined arrowhead shapes as static methods, each one returning Shape.fromId result for respective shape identifier.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
headShape Property
baseShape Property

DiagramLink.intermediateShapeSize Property

Gets or sets the size of the arrowhead displayed at the middle of each segment of the link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get intermediateShapeSize() {}
set intermediateShapeSize(value) {}
```

Property Value

Number

A number that represents the size of the arrowhead shape.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.origin Property

Gets or sets the origin node of this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get origin() {}
set origin(value) {}
```

Property Value

DiagramNode

A reference to the origin DiagramNode.

Remarks

When handling validation events, this property keeps pointing to the original origin node. Use the origin property of LinkEventArgs to determine which node the link will connect to if operation is approved.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
destination Property
originIndex Property

DiagramLink.originAnchor Property

Gets or sets the anchor point to which the link connects at its origin.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get originAnchor() {}
set originAnchor(value) {}
```

Property Value

Number

An integer value specifying the index of the origin anchor point.

Remarks

This property is valid when origin node's anchorPattern is set.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.originIndex Property

Gets or sets the index of link's origin row when origin is a TableNode.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get originIndex() {}
set originIndex(value) {}
```

Property Value

Number

An integer value specifying the index of origin row.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.segmentCount Property

Gets or sets the number of segments that form the link's geometry.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get segmentCount() {}
set segmentCount(value) {}
```

Property Value

Number

An integer value specifying the number of segments of this link.

Remarks

Link segments are either straight lines or Bezier curves, depending on the value of the link's shape property. The segment coordinates are defined by the elements of controlPoints, which contains four points per Bezier segment or two points per straight line segment, where each pair of adjacent segments share a common point. Changing the **segmentCount** property value will resize the **controlPoints** array so that it contains appropriate number of points.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.shape Property

Gets or sets the type of link segments and how they are oriented relatively to each other.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get shape() {}
set shape(value) {}
```

Property Value

LinkShape

A member of the LinkShape enumeration.

Remarks

This property specifies whether the segments of a link are straight lines or curves. In case of straight lines, they can be set all horizontal and vertical, each segment orthogonal to its neighbor, by using the Cascading shape.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
controlPoints Property

DiagramLink.startPoint Property

Gets or sets the first control point of this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get startPoint() {}
set startPoint(value) {}
```

Property Value

Point

A Point instance containing the control point coordinates.

Remarks

This is a shortcut for accessing first element of controlPoints.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace
endPoint Property

DiagramLink.text Property

Gets or sets the text displayed by this link.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get text() {}
set text(value) {}
```

Property Value

String

A string containing the link's text. The default value is empty string "".

Remarks

Gets or sets the text that should be drawn over a link. Use this to provide to end-users a short description of the relation's purpose. The text appearance can be customized through the textColor and textStyle properties.

Version 1.6 of the control adds the capability to display multiple labels per link via the LinkLabel class. Labels can automatically avoid overlapping nodes or other labels when their autoArrange property is set.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramLink.textAlignment Property

Gets or sets the alignment of link's text.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

```
JavaScript
get textAlignment() {}
set textAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration. The default value is **Center**.

Remarks

Currently this property is considered only when textStyle is set to OverLongestSegment.

See Also

DiagramLink Members
DiagramLink Class

MindFusion.Diagramming Namespace

DiagramLink.textStyle Property

Gets or sets the placement and orientation of link's text.

Namespace: MindFusion.Diagramming
File: DiagramLink.js

Syntax

JavaScript

```
get textStyle() {}
set textStyle(value) {}
```

Property Value

LinkTextStyle

A member of the LinkTextStyle enumeration.

See Also

DiagramLink Members
DiagramLink Class
MindFusion.Diagramming Namespace

DiagramNode Class

Represents a diagram node.

Syntax

JavaScript

```
// class
DiagramNode.prototype = {}
```

Remarks

DiagramNode objects can represent graph vertices, nodes in organizational or flow diagrams, entities in ER diagrams, and so on. **DiagramNode** is an abstract class. You cannot instantiate it directly, but instead use objects from specific classes such as ShapeNode, TableNode, ContainerNode, CompositeNode. You can also define your own node type by deriving from **DiagramNode** or its descendants.

Use the bounds property to get or change the location and size of a node. Constraints upon how users are allowed to modify a node are set through the constraints and enabledHandles properties. The former specifies constraints on move direction and node size. The latter enables or disables specific manipulation handles of a node.

The main textual content of a node is set using its text property and can be customized by setting font and textAlignment. Supplementary labels can be created by calling addLabel.

Relations between nodes are indicated by connecting the nodes with DiagramLink instances. Links are considered directed graph edges, so each node has two collections referring to links: incomingLinks and outgoingLinks. anchorPattern lets you specify the points of a node to which incoming or outgoing links

can be connected. The allowIncomingLinks property lets you specify whether incoming links can be drawn to a node. The allowOutgoingLinks property lets you do that for outgoing links.

A user is allowed to expand or collapse the tree starting from a node if expandable is set to **true**. The expanded property gets or sets whether the subtree starting from a node is expanded or not.

Nodes can be attached to other items, and be used as simple labels, image-holders or other kinds of UI elements. The attachTo method overloads provide various options for that.

Nodes can be rotated, either interactively using the rotation handle above a node, or programmatically by setting the rotationAngle property.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode

    MindFusion.Diagramming.CaptionedNode
    MindFusion.Diagramming.CompositeNode
    MindFusion.Diagramming.ControlNode
    MindFusion.Diagramming.DummyNode
    MindFusion.Diagramming.FreeFormNode
    MindFusion.Diagramming.ShapeNode
    MindFusion.Diagramming.VideoNode

See Also

DiagramNode Members
MindFusion.Diagramming Namespace

DiagramNode Members

Represents a diagram node.

The following tables list the members exposed by the DiagramNode type.

Public Constructors

	Name	Description
	DiagramNode	Initializes a new instance of the **DiagramNode** class.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node.
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node.
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node.
	attachedNodes	Returns the nodes attached to this node.
	bounds	Gets or sets a rectangle that defines the position of this node.

	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle.
	constraints	Gets or sets constraints applied to the location and size of this node.
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle.
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node.
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)

	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node.
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node.
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node.
	expanded	Gets or sets whether the subtree rooted at this node is expanded.
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles.
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node.
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached.
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms.
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node.
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated.

	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	showDeleteButton	Gets or sets whether this node displays a delete button.
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)

	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type.
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override.
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it.
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it.
	clone	DiagramItem.clone override.
	containsPoint	**DiagramItem.containsPoint** override.
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point.
	detach	Detaches this node from its current master node.
	endDrag	**DiagramItem.endDrag** override.
	fromJson	DiagramItem.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getAllLinks	Returns all links connected to this node.
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline.
	getOutline	Gets the bounding rectangle of this node.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding **rectangle** of this item when it is rotated.
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)

	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramItem.loadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked.
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked.
	removeCanvasElements	DiagramItem.removeCanvasElements override.
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	DiagramItem.restoreLocationState override.
	restoreState	DiagramItem.restoreState override.
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	**DiagramItem.saveLocationState** override.
	saveState	DiagramItem.saveState override.
	saveToXml	Saves the item content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node.
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramItem.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override.

See Also

DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode Constructor

Initializes a new instance of the DiagramNode class.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function DiagramNode ([parent])
```

Parameters

parent
Optional.
Diagram. A Diagram instance whose default node attributes are copied to this node.

See Also

DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode Methods

DiagramNode.acceptLinks Method

Determines whether the node can be connected with links of the specified type.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function acceptLinks (outgoing)
```

Parameters

outgoing
Boolean. **true** to check for outgoing links, or **false** otherwise.

Return Value

Boolean. **true** if the node accepts connections of the specified type, or **false** otherwise.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.allowDrag Method

DiagramItem.allowDrag override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function allowDrag (ist)
```

Parameters

ist
InteractionState. An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.attach Method

Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function attach (subordinate)
```

Parameters

subordinate
A DiagramNode instance specifying the node that should be attached.

Remarks

Layout algorithms can treat a group of attached nodes either as a single node to be placed in the arranged graph, or as distinct nodes whose final positions are set independently from the initial group layout. This is controlled via the keepGroupLayout property.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
detach Method
attachTo Method
attachedNodes Property

DiagramNode.attachTo Method

Attaches this node to the specified master node so that when the master is moved, the attached node follows it.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function attachTo (node)
```

Parameters

node
A DiagramNode instance specifying the master node.

Remarks

Layout algorithms can treat a group of attached nodes either as a single node to be placed in the arranged graph, or as distinct nodes whose final positions are set independently from the initial group layout. This is controlled via the keepGroupLayout property.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
detach Method
attach Method
attachedNodes Property

DiagramNode.clone Method

DiagramItem.clone override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function clone ()
```

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.containsPoint Method

DiagramItem.containsPoint override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function containsPoint (point)
```

Parameters

point
Point. The point to check.

Return Value

Boolean. true it this node contains the specified point; otherwise, false.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.createConnectionPoint Method

Creates a connection point between this node and the specified link at the specified point.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function createConnectionPoint (link, point, incoming)
```

Parameters

link
A DiagramLink that should be connected to this node.

point
The Point at which the link should be connected.

incoming
Boolean. **true** if link is an incoming link, or **false** otherwise.

Return Value

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.detach Method

Detaches this node from its current master node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function detach ()
```

Remarks

This method does nothing if the node hasn't been previously attached to another node via the attach or attachTo methods.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
attachedNodes Property

DiagramNode.endDrag Method

DiagramItem.endDrag override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function endDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.fromJson Method

DiagramItem.fromJson override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.getAllLinks Method

Returns all links connected to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function getAllLinks ()
```

Return Value

Array. An array of DiagramLink objects.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.getIntersection Method

Calculates the intersection point between the segment defined by the specified point pair and the node's outline.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function getIntersection (point1, point2)
```

Parameters

point1
The first Point of the segment.

point2
The second **Point** of the segment.

Return Value

A **Point** object representing the intersection point.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.getOutline Method

Gets the bounding rectangle of this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function getOutline ()
```

Return Value

A rectangle.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.getRotatedBounds Method

Gets the bounding rectangle of this item when it is rotated.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function getRotatedBounds ()
```

Return Value

A Rect that bounds the rotated shape of this node.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
rotationAngle Property

DiagramNode.loadFromXml Method

DiagramItem.loadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.raiseClicked Method

Raises an event which notifies that a node has been clicked.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function raiseClicked (mousePosition, mouseButton)
```

Parameters

mousePosition
Point. The position of the mouse.

mouseButton
A value identifying the clicked mouse button.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.raiseDoubleClicked Method

Raises an event which notifies that a node has been double-clicked.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function raiseDoubleClicked (mousePosition, mouseButton)
```

Parameters

mousePosition
Point. The position of the mouse.

mouseButton
MouseButton. A value identifying the clicked mouse button.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.removeCanvasElements Method

DiagramItem.removeCanvasElements override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function removeCanvasElements ()
```

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.restoreLocationState Method

DiagramItem.restoreLocationState override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function restoreLocationState (ist)
```

Parameters

ist
DiagramNodeState. A DiagramNodeState instance.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.restoreState Method

DiagramItem.restoreState override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.saveLocationState Method

DiagramItem.saveLocationState override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function saveLocationState ()
```

Return Value

DiagramNodeState. A DiagramNodeState instance.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.saveState Method

DiagramItem.saveState override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function saveState ()
```

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.saveToXml Method

Saves the item content into an XML element.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.setBounds Method

Sets the rectangle that defines the position of the diagram node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
function setBounds (bounds, [updateDependencies])
```

Parameters

bounds
The node's bounding rectangle.

updateDependencies
Optional. **true** if links and attachments must be updated too, or **false** otherwise.

Remarks

Use this method to set the size and/or location of a node. If updateDependencies is set, moving the node will also update the control-point coordinates of all links connected to it.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
bounds Property

DiagramNode.toJson Method

DiagramItem.toJson override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function toJson ()
```

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.updateDrag Method

DiagramItem.updateDrag override.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
function updateDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

DiagramNode Members
DiagramNode Class

DiagramNode Properties

DiagramNode.allowIncomingLinks Property

Gets or sets whether users are allowed to draw incoming links to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get allowIncomingLinks() {}
set allowIncomingLinks(value) {}
```

Property Value

Boolean

true if users are allowed to draw incoming links, or **false** otherwise.

Remarks

Links to a node can always be created programmatically regardless of the value of this property.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.allowOutgoingLinks Property

Gets or sets whether users are allowed to draw outgoing links from this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get allowOutgoingLinks() {}
set allowOutgoingLinks(value) {}
```

Property Value

Boolean

true if users are allowed to draw outgoing links, or **false** otherwise.

Remarks

Links from a node can always be created programmatically regardless of the value of this property.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.anchorPattern Property

Gets or sets the anchor points to which links attach when connected to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get anchorPattern() {}
set anchorPattern(value) {}
```

Property Value

AnchorPattern

An AnchorPattern object that defines a set of anchor points.

Remarks

This property specifies a set of points to which links can be attached. Each anchor point can accept either incoming links, outgoing links or both. If a node does not have any anchor points defined, links can be connected to any part of the node.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
AnchorPoint Class
Shape.anchorPattern Property

DiagramNode.attachedNodes Property

Returns the nodes attached to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get attachedNodes() {}
```

Property Value

Array

An array of nodes attached to this node via the attach or attachTo methods.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
detach Method

DiagramNode.bounds Property

Gets or sets a rectangle that defines the position of this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get bounds() {}
set bounds(value) {}
```

Property Value

Rect

The node's bounding rectangle.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
Diagram.enableSpatialIndex Property

DiagramNode.center Property

Gets the center of this node's bounding rectangle.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get center() {}
```

Property Value

Point

A Point representing the center point of this node.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.constraints Property

Gets or sets constraints applied to the location and size of this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get constraints() {}
set constraints(value) {}
```

Property Value

NodeConstraints

An instance of the NodeConstraints class.

Remarks

These constraints are considered when a user modifies the node interactively. They are ignored if node placement is changed programmatically by using methods and properties.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
enabledHandles Property

DiagramNode.cornerRadius Property

Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get cornerRadius() {}
set cornerRadius(value) {}
```

Property Value

Number

A number specifying the corner radius.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
ShapeNode.shape Property
TableNode.shape Property
ContainerNode.shape Property

DiagramNode.effectiveEffects Property

Gets the effects applied to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get effectiveEffects() {}
```

Property Value

Array

An array of NodeEffect objects.

Remarks

Returns the local effects array if it's not empty, or otherwise the effects defined in item's style or diagram's theme.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.effects Property

Gets the effects applied to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get effects() {}
set effects(value) {}
```

Property Value

Array

An array of NodeEffect objects.

Remarks

The local node effects are applied after the diagram-wide node effects defined in nodeEffects. Setting **null** marker as first element in this list allows ignoring the global diagram effects.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
Node Effects

DiagramNode.enabledHandles Property

Gets or sets a value indicating which adjustment handles are enabled for this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get enabledHandles() {}
set enabledHandles(value) {}
```

Property Value

AdjustmentHandles

A combination of the AdjustmentHandles enumeration members. The default is **AdjustmentHandles.All**.

Remarks

Disabling some selection handles allows restricting the types of modifications that users can perform on a node. Each manipulation handle can be enabled or disabled by setting or clearing the respective flag in the combination.

For example, to make a node resizable only horizontally, set this property to a combination of the Move, ResizeMiddleLeft and ResizeMiddleRight flags. That will disable all manipulation handles, except the left and right ones, which allow horizontal resizing, and the central one allowing movement of the node.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
handlesStyle Property
constraints Property

DiagramNode.expandable Property

Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get expandable() {}
set expandable(value) {}
```

Property Value

Boolean

true if users are allowed to expand/collapse the subtree, or **false** otherwise.

Remarks

This property is useful mostly when nodes form a tree-like hierarchical structure.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
expanded Property
treeCollapsed Event
treeExpanded Event
expandButtonAction Property

DiagramNode.expanded Property

Gets or sets whether the subtree rooted at this node is expanded.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get expanded() {}
set expanded(value) {}
```

Property Value

Boolean

true if the subtree is expanded, or **false** otherwise.

Remarks

By default, the control considers nodes that can be reached by following outgoing links to be the child nodes in a tree, and respectively outgoing links determine the direction in which tree branches collapse or expand. This direction can be reversed by setting the expandOnIncoming property to **true**.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
expandable Property

outgoingLinks Property
incomingLinks Property

DiagramNode.handlesStyle Property

Gets or sets a value specifying appearance and behavior of node's adjustment handles.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get handlesStyle() {}
set handlesStyle(value) {}
```

Property Value

HandlesStyle

A member of the HandlesStyle enumeration. The default value is **SquareHandles**.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace
enabledHandles Property

DiagramNode.incomingLinks Property

Gets the incoming links to this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get incomingLinks() {}
```

Property Value

Array

An array of DiagramLink objects.

Remarks

Set link's destination property to programmatically create incoming link to a node, or set respective argument of createDiagramLink method.

See Also

DiagramNode Members
DiagramNode Class

DiagramNode.masterNode Property

Returns the node to which this node is attached.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get masterNode() {}
```

Property Value

DiagramNode

A DiagramNode reference specifying the node to whichthis node has been previously attached via the attach or attachTo method.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.obstacle Property

Gets or sets whether this node is considered an obstacle by link routing algorithms.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

JavaScript

```
get obstacle() {}
set obstacle(value) {}
```

Property Value

Boolean

true to route links around this node, or **false** to lets links cross the node. The default value is **true**.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.outgoingLinks Property

Gets the outgoing links from this node.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get outgoingLinks() {}
```

Property Value

Array

An array of DiagramLink objects.

Remarks

Set link's origin property to programmatically create outgoing link from a node, or set respective argument of createDiagramLink method.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.rotationAngle Property

Gets or sets the angle at which this node is rotated.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get rotationAngle() {}
set rotationAngle(value) {}
```

Property Value

Number

A number specifying the rotation angle. The default is 0.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramNode.showDeleteButton Property

Gets or sets whether this node displays a delete button.

Namespace: MindFusion.Diagramming
File: DiagramNode.js

Syntax

```
JavaScript
get showDeleteButton() {}
set showDeleteButton(value) {}
```

Property Value

Boolean

true to display delete button for the node, or **false** otherwise.

See Also

DiagramNode Members
DiagramNode Class
MindFusion.Diagramming Namespace

DiagramPage Class

Represents a single page in a DiagramDocument object.

Syntax

```
JavaScript
// class
DiagramPage.prototype = {}
```

Remarks

DiagramPage is derived from Diagram and only adds a few properties, so any code that processes **Diagram** objects will work with **DiagramPage** objects too. A DiagramDocument forwards each event raised by its pages, giving you a chance to use the same handler for events raised by different pages. If a page should implement distinct user interaction or validation logic, it is also possible to attach handlers to the **Diagram** events inherited by **DiagramPage**.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramBase
  MindFusion.Diagramming.Diagram
    MindFusion.Diagramming.DiagramPage

See Also

DiagramPage Members
MindFusion.Diagramming Namespace

DiagramPage Members

Represents a single page in a DiagramDocument object.

The following tables list the members exposed by the DiagramPage type.

Public Constructors

	Name	Description
	DiagramPage	Initializes a new instance of the DiagramPage class.

Public Properties

	Name	Description
	activeItem	Gets the active diagram item. (Inherited from Diagram.)
	activeItemHandlesStyle	Gets or sets the visual style of adjustment handles of the active item. (Inherited from Diagram.)
	adjustmentHandlesSize	Gets or sets the size of adjustment handles that appear around selected items. (Inherited from Diagram.)
	alignToGrid	Gets or sets a value indicating whether the diagram items should be aligned to a grid. (Inherited from Diagram.)
	allowLinksRepeat	Gets or sets a value indicating whether more than one links can connect the same origin and destination diagram nodes. (Inherited from Diagram.)
	allowMultipleResize	Gets or sets a value indicating whether multiple selected nodes can be resized simultaneously. (Inherited from Diagram.)
	allowSelfLoops	Gets or sets a value indicating whether users are allowed to draw reflexive links. (Inherited from Diagram.)
	allowSplitLinks	Gets or sets a value indicating whether link segments can be added and removed interactively. (Inherited from Diagram.)
	allowUnanchoredLinks	Gets or sets a value indicating whether users can link nodes that do not have designated anchor points. (Inherited from Diagram.)
	allowUnconnectedLinks	Gets or sets a value indicating whether users are allowed to draw links that are not connected to any node. (Inherited from Diagram.)
	autoArrangeAvoidContainers	Gets or sets a value indicating whether auto-arranged link labels should avoid containers. (Inherited from Diagram.)
	autoArrangeAvoidSegments	Gets or sets a value indicating whether auto-arranged link labels should avoid link segments. (Inherited from Diagram.)
	autoCloseDistance	Gets or sets the maximum distance between first and last points of a FreeFormNode for which the node's outline is closed automatically. (Inherited from Diagram.)
	autoDeleteLinks	Gets or sets whether links are deleted automatically when their origin or destination is deleted. (Inherited from Diagram.)
	autoResize	Gets or sets a value indicating whether the document area should be resized automatically so it fits the diagram contents. (Inherited from Diagram.)
	backBrush	Gets or sets an object specifying how to fill the diagram background. (Inherited from Diagram.)

	backgroundImageAlign	Gets or sets a value indicating how to align the background image. (Inherited from Diagram.)
	backgroundImageContent	Gets or sets a Base64-encoded image to display as diagram background. (Inherited from Diagram.)
	backgroundImageUrl	Gets or sets URL of the image to paint as diagram's background. (Inherited from Diagram.)
	crossingRadius	Gets or sets the radius length of decorations displayed at link intersection points. (Inherited from Diagram.)
	customLinkType	Gets or sets the type of DiagramLink objects to create when users starts drawing. (Inherited from Diagram.)
	customNodeType	Gets or sets the type of DiagramNode objects to create when users starts drawing. (Inherited from Diagram.)
	defaultShape	Gets or sets the default shape assigned to new ShapeNode objects. (Inherited from Diagram.)
	dirty	Gets or sets a value indicating whether the diagram has changed since loading it. (Inherited from Diagram.)
	disabledHandlesStyle	Gets or sets the visual style of disabled adjustment handles. (Inherited from Diagram.)
	dynamicLinks	Gets or sets initial value of the dynamic property of new links. (Inherited from Diagram.)
	enableSpatialIndex	Gets or sets whether to index item positions for faster hit-testing and viewport clipping queries. (Inherited from Diagram.)
	expandButtonAction	Gets or sets how the control responds to users clickingthe +/- button displayed by expandable nodes. (Inherited from Diagram.)
	expandOnIncoming	Gets or sets a value indicating the link direction in which tree branches are expanded. (Inherited from Diagram.)
	factory	Gets a Factory object that lets you programmatically add new items to the diagram. (Inherited from Diagram.)
	font	Gets or sets the default font of text displayed by diagram items. (Inherited from Diagram.)
	freeFormAttractDistance	Gets or sets the radius around dragged free-form adjustment handle in which other points of **FreeFormNode** are modified too. (Inherited from Diagram.)
	freeFormTargets	Gets or sets an array of Shape objects used to replace **FreeFormNode** instances with ShapeNode ones when behavior is set to LinkFreeShapes or **DrawFreeShapes**. (Inherited from Diagram.)
	gridColor	Gets or sets the color of grid points. (Inherited from Diagram.)
	gridOffsetX	Gets or sets the horizontal offset of the first point of the alignment grid. (Inherited from Diagram.)
	gridOffsetY	Gets or sets the vertical offset of the first point of the alignment grid. (Inherited from Diagram.)

	gridPointSize	Gets or sets the size of shapes used to represent grid points. (Inherited from Diagram.)
	gridSizeX	Gets or sets the horizontal distance between adjacent grid points. (Inherited from Diagram.)
	gridSizeY	Gets or sets the vertical distance between adjacent grid points. (Inherited from Diagram.)
	gridStyle	Gets or sets the visual style of the alignment grid. (Inherited from Diagram.)
	iconUrl	Gets or sets URL of the image to paint as the page's icon.
	items	Returns an array of all items in this diagram. (Inherited from Diagram.)
	laneGrid	Gets the lane grid. (Inherited from Diagram.)
	linkBaseShape	Gets or sets the shape displayed at the base of new links. (Inherited from Diagram.)
	linkBaseShapeSize	Gets or sets the default size of arrowheads at base of new links. (Inherited from Diagram.)
	linkBrush	Gets or sets the default fill of link arrowheads. (Inherited from Diagram.)
	linkCrossings	Gets or sets a value indicating how to render the intersection points of links. (Inherited from Diagram.)
	linkHeadShape	Gets or sets the shape displayed as head of new links. (Inherited from Diagram.)
	linkHeadShapeSize	Gets or sets the default size of arrowheads. (Inherited from Diagram.)
	linkHitDistance	Gets or sets a value specifying how far from a link a click is still considered a hit. (Inherited from Diagram.)
	linkPen	Gets or sets the default stroke of links. (Inherited from Diagram.)
	linkRouter	Gets or sets the object used to find paths for auto-routed links when diagram nodes are added or their positions change. (Inherited from Diagram.)
	links	Returns an array of all links in this diagram. (Inherited from Diagram.)
	linkSegments	Gets or sets the default number of link segments. (Inherited from Diagram.)
	linkShape	Gets or sets the initial shape assigned to new links. (Inherited from Diagram.)
	linkTextStyle	Gets or sets the default text placement and orientation of new links. (Inherited from Diagram.)
	mergeThreshold	Gets or sets the maximum distance between adjacent control points of a link at which the respective segments can be merged. (Inherited from Diagram.)

	nodeEffects	Gets the global effects applied to nodes in this diagram. (Inherited from Diagram.)
	nodes	Returns an array of all nodes in this diagram. (Inherited from Diagram.)
	nodesExpandable	Gets or sets a value indicating whether newly created nodes are expandable. (Inherited from Diagram.)
	roundedLinks	Gets or sets a value indicating whether the joints between segments of Polyline and **Cascading** links should be rounded. (Inherited from Diagram.)
	roundedLinksRadius	Gets or sets the radius of joint arcs of rounded links' segments. (Inherited from Diagram.)
	routeLinks	Gets or sets initial value of the autoRoute property of new links. (Inherited from Diagram.)
	routeMargin	Gets or sets the minimal distance between nodes and auto-routed links. (Inherited from Diagram.)
	selectedItemHandlesStyle	Gets or sets the visual style of adjustment handles of selected items. (Inherited from Diagram.)
	selection	Gets a Selection instance that represents the selection of items in this diagram. (Inherited from Diagram.)
	shadowOffsetX	Gets or sets the horizontal offset of items' shadows. (Inherited from Diagram.)
	shadowOffsetY	Gets or sets the vertical offset of items' shadows. (Inherited from Diagram.)
	shadowsStyle	Gets or sets the style of shadows. (Inherited from Diagram.)
	shapeBrush	Gets or sets the default fill of **ShapeNode** objects. (Inherited from Diagram.)
	shapeLibraryLocation	Gets or sets the location of a shape library file containing custom shape definitions. It can be set to a full URL of the file, or a relative file path. (Inherited from Diagram.)
	shapePen	Gets or sets the default stroke of **ShapeNode** objects. (Inherited from Diagram.)
	showAnchors	Gets or sets a value indicating when to display anchor point marks. (Inherited from Diagram.)
	showDisabledHandles	Gets or sets a value indicating whether adjustment handles are painted when they are disabled. (Inherited from Diagram.)
	showGrid	Gets or sets a value indicating whether the alignment grid is visible. (Inherited from Diagram.)
	showLaneGrid	Gets or sets a value indicating whether the lane grid is visible. (Inherited from Diagram.)
	style	Gets or sets the style associated with this diagram. (Inherited from Diagram.)
	tag	Gets or sets custom data associated with this diagram. (Inherited from Diagram.)

	theme	Gets or sets the theme associated with this diagram. (Inherited from Diagram.)
	title	Gets or sets the page title.
	titleColor	Gets or sets the color of the title when the title is displayed.
	undoEnabled	Gets or sets a value indicating whether saving action records for later undo is enabled. (Inherited from Diagram.)
	validityChecks	Gets or sets a value indicating whether validity checks are performed each time an item is added to the diagram. (Inherited from Diagram.)
	version	Gets a string identifying current version of the library. (Inherited from Diagram.)

Public Methods

	Name	Description
	addItem	Adds an item to the **Diagram**. (Inherited from Diagram.)
	alignPointToGrid	Returns a point of the alignment grid nearest to the one passed as an argument. (Inherited from Diagram.)
	arrange	Arranges the **diagram** with the given layout. (Inherited from Diagram.)
	arrangeAnimated	Arranges the diagram using the given layout, and animates items moving to their new positions. (Inherited from Diagram.)
	arrangeLinkLabels	Rearranges link labels whose autoArrange property is enabled. (Inherited from Diagram.)
	cancelCompositeOperation	Stops recording undo/redo commands into a CompositeCommand and discards the records created so far. (Inherited from Diagram.)
	clearAll	Removes all items from the diagram. (Inherited from Diagram.)
	commitCompositeOperation	Stops recording undo/redo commands into a **CompositeCommand** and saves the composite in the history queue. (Inherited from Diagram.)
	executeCommand	Executes the specified command on this diagram. (Inherited from Diagram.)
	fromJson	Deserializes the diagram from a JSON string. (Inherited from Diagram.)
	fromXmlDocument	Loads diagram contents from specified XML Document. (Inherited from Diagram.)
	getContentBounds	Returns the smallest rectangle that bounds all diagram items. (Inherited from Diagram.)
	getItemAt	Finds and returns the topmost DiagramItem found at the specified location. (Inherited from Diagram.)

	getItemsAt	Finds and returns an array of the **diagram items** found at the specified location. (Inherited from Diagram.)
	getLinkAt	Finds and returns the topmost **DiagramLink** found at the specified location. (Inherited from Diagram.)
	getLinksAt	Finds and returns an array of the diagram **DiagramLink**s found at the specified location. (Inherited from Diagram.)
	getNearestNode	Returns the node nearest to the specified point. (Inherited from Diagram.)
	getNodeAt	Finds and returns the topmost **DiagramNode** found at the specified location. (Inherited from Diagram.)
	getNodesAt	Finds and returns an array of the **DiagramNode**s found at the specified location. (Inherited from Diagram.)
	isItemInteractive	Checks if the specified item can be modified interactively. (Inherited from Diagram.)
	isItemVisible	Checks if the specified item is currently visible. (Inherited from Diagram.)
	loadFromString	Loads diagram contents from specified XML or JSON string. (Inherited from Diagram.)
	loadFromXml	Loads the diagram from an XML file. (Inherited from Diagram.)
	raiseClicked	Raises an event which notifies that a Diagram item has been clicked. (Inherited from Diagram.)
	raiseCreated	Raises an event which notifies that a Diagram item has been created. (Inherited from Diagram.)
	raiseDeleted	Raises an event which notifies that a Diagram item has been deleted. (Inherited from Diagram.)
	raiseDeselected	Raises an event which notifies that a Diagram item has been deselected. (Inherited from Diagram.)
	raiseDoubleClicked	Raises an event which notifies that a Diagram item has been double-clicked. (Inherited from Diagram.)
	raiseModified	Raises an event which notifies that a Diagram item has been modified. (Inherited from Diagram.)
	raisePasted	Raises an event which notifies that a Diagram item has been pasted from clipboard. (Inherited from Diagram.)
	raiseSelected	Raises an event which notifies that a Diagram item has been selected. (Inherited from Diagram.)
	raiseSelecting	Raises an event which notifies that a Diagram item will be selected. (Inherited from Diagram.)
	raiseSelectingAsync	Raises an event which notifies that a Diagram item will be selected. (Inherited from DiagramBase.)
	raiseSelectionChanged	Raises an event which notifies that the selection has changed. (Inherited from Diagram.)

	redo	Executes again an action saved in the command history. (Inherited from Diagram.)
S	registerClass	Registers a diagram item class. (Inherited from Diagram.)
S	registerComponent	Registers a component class. (Inherited from Diagram.)
S	registerItemClass	Registers a diagram item class for XML serialization support. (Inherited from Diagram.)
	removeItem	Removes the specified **item** from the **Diagram**. (Inherited from Diagram.)
	resizeToFitItem	Resize the document extents so that the specified item is contained within. (Inherited from Diagram.)
	resizeToFitItems	Resizes the document scrollable area so that it fits all diagram items. (Inherited from Diagram.)
	routeAllLinks	Routes all links in the diagram in such a way they do not cross nodes. (Inherited from Diagram.)
	saveToString	Encodes the diagram contents into an XML or JSON string. (Inherited from Diagram.)
	saveToXml	Saves the diagram to an XML file. (Inherited from Diagram.)
	startCompositeOperation	Starts recording undo/redo commands into a **CompositeCommand**. (Inherited from Diagram.)
	toJson	Serializes the diagram into a JSON string. (Inherited from Diagram.)
	toXmlDocument	Saves the diagram into an XML Document. (Inherited from Diagram.)
	undo	Undoes an action saved in the command history. (Inherited from Diagram.)

Public Events

	Name	Description
	actionRedone	Raised when an action is redone. (Inherited from DiagramBase.)
	actionUndone	Raised when an action is undone. (Inherited from DiagramBase.)
	animatedLayoutCompleted	Raised when an animated layout completes. (Inherited from DiagramBase.)
	cellTextEdited	Raised when the user edits the text of a table cell. (Inherited from DiagramBase.)
	cellTextEditing	Raised when the user starts to edit the text of a table cell. (Inherited from DiagramBase.)
	clicked	Raised when the user clicks the diagram at a point where no items are located. (Inherited from DiagramBase.)
	containerChildAdded	Raised when the user adds a node to a ContainerNode. (Inherited from DiagramBase.)

	containerChildAdding	A validation event raised to let you prevent users from adding child nodes to a container. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	containerChildRemoved	Raised when the user removes a node from a **ContainerNode**. (Inherited from DiagramBase.)
	containerChildRemoving	Raised when the user removes a node from a **ContainerNode**. (Inherited from DiagramBase.)
	containerFolded	Raised when the user folds a container by clicking the arrow-up button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	containerPageChanged	Raised when the user changes current page of a PagedContainerNode. (Inherited from DiagramBase.)
	containerUnfolded	Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar. Event handlers receive a NodeEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	dataLoaded	Raised when the diagram data is loaded. (Inherited from DiagramBase.)
	deserializeTag	Raised when the Tag or Id of a diagram item must be deserialized from XML format. Event handlers receive a SerializeTagEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	drawAdjustmentHandles	Raised to draw custom adjustment handles. (Inherited from DiagramBase.)
	drawAnchorPoint	Raised when an anchor point must be custom-drawn. (Inherited from DiagramBase.)
	expandButtonClicked	Raised if **expandButtonAction** is set to RaiseEvents and the [+/-]button of an expandable node is clicked. (Inherited from DiagramBase.)
	headerResized	Raised when the user finished resizing a header. (Inherited from DiagramBase.)
	headerStartResizing	Raised when the user starts resizing a header. (Inherited from DiagramBase.)
	hitTestAdjustmentHandles	Raised to hit-test custom adjustment handles. (Inherited from DiagramBase.)
	hyperlinkClicked	Raised when the user clicks a hyperlink inside a node. (Inherited from DiagramBase.)
	initializeLink	Raised when the user starts drawing a new link, just after the DiagramLink instance is created. (Inherited from DiagramBase.)
	initializeNode	Raised when the user starts drawing a new node, just after the DiagramNode instance is created. (Inherited from DiagramBase.)
	itemAdded	Raised when an item is added to the diagram. (Inherited from DiagramBase.)

	itemRemoved	Raised when an item is removed from the diagram. (Inherited from DiagramBase.)
	laneGridCellTextEdited	Raised when the user edits the text of a grid cell. (Inherited from DiagramBase.)
	linkClicked	Raised when the user clicks a link. (Inherited from DiagramBase.)
	linkCreated	Raised when the user draws a new link. (Inherited from DiagramBase.)
	linkCreating	A validation event raised while the user is drawing a new link. (Inherited from DiagramBase.)
	linkDeleted	Raised when a link is deleted, either programmatically or by the user. (Inherited from DiagramBase.)
	linkDeleting	Raised when the user tries to delete a link. This event lets you cancel the operation. (Inherited from DiagramBase.)
	linkDeselected	Raised when a link is deselected, either programmatically or by the user. (Inherited from DiagramBase.)
	linkDoubleClicked	Raised when the user double-clicks a link. (Inherited from DiagramBase.)
	linkHovered	Raised when the mouse pointer hovers over a link. (Inherited from DiagramBase.)
	linkModified	Raised when the user moves any control point of a link. (Inherited from DiagramBase.)
	linkModifying	A validation event raised while the user is modifying a link. (Inherited from DiagramBase.)
	linkPasted	Raised when a link has been pasted from the clipboard. (Inherited from DiagramBase.)
	linkPointerEnter	Raised when the mouse pointer enters the boundaries of a link. (Inherited from DiagramBase.)
	linkPointerLeave	Raised when the mouse pointer leaves the boundaries of a link. (Inherited from DiagramBase.)
	linkRouted	Raised when a link is routed by the auto-routing algorithm. (Inherited from DiagramBase.)
	linkSelected	Raised when a link is selected, either programmatically or by the user. (Inherited from DiagramBase.)
	linkSelecting	Raised when the user tries to select a link. This event lets you cancel the operation. (Inherited from DiagramBase.)
	linkSplit	Raised when the user splits a link interactively. (Inherited from DiagramBase.)
	linkTextEdited	Raised when the user edits the text of a link. (Inherited from DiagramBase.)
	linkTextEditing	Raised when the user starts to edit the text of a link. (Inherited from DiagramBase.)

	nodeClicked	Raised when the user clicks a node. (Inherited from DiagramBase.)
	nodeCreated	Raised when the user draws a new node. (Inherited from DiagramBase.)
	nodeCreating	A validation event raised while the user is drawing a new node. (Inherited from DiagramBase.)
	nodeDeleted	Raised when a node is deleted, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeDeleting	Raised when the user tries to delete a node. This event lets you cancel the operation. (Inherited from DiagramBase.)
	nodeDeselected	Raised when a node is deselected, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeDomCreated	Raised when the content of a ControlNode is added to the page DOM. (Inherited from DiagramBase.)
	nodeDoubleClicked	Raised when the user double-clicks a node. (Inherited from DiagramBase.)
	nodeHovered	Raised when the mouse pointer hovers over a node. (Inherited from DiagramBase.)
	nodeModified	Raised when the user moves or resizes a node. (Inherited from DiagramBase.)
	nodeModifying	A validation event raised while the user is moving or resizing a node. (Inherited from DiagramBase.)
	nodePasted	Raised when a node has been pasted from the clipboard. (Inherited from DiagramBase.)
	nodePointerEnter	Raised when the mouse pointer enters the boundaries of a node. (Inherited from DiagramBase.)
	nodePointerLeave	Raised when the mouse pointer leaves the boundaries of a node. (Inherited from DiagramBase.)
	nodeSelected	Raised when a node is selected, either programmatically or by the user. (Inherited from DiagramBase.)
	nodeSelecting	Raised when the user tries to select a node. This event lets you cancel the operation. (Inherited from DiagramBase.)
	nodeTextEdited	Raised when the user edits the text of a node. (Inherited from DiagramBase.)
	nodeTextEditing	Raised when the user starts to edit the text of a node. (Inherited from DiagramBase.)
	queryView	Raised to enumerate diagram views. (Inherited from DiagramBase.)
	selectionChanged	Raised when the diagram selection has changed. (Inherited from DiagramBase.)
	selectionModifying	A validation event raised while the user is moving or resizing multiple items. (Inherited from DiagramBase.)
	serializeTag	Raised when the Tag or Id of a diagram item must be serialized in XML format. Event handlers receive a

		SerializeTagEventArgs instance that contains data about this event. (Inherited from DiagramBase.)
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from **shapeLibraryLocation**. (Inherited from DiagramBase.)
	sizeChanged	Raised when the size of the diagram canvas has changed. (Inherited from DiagramBase.)
	tableColumnResized	Raised when the user finishes resizing a column in a table. (Inherited from DiagramBase.)
	tableColumnResizing	Raised when the user resizes a column in a table interactively. (Inherited from DiagramBase.)
	tableRowResized	Raised when the user finishes resizing a row in a table. (Inherited from DiagramBase.)
	tableRowResizing	Raised when the user resizes a row in a table interactively. (Inherited from DiagramBase.)
	tableSectionCollapsed	Raised when a table section is collapsed. (Inherited from DiagramBase.)
	tableSectionExpanded	Raised when a table section is expanded. (Inherited from DiagramBase.)
	treeCollapsed	Raised when a user collapses a tree branch by clicking the [-] button of an **expandable** node. (Inherited from DiagramBase.)
	treeExpanded	Raised when a user expands a tree branch by clicking the [+] button of an **expandable** node. (Inherited from DiagramBase.)
	treeItemTextEdited	Raised when the user edits the text of a tree view item. (Inherited from DiagramBase.)
	treeItemTextEditing	Raised when the user starts to edit the text of a tree view item. (Inherited from DiagramBase.)
	validateAnchorPoint	A validation event that lets you approve or reject connecting a link to an anchor point. (Inherited from DiagramBase.)
	validateLinkCrossing	A validation event that lets you approve or reject displaying a decoration at the intersection point of links. (Inherited from DiagramBase.)

See Also

DiagramPage Class
MindFusion.Diagramming Namespace

DiagramPage Constructor

Initializes a new instance of the DiagramPage class.

Namespace: MindFusion.Diagramming
File: DiagramPage.js

Syntax

```
JavaScript
function DiagramPage ([title])
```

Parameters

title
Optional.
String. The page title.

See Also

DiagramPage Class
MindFusion.Diagramming Namespace

DiagramPage Properties

DiagramPage.iconUrl Property

Gets or sets URL of the image to paint as the page's icon.

Namespace: MindFusion.Diagramming
File: DiagramPage.js

Syntax

```
JavaScript
get iconUrl() {}
set iconUrl(value) {}
```

Property Value

String

A string containing image URL.

See Also

DiagramPage Members
DiagramPage Class
MindFusion.Diagramming Namespace

DiagramPage.title Property

Gets or sets the page title.

Namespace: MindFusion.Diagramming
File: DiagramPage.js

Syntax

```
JavaScript
get title() {}
set title(value) {}
```

Property Value

String

A string specifying the page title.

See Also

DiagramPage Members
DiagramPage Class
MindFusion.Diagramming Namespace

DiagramPage.titleColor Property

Gets or sets the color of the title when the title is displayed.

Namespace: MindFusion.Diagramming
File: DiagramPage.js

Syntax

JavaScript

```
get titleColor() {}
set titleColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

DiagramPage Members
DiagramPage Class
MindFusion.Diagramming Namespace

DiagramPage Events

DiagramView Class

Represents a control that displays the content of a Diagram object.

Syntax

JavaScript

```
// class
DiagramView.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CanvasControl
MindFusion.Diagramming.DiagramView

See Also

DiagramView Members
MindFusion.Diagramming Namespace

DiagramView Members

Represents a control that displays the content of a Diagram object.

The following tables list the members exposed by the DiagramView type.

Public Constructors

	Name	Description
	DiagramView	Initializes a new instance of the DiagramView class.

Public Properties

	Name	Description
	allowInplaceEdit	Gets or sets whether users are allowed to edit the text of diagram items.
	autoScroll	Gets or sets a value indicating whether automatic scrolling is enabled.
	autoScrollAmount	Gets or sets the amount by which to auto-scroll the view when the mouse leaves the view boundaries while drawing.
	behavior	Gets or sets a value indicating how the control responds to user actions.
	defaultControlTemplate	Gets or sets HTML string representing the DOM content rendered by default by newly created ControlNode objects.
	delKeyAction	Gets or sets what action to perform when users hit the Del key.
	diagram	Gets or sets the diagram displayed by this view.
	dragIndicatorOpacity	Gets or sets the opacity of ghost image displayed during drag-and-drop from NodeListView.
	enableMultiTouch	Gets or sets a value indicating whether the control handles DOM Pointer events to implement multi-touch interactions, such as zooming, node rotation or simultaneous drawing of multiple diagram items.
	handleTouchHitDistance	Gets or sets maximum distance between touch position and adjustment handle at which the handle can still be grabbed to modify its item.
	horizontalDropAlignment	Gets or sets the horizontal placement of nodes created by drag-and-drop operations, relatively to mouse cursor.
	hoverDelay	Gets or sets the time in milliseconds that mouse pointer should remain stationary before reporting hover events.
	inplaceEditAcceptOnEnter	Gets or sets a value indicating whether pressing the ENTER keyshould exit inplace-edit mode and accept changes made

		to text.
	inplaceEditCancelOnEsc	Gets or sets a value indicating whether pressing the ESC keyshould exit inplace-edit mode and reject changes made to text.
	leftButtonActions	Gets or sets a combination of flags that specify what actions can be performed via the left mouse button.
	licenseLocation	Gets or sets URL of the control's license file.
	magnifierEnabled	Gets or sets a value indicating whether the magnifier tool is currently enabled.
	magnifierFactor	Gets or sets the zoom factor of the magnifier tool.
	magnifierFrameColor	Gets or sets the color of the magnifier tool's frame.
	magnifierFrameThickness	Gets or sets the thickness of the magnifier frame.
	magnifierHeight	Gets or sets the height of the magnifier tool.
	magnifierSecondaryFrameColor	Gets or sets the color of the magnifier tool's secondary frame.
	magnifierShading	Gets or sets whether to enhance the effect of a magnifier lense by using a gradient shading.
	magnifierShape	Gets or sets a reference to the geometric shape definition of the magnifier tool.
	magnifierWidth	Gets or sets the width of the magnifier tool.
	middleButtonActions	Gets or sets a combination of flags that specify what actions can be performed via the middle mouse button.
	modificationStart	Gets or sets how users can start modifying diagram items.
	modifierKeyActions	Gets an object that allows changing the function of keyboard modifier keys such as Ctrl and Alt.
	mouseWheelAction	Gets or sets what action to perform when users rotate the mouse wheel.
	multiTouchDraw	Gets or sets a value indicating whether the control allows drawing multiple items simultaneously.
	multiTouchModify	Gets or sets a value indicating whether multi-touch modifications are enabled.
	multiTouchZoom	Gets or sets a value indicating whether multi-touch zoom is enabled.
	rightButtonActions	Gets or sets a combination of flags that specify what actions can be performed via the right mouse button.
	scrollX	Gets or sets the diagram's horizontal scroll position.
	scrollY	Gets or sets the diagram's vertical scroll position.
	scrollZoneSize	Gets or sets the size of the auto scroll zone near the edges of the control.
	tooltipDelay	Gets or sets the delay before a tooltip is shown.

	verticalDropAlignment	Gets or sets the vertical placement of nodes created by drag-and-drop operations, relatively to mouse cursor.
	viewport	Returns the visible portion of the diagram area.
	virtualScroll	Gets or sets a value indicating whether virtual scroll mode is enabled.
	zoomFactor	Gets or sets the zoom factor.

Public Methods

	Name	Description
	animate	Animates the view state.
	beginEdit	Begins in-place editing of the specified object.
	bringIntoView	Scrolls the view to make the specified item visible.
	bringRectIntoView	Scrolls the view to make the specified rectangular area visible.
	cancelDrag	Cancels all current drag operations.
	clearTooltip	Forces the removal of any created tooltips from the diagram canvas.
	copyToClipboard	Copies the current selection of items to the clipboard.
S	create	Creates and initializes a new **DiagramView** from the specified element. This method is static and can be called without creating an instance of the class.
	cutToClipboard	Cuts the currently selected items to the clipboard.
	dispose	Control.dispose override.
	endEdit	Exits the in-place editing mode and either accepts or rejects the changes made to the item's text.
S	find	Returns the specified **DiagramView** object. This member is static and can be invoked without creating an instance of the class.
	fromJson	Loads the diagram from a JSON string.
	initialize	Control.initialize override.
	loadFromXml	Loads the diagram from an XML file.
	pasteFromClipboard	Pastes diagram items from the clipboard.
	print	Exports the diagram as a list of images to a new HTML document and opens the browser's print dialog.
	printPreview	Exports the diagram as a list of images to a new HTML document.
	record	Starts recording mouse and keyboard events for subsequent replay.

	replay	Replays a sequence of recorded mouse and keyboard events created by record and stopRecording methods.
	saveToXml	Saves the diagram to an XML file.
	scrollTo	Scrolls the diagram to the specified point.
	setScroll	IZoomTarget implementation
	setZoomFactorPivot	Sets the zoom factor, preserving the on-screen position of the specified diagram point.
	stopRecording	Stops recording mouse and keyboard events.
	toJson	Saves the diagram to a JSON string.
	zoomToFit	Zooms the view to fit the **Diagram** contents in component's area.
	zoomToRect	Zooms and scrolls the view to fit the specified **Diagram** region in component's area.

Public Events

	Name	Description
	createEditControl	Raised to let you create a custom inplace editor control.
	diagramChanged	Raised when the **diagram** property changes.
	enterInplaceEditMode	Raised when entering inplace-edit mode.
	leaveInplaceEditMode	Raised when leaving inplace-edit mode.
	mouseWheel	Raised when the user rotates the mouse wheel while the cursor is positioned over the diagram surface.
	zoomChanged	Raised when the zoom factor property changes.

See Also

DiagramView Class
MindFusion.Diagramming Namespace

DiagramView Constructor

Initializes a new instance of the DiagramView class.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function DiagramView (element, [diagram])
```

Parameters

element

HTMLCanvasElement. The Canvas DOM Element this DiagramView is associated with.

diagram
Optional.
Diagram. The Diagram instance this control is associated with.

See Also

DiagramView Class
MindFusion.Diagramming Namespace

DiagramView Methods

DiagramView.animate Method

Animates the view state.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function animate (animation)
```

Parameters

animation
ViewAnimation. A **ViewAnimation** -derived object.

Example

This example centers specified node in the viewport by running a ScrollAnimation:

JavaScript

```
var endNode = diagram.nodes.find(
    n => n.id == "end");

if (endNode != null)
{
    var viewSize = diagramView.viewport.getSize();
    var newScroll = new Point(
        endNode.center.x - viewSize.width / 2,
        endNode.center.y - viewSize.height / 2);

    var currentScroll = new Point(
        diagramView.scrollX, diagramView.scrollY);

    diagramView.animate(
        new ScrollAnimation(
        {
            fromValue: currentScroll,
            toValue: newScroll,
            animationType: Animations.AnimationType.Power,
            easingType: Animations.EasingType.EaseInOut
        }));
}
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.beginEdit Method

Begins in-place editing of the specified object.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function beginEdit (item, [point, [editTarget]])
```

Parameters

item
InplaceEditable. The DiagramItem whose text should be edited.

point
Optional.
Point. A Point specifying where the in-place edit box should appear.

editTarget
Optional.
Cell | LinkLabel. The element to use as a target for the inplace edit.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.bringIntoView Method

Scrolls the view to make the specified item visible.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function bringIntoView (item)
```

Parameters

item
DiagramItem. The item to bring into view.

Remarks

Scrolls the diagram view to ensure that the specified item is visible. No scrolling is done if the item is already inside the viewport.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
scrollTo Method
bringRectIntoView Method
scrollX Property
scrollY Property
zoomToRect Method

DiagramView.bringRectIntoView Method

Scrolls the view to make the specified rectangular area visible.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function bringRectIntoView (rect)
```

Parameters

rect
Rect. A Rect specifying the diagram region to show in the viewport.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.cancelDrag Method

Cancels all current drag operations.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function cancelDrag ()
```

Remarks

This method is useful in situations when the current draw operations must be cancelled, for example when starting external drag-and-drop from **pointerMove** DOM event handlers, or in response to Esc keypress. Multiple operations could be running simultaneously when enableMultiTouch is set to **true**, and this method cancels them all. Individual operations can be cancelled by calling cancelDrag method of event arguments when handling validation events, such as nodeCreating.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.clearTooltip Method

Forces the removal of any created tooltips from the diagram canvas.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function clearTooltip ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.copyToClipboard Method

Copies the current selection of items to the clipboard.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function copyToClipboard ([systemClipboard])
```

Parameters

systemClipboard
Optional.
Boolean. **true** to copy to the system clipboard, or **false** otherwise. The default value is **false**.

Remarks

The diagram control maintains its own local clipboard and items are copied to it by default. If you specify a **true** argument and running under browsers that allow access to the system clipboard, the items are copied to the system clipboard; in all other cases they are copied to the local clipboard.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

cutToClipboard Method
pasteFromClipboard Method

DiagramView.create Method

Creates and initializes a new DiagramView from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function create (element, [diagram])
```

Parameters

element
HTMLCanvasElement. The DOM <canvas> element that the DiagramView should be attached to.

diagram
Optional.
Diagram. The Diagram instance that should be rendered by the new view.

Return Value

DiagramView. The newly created **DiagramView** object.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
find Method
WebComponents.register Method

DiagramView.cutToClipboard Method

Cuts the currently selected items to the clipboard.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function cutToClipboard ([systemClipboard])
```

Parameters

systemClipboard
Optional.
Boolean. **true** to copy to the system clipboard, or **false** otherwise. The default value is **false**.

Remarks

The diagram control maintains its own local clipboard and items are copied to it by default. If you specify a **true** argument and running under browsers that allow access to the system clipboard, the items are copied to the system clipboard; in all other cases they are copied to the local clipboard.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
copyToClipboard Method
pasteFromClipboard Method

DiagramView.dispose Method

Control.dispose override.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function dispose ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.endEdit Method

Exits the in-place editing mode and either accepts or rejects the changes made to the item's text.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function endEdit ([accept])
```

Parameters

accept
Optional.
Boolean. **true** to accept changes made to text, or **false** to reject them.

See Also

DiagramView Members
DiagramView Class

MindFusion.Diagramming Namespace

DiagramView.find Method

Returns the specified DiagramView object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function find (id)
```

Parameters

id
String. The id of the DiagramView's element.

Return Value

DiagramView. A **DiagramView** object with the specified id, if found, or **null** otherwise.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
create Method

DiagramView.fromJson Method

Loads the diagram from a JSON string.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.initialize Method

Control.initialize override.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function initialize ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.loadFromXml Method

Loads the diagram from an XML file.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function loadFromXml (fileUrl, [onLoad, [onError]])
```

Parameters

fileUrl
String. The URL of an XML file where the data should be read from.

onLoad
Optional.
function. A callback that should be invoked if the file is loaded successfully.

onError
Optional.
function. A callback that should be invoked if the file could not be downloaded.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.pasteFromClipboard Method

Pastes diagram items from the clipboard.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
```

```
function pasteFromClipboard (offsetX, offsetY, [systemClipboard, [unconnectedLinks]])
```

Parameters

offsetX
Number. Horizontal offset of pasted items from their original positions.

offsetY
Number. Vertical offset of pasted items from their original positions.

systemClipboard
Optional.
Boolean. **true** to paste from the system clipboard, or **false** otherwise. The default value is **false**.

unconnectedLinks
Optional.
Boolean. **true** to allow pasting links whose origin or destination node has not been copied, or **false** otherwise. The default value is **false**.

Remarks

The diagram control maintains its own local clipboard and items are copied to it by default. If you specify **true** for the *systemClipboard* argument and running under browsers that allow access to the system clipboard, the items are copied to the system clipboard; in all other cases they are copied to the local clipboard.

Due to **Navigator.clipboard** API being asynchronous, pasted items are not available immediately after calling this method with *systemClipboard* enabled. You will be able to access them only after nodePasted and linkPasted events are raised.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
copyToClipboard Method
cutToClipboard Method

DiagramView.print Method

Exports the diagram as a list of images to a new HTML document and opens the browser's print dialog.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function print ([options])
```

Parameters

options
Optional.
Object. An object, containing customization properties, including printArea (defaults to diagram.getContentBounds()) and pageSize (defaults to diagramView.viewport).

See Also

DiagramView.printPreview Method

Exports the diagram as a list of images to a new HTML document.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function printPreview ([options])
```

Parameters

options
Optional.
Object. An object, containing customization properties, including printArea (defaults to diagram.getContentBounds()) and pageSize (defaults to diagramView.viewport).

Example

JavaScript

```
document.getElementById("printPreview").addEventListener("click", () => {
    var diagramView = DiagramView.find("diagram");

    // preferred image size is A4 with subtracted margins
    var A4 = new Size(210, 297);
    A4.width -= 25.4;
    A4.height -= 25.4;

    diagramView.printPreview(
        {
            printArea: diagramView.diagram.bounds,
            pageSize: A4
        });
});
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
print Method

DiagramView.record Method

Starts recording mouse and keyboard events for subsequent replay.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function record ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
replay Method
stopRecording Method

DiagramView.replay Method

Replays a sequence of recorded mouse and keyboard events created by record and stopRecording methods.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function replay (recording, timer, [savedState, [timerInterval]])
```

Parameters

recording
Array. An array of event records returned by stopRecording.

timer
Boolean. **true** to replay records using a timer, preserving their original relative timing, or **false** to replay them immediately.

savedState
Optional.
String. A diagram JSON string that should be loaded before replaying recorded events.

timerInterval
Optional.
Number. A number of milliseconds to use as a constant interval instead of relative timing.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
record Method
stopRecording Method

DiagramView.saveToXml Method

Saves the diagram to an XML file.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function saveToXml (url)
```

Parameters

url
String. A URL specifying where the diagram's XML should be posted to.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.scrollTo Method

Scrolls the diagram to the specified point.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function scrollTo (x, y)
```

Parameters

x
Number. A number specifying the new horizontal scroll position in the current unit of measure.

y
Number. A number specifying the new vertical scroll position in the current unit of measure.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
bringIntoView Method

DiagramView.setScroll Method

IZoomTarget implementation

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function setScroll ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.setZoomFactorPivot Method

Sets the zoom factor, preserving the on-screen position of the specified diagram point.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function setZoomFactorPivot (zoomFactor, pivotPoint)
```

Parameters

zoomFactor
Number. The new zoom factor.

pivotPoint
Point. A Point instance specifying the zoom center.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.stopRecording Method

Stops recording mouse and keyboard events.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
function stopRecording ()
```

Return Value

Array. An array of event records representing user's input.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

record Method
replay Method

DiagramView.toJson Method

Saves the diagram to a JSON string.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function toJson ()
```

Return Value

String. A string containing the diagram's JSON representation.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.zoomToFit Method

Zooms the view to fit the Diagram contents in component's area.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
function zoomToFit ()
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.zoomToRect Method

Zooms and scrolls the view to fit the specified Diagram region in component's area.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
```

```
function zoomToRect (rect)
```

Parameters

rect
Rect. A Rect instance specifying desired viewport position and size.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
zoomFactor Property
bringRectIntoView Method
bringIntoView Method

DiagramView Properties

DiagramView.allowInplaceEdit Property

Gets or sets whether users are allowed to edit the text of diagram items.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get allowInplaceEdit() {}
set allowInplaceEdit(value) {}
```

Property Value

Boolean

true to allow in-place editing, or **false** otherwise. The default value is **false**.

Remarks

Specifies whether users are allowed to edit text of diagram nodes in-place. If enabled, a text-entry field appears at node's position when the node is double-clicked. In-place editing mode can be exited by clicking anywhere outside the text field.

To enter or exit in-place editing mode programmatically, call the beginEdit and endEdit methods of DiagramView.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
inplaceEditAcceptOnEnter Property
inplaceEditCancelOnEsc Property
nodeTextEdited Event
linkTextEdited Event
enterInplaceEditMode Event

DiagramView.autoScroll Property

Gets or sets a value indicating whether automatic scrolling is enabled.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get autoScroll() {}
set autoScroll(value) {}
```

Property Value

Boolean

true to enable auto scrolling, or **false** otherwise. The default value is **true**.

Remarks

If enabled, the control will scroll its content automatically when the user drags the mouse near the viewport edges. This works only if the diagram view is placed inside a DIV whose **overflow** attribute is set to **scroll** or **auto**.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
scrollZoneSize Property
scrollX Property
scrollY Property
bringIntoView Method

DiagramView.autoScrollAmount Property

Gets or sets the amount by which to auto-scroll the view when the mouse leaves the view boundaries while drawing.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get autoScrollAmount() {}
set autoScrollAmount(value) {}
```

Property Value

Number

A number specifying the amount by which to auto-scroll.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.behavior Property

Gets or sets a value indicating how the control responds to user actions.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get behavior() {}
set behavior(value) {}
```

Property Value

Behavior

A member of the Behavior enumeration. The default value is **LinkShapes**.

Remarks

The property specifies how the control responds to user actions with the mouse. The default value, LinkShapes, facilitates drawing of flowchart diagrams. **LinkTables** makes JsDiagram create table nodes and links while the user draws with the mouse, allowing creation of entity-relationship or class-hierarchy diagrams.

Modes such as **Modify**, **PanAndModify**, **MoveNodes**, **SelectOnly** allow users to interact with existing items, but not draw new ones on the canvas. Note that NodeListView can be used to create new nodes regardless of DiagramView's current behavior mode.

There are several more values that can be assigned to the property, that make the control draw specific types of items, for example only shape nodes, only free-form nodes, only links, and so on. These are useful when implementing the ability to select specific drawing tools in applications.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.defaultControlTemplate Property

Gets or sets HTML string representing the DOM content rendered by default by newly created ControlNode objects.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get defaultControlTemplate() {}
set defaultControlTemplate(value) {}
```

Property Value

String

An HTML string.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.delKeyAction Property

Gets or sets what action to perform when users hit the Del key.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get delKeyAction() {}
set delKeyAction(value) {}
```

Property Value

DelKeyAction

A member of the DelKeyAction enumeration. The default value is **DeleteSelectedItems**.

Remarks

Depending on this property value, the control can delete all selected items, delete only the active item, or ignore the Del key press.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
nodeDeleting Event
linkDeleting Event
autoDeleteLinks Property
showDeleteButton Property

DiagramView.diagram Property

Gets or sets the diagram displayed by this view.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get diagram() {}
set diagram(value) {}
```

Property Value

Diagram

The Diagram instance displayed by this view.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.dragIndicatorOpacity Property

Gets or sets the opacity of ghost image displayed during drag-and-drop from NodeListView.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get dragIndicatorOpacity() {}
set dragIndicatorOpacity(value) {}
```

Property Value

Number

A number specifying opacity. The default value is 0.4.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.enableMultiTouch Property

Gets or sets a value indicating whether the control handles DOM Pointer events to implement multi-touch interactions, such as zooming, node rotation or simultaneous drawing of multiple diagram items.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get enableMultiTouch() {}
set enableMultiTouch(value) {}
```

Property Value

Boolean

true to enable multi-touch, or **false** otherwise. The default value is **true**.

Remarks

Set this property to **false** if you need to handle touch events yourself or through a third-party gesture library. If disabled, the control will revert to its mouse event handling code from previous versions. It will also no longer be able to capture the mouse input during drag operations (to detect mouse pointer leaving the view), which is handled through DOM Pointers API.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
multiTouchModify Property
multiTouchZoom Property
multiTouchDraw Property

DiagramView.handleTouchHitDistance Property

Gets or sets maximum distance between touch position and adjustment handle at which the handle can still be grabbed to modify its item.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get handleTouchHitDistance() {}
set handleTouchHitDistance(value) {}
```

Property Value

Number

A number specifying the allowed hit distance, in diagram's current unit. The default value is 5.

Remarks

This value is considered only when handling touch events.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.horizontalDropAlignment Property

Gets or sets the horizontal placement of nodes created by drag-and-drop operations, relatively to mouse cursor.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get horizontalDropAlignment() {}
set horizontalDropAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration. The default value is **Near**.

Remarks

This property lets you center the new node or align specified edge to the mouse pointer position when drag-and-drop ends.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
verticalDropAlignment Property

DiagramView.hoverDelay Property

Gets or sets the time in milliseconds that mouse pointer should remain stationary before reporting hover events.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get hoverDelay() {}
set hoverDelay(value) {}
```

Property Value

Number

The hover delay in milliseconds. The default value is 1500.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
nodeHovered Event
linkHovered Event

DiagramView.inplaceEditAcceptOnEnter Property

Gets or sets a value indicating whether pressing the ENTER keyshould exit inplace-edit mode and accept changes made to text.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get inplaceEditAcceptOnEnter() {}
set inplaceEditAcceptOnEnter(value) {}
```

Property Value

Boolean

true to end inplace editing and accept changes when ENTER is pressed, or **false** otherwise. The default value is **false**.

Remarks

If the value of this property is **false**, pressing the ENTER key inserts a new line in the text while in inplace-edit mode.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property
inplaceEditCancelOnEsc Property

DiagramView.inplaceEditCancelOnEsc Property

Gets or sets a value indicating whether pressing the ESC keyshould exit inplace-edit mode and reject changes made to text.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get inplaceEditCancelOnEsc() {}
set inplaceEditCancelOnEsc(value) {}
```

Property Value

Boolean

true to end inplace editing and cancel changes when ESC is pressed, or **false** otherwise. The default value is **true**.

Remarks

If the value of this property is **false**, pressing the ESC key has no effect while in inplace-edit mode.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property
inplaceEditAcceptOnEnter Property

DiagramView.leftButtonActions Property

Gets or sets a combination of flags that specify what actions can be performed via the left mouse button.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get leftButtonActions() {}
set leftButtonActions(value) {}
```

Property Value

MouseButtonActions

A combination of MouseButtonActions members.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.licenseLocation Property

Gets or sets URL of the control's license file.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get licenseLocation() {}
set licenseLocation(value) {}
```

Property Value

String

A URL string that specifies the location of the license key.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierEnabled Property

Gets or sets a value indicating whether the magnifier tool is currently enabled.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get magnifierEnabled() {}
set magnifierEnabled(value) {}
```

Property Value

Boolean

true if the magnifier is enabled, or **false** otherwise. The default value is **false**.

Remarks

The magnifier tool can be alternatively enabled on-demand through a modifier key specified by modifierKeyActions, or a mouse button specified in leftButtonActions, middleButtonActions, rightButtonActions.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierFactor Property

Gets or sets the zoom factor of the magnifier tool.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get magnifierFactor() {}
set magnifierFactor(value) {}
```

Property Value

Number

The zoom factor of the magnifier tool as percentage.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierFrameColor Property

Gets or sets the color of the magnifier tool's frame.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get magnifierFrameColor() {}
set magnifierFrameColor(value) {}
```

Property Value

String

The color of the magnifier tool's frame.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierFrameThickness Property

Gets or sets the thickness of the magnifier frame.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get magnifierFrameThickness() {}
set magnifierFrameThickness(value) {}
```

Property Value

Number

A number specifying the thickness of the magnifier frame.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierHeight Property

Gets or sets the height of the magnifier tool.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get magnifierHeight() {}
set magnifierHeight(value) {}
```

Property Value

Number

A number specifying the height of the magnifier tool.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierSecondaryFrameColor Property

Gets or sets the color of the magnifier tool's secondary frame.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get magnifierSecondaryFrameColor() {}
set magnifierSecondaryFrameColor(value) {}
```

Property Value

String

The color of the magnifier tool's secondary frame.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierShading Property

Gets or sets whether to enhance the effect of a magnifier lense by using a gradient shading.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get magnifierShading() {}
set magnifierShading(value) {}
```

Property Value

Boolean

true to enable shading, or **false** otherwise.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierShape Property

Gets or sets a reference to the geometric shape definition of the magnifier tool.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get magnifierShape() {}
set magnifierShape(value) {}
```

Property Value

Shape | *String*

The Shape or shape identifier of the magnifier tool.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.magnifierWidth Property

Gets or sets the width of the magnifier tool.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get magnifierWidth() {}
set magnifierWidth(value) {}
```

Property Value

Number

A number specifying the width of the magnifier tool.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.middleButtonActions Property

Gets or sets a combination of flags that specify what actions can be performed via the middle mouse button.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get middleButtonActions() {}
set middleButtonActions(value) {}
```

Property Value

MouseButtonActions

A combination of MouseButtonActions members.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.modificationStart Property

Gets or sets how users can start modifying diagram items.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get modificationStart() {}
set modificationStart(value) {}
```

Property Value

ModificationStart

A member of the ModificationStart enumeration.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.modifierKeyActions Property

Gets an object that allows changing the function of keyboard modifier keys such as Ctrl and Alt.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get modifierKeyActions() {}
```

Property Value

ModifierKeyActions

An instance of the ModifierKeyActions class.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.mouseWheelAction Property

Gets or sets what action to perform when users rotate the mouse wheel.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get mouseWheelAction() {}
set mouseWheelAction(value) {}
```

Property Value

MouseWheelAction

A member of the MouseWheelAction enumeration. The default value is **Scroll**.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
mouseWheel Event

DiagramView.multiTouchDraw Property

Gets or sets a value indicating whether the control allows drawing multiple items simultaneously.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get multiTouchDraw() {}
set multiTouchDraw(value) {}
```

Property Value

Boolean

true to allow drawing multiple items simultaneously, or **false** otherwise. The default value is **true**.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
multiTouchModify Property
multiTouchZoom Property
enableMultiTouch Property

DiagramView.multiTouchModify Property

Gets or sets a value indicating whether multi-touch modifications are enabled.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get multiTouchModify() {}
set multiTouchModify(value) {}
```

Property Value

Boolean

true to enable multi-touch node modification, or **false** otherwise. The default value is **true**.

Remarks

If this property is enabled, two touches on same node will scale and rotate it. Otherwise each touch is handled independently according to current behavior value, e.g. by drawing a new diagram link.

Multi-touch rotation is applied only if the Rotate handle bit in enabledHandles is set.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
multiTouchZoom Property
multiTouchDraw Property
enableMultiTouch Property

DiagramView.multiTouchZoom Property

Gets or sets a value indicating whether multi-touch zoom is enabled.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get multiTouchZoom() {}
set multiTouchZoom(value) {}
```

Property Value

Boolean

true to enable multi-touch zoom, or **false** otherwise. The default value is **true**.

Remarks

If this property is disabled, each touch is handled independently according to current behavior value, e.g. by drawing a new diagram item or moving an existing one.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
multiTouchModify Property
enableMultiTouch Property

DiagramView.rightButtonActions Property

Gets or sets a combination of flags that specify what actions can be performed via the right mouse button.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get rightButtonActions() {}
set rightButtonActions(value) {}
```

Property Value

MouseButtonActions

A combination of MouseButtonActions members.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.scrollX Property

Gets or sets the diagram's horizontal scroll position.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get scrollX() {}
set scrollX(value) {}
```

Property Value

Number

A number specifying the horizontal scroll position in the current unit of measure.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
scrollY Property
autoScroll Property
scrollTo Method
bringIntoView Method

DiagramView.scrollY Property

Gets or sets the diagram's vertical scroll position.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get scrollY() {}
set scrollY(value) {}
```

Property Value

Number

A number specifying the vertical scroll position in the current unit of measure.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
scrollX Property
autoScroll Property
scrollTo Method
bringIntoView Method

DiagramView.scrollZoneSize Property

Gets or sets the size of the auto scroll zone near the edges of the control.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get scrollZoneSize() {}
set scrollZoneSize(value) {}
```

Property Value

Number

A number specifying the size of the auto scroll zone. The default value is 8.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
autoScroll Property

DiagramView.tooltipDelay Property

Gets or sets the delay before a tooltip is shown.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get tooltipDelay() {}
set tooltipDelay(value) {}
```

Property Value

Number

The tooltip delay in milliseconds.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.verticalDropAlignment Property

Gets or sets the vertical placement of nodes created by drag-and-drop operations, relatively to mouse cursor.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get verticalDropAlignment() {}
set verticalDropAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration. The default value is **Near**.

Remarks

This property lets you center the new node or align specified edge to the mouse pointer position when drag-and-drop ends.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
horizontalDropAlignment Property

DiagramView.viewport Property

Returns the visible portion of the diagram area.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
get viewport() {}
```

Property Value

Rect

A Rect instance specifying the viewport coordinates.

Remarks

When the view's canvas element is placed inside a scrollable div element, this method returns what part of the diagram is currently visible within the div. The result is specified in the current measure unit.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.virtualScroll Property

Gets or sets a value indicating whether virtual scroll mode is enabled.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get virtualScroll() {}
set virtualScroll(value) {}
```

Property Value

Boolean

true if virtual scroll is enabled, or **false** otherwise. The default value is **true**.

Remarks

If this mode is enabled, the canvas stays as big as its containing div, and its content is drawn with a translation bound to the div's scrollbar positions. This allow displaying large diagrams in browsers and on mobile devices where the size of HTML canvas elements is limited.

Virtual scroll mode requires browser support for grid layout. If you need to use it in older browsers, set CompatConfig.gridLayout = false to use absolute positioning instead.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
CompatConfig.gridLayout Field

DiagramView.zoomFactor Property

Gets or sets the zoom factor.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
get zoomFactor() {}
set zoomFactor(value) {}
```

Property Value

Number

The current zoom factor as percentage.

See Also

DiagramView Members
DiagramView Class

DiagramView Events

DiagramView.createEditControl Event

Raised to let you create a custom inplace editor control.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
diagramView.createEditControl.addEventListener(onCreateEditControl);

function onCreateEditControl(sender, args)
{
}
```

Event Data

Event handlers receive an InplaceEditEventArgs instance as argument.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property

DiagramView.diagramChanged Event

Raised when the diagram property changes.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

```
JavaScript
diagramView.diagramChanged.addEventListener(onDiagramChanged);

function onDiagramChanged(sender, args)
{
}
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DiagramView.enterInplaceEditMode Event

Raised when entering inplace-edit mode.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
diagramView.enterInplaceEditMode.addEventListener(onEnterInplaceEditMode);

function onEnterInplaceEditMode(sender, args)
{
}
```

Event Data

Event handlers receive an InplaceEditEventArgs instance as argument.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property
leaveInplaceEditMode Event

DiagramView.leaveInplaceEditMode Event

Raised when leaving inplace-edit mode.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
diagramView.leaveInplaceEditMode.addEventListener(onLeaveInplaceEditMode);

function onLeaveInplaceEditMode(sender, args)
{
}
```

Event Data

Event handlers receive an InplaceEditEventArgs instance as argument.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
allowInplaceEdit Property
enterInplaceEditMode Event

DiagramView.mouseWheel Event

Raised when the user rotates the mouse wheel while the cursor is positioned over the diagram surface.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
diagramView.mouseWheel.addEventListener(onMouseWheel);

function onMouseWheel(sender, args)
{
}
```

Event Data

Event handlers receive a DiagramEventArgs instance as argument.

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace
mouseWheelAction Property

DiagramView.zoomChanged Event

Raised when the zoom factor property changes.

Namespace: MindFusion.Diagramming
File: DiagramView.js

Syntax

JavaScript

```
diagramView.zoomChanged.addEventListener(onZoomChanged);

function onZoomChanged(sender, args)
{
}
```

See Also

DiagramView Members
DiagramView Class
MindFusion.Diagramming Namespace

DirectionConstraint Enumeration

Defines values that specify in what direction to constrain node movement.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript
`// enum` `DirectionConstraint = {}`

Members

	Member name	Description
	Horizontal	Users are allowed to move the node horizontally.
	None	No restrictions.
	Vertical	Users are allowed to move the node vertically.

See Also

MindFusion.Diagramming Namespace

DoNothingBehavior Class

Implements the DoNothing behavior.

Syntax

JavaScript
`// class` `DoNothingBehavior.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.DoNothingBehavior

See Also

DoNothingBehavior Members
MindFusion.Diagramming Namespace

DoNothingBehavior Members

Implements the DoNothing behavior.

The following tables list the members exposed by the DoNothingBehavior type.

Public Constructors

	Name	Description
	DoNothingBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.
	setMouseCursor	BehaviorBase.setMouseCursor override.

See Also

DoNothingBehavior Class
MindFusion.Diagramming Namespace

DoNothingBehavior Constructor

Overload List

Name	Description
DoNothingBehavior	Initializes a new instance of the DoNothingBehavior class.
DoNothingBehavior	Initializes a new instance of the DoNothingBehavior class.

See Also

DoNothingBehavior Class
MindFusion.Diagramming Namespace

DoNothingBehavior Constructor

Initializes a new instance of the DoNothingBehavior class.

Namespace: MindFusion.Diagramming
File: DoNothingBehavior.js

Syntax

JavaScript
`function DoNothingBehavior (parent)`

Parameters

parent
The Diagram this behavior will be associated with.

See Also

DoNothingBehavior Constructor Overload List
DoNothingBehavior Class
MindFusion.Diagramming Namespace

DoNothingBehavior Constructor

Initializes a new instance of the DoNothingBehavior class.

Namespace: MindFusion.Diagramming
File: DoNothingBehavior.js

Syntax

JavaScript

```
function DoNothingBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with..

See Also

DoNothingBehavior Constructor Overload List
DoNothingBehavior Class
MindFusion.Diagramming Namespace

DoNothingBehavior Methods

DoNothingBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: DoNothingBehavior.js

Syntax

JavaScript

```
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

DoNothingBehavior Members
DoNothingBehavior Class
MindFusion.Diagramming Namespace

DoNothingBehavior.setMouseCursor Method

BehaviorBase.setMouseCursor override.

Namespace: MindFusion.Diagramming
File: DoNothingBehavior.js

Syntax

JavaScript

```
function setMouseCursor (point, startInteraction)
```

Parameters

point

startInteraction

See Also

DoNothingBehavior Members
DoNothingBehavior Class
MindFusion.Diagramming Namespace

DrawAnchorPointEventArgs Class

Contains the arguments passed to the drawAnchorPoint event handlers.

Syntax

JavaScript

```
// class
DrawAnchorPointEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.CancelEventArgs
  MindFusion.Diagramming.ItemEventArgs
    MindFusion.Diagramming.NodeEventArgs
      MindFusion.Diagramming.DrawAnchorPointEventArgs

See Also

DrawAnchorPointEventArgs Members
MindFusion.Diagramming Namespace

DrawAnchorPointEventArgs Members

Contains the arguments passed to the drawAnchorPoint event handlers.

The following tables list the members exposed by the DrawAnchorPointEventArgs type.

Public Constructors

	Name	Description
	DrawAnchorPointEventArgs	Initializes a new instance of the **DrawAnchorPointEventArgs** class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user. (Inherited from ItemEventArgs.)
	anchorIndex	Gets the zero-based index of the anchor point related to the event.
	anchorPattern	Gets the anchor pattern containing the anchor point being drawn.
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	container	Gets the container a node has been added to or removed from. (Inherited from NodeEventArgs.)
	context	Gets the canvas rendering context. (Inherited from ItemEventArgs.)
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	hyperlink	Gets the hyperlink that was clicked inside a node. (Inherited from NodeEventArgs.)
	item	Gets the diagram item for which the event is raised. (Inherited from ItemEventArgs.)
	label	Gets the node label for which the event is raised. (Inherited from NodeEventArgs.)
	location	Gets the location of the anchor point being drawn.
	mouseButton	Gets which mouse button has been pressed. (Inherited from ItemEventArgs.)
	mousePosition	Gets the current mouse pointer position. (Inherited from ItemEventArgs.)
	newText	Gets the text that has just been entered by the user. (Inherited from ItemEventArgs.)
	node	Gets the node for which the event is raised. (Inherited from NodeEventArgs.)
	oldText	Gets the text that had been displayed in the node before the user edited it. (Inherited from ItemEventArgs.)

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the creation or modification of the node. (Inherited from ItemEventArgs.)

See Also

DrawAnchorPointEventArgs Class
MindFusion.Diagramming Namespace

DrawAnchorPointEventArgs Constructor

Initializes a new instance of the DrawAnchorPointEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function DrawAnchorPointEventArgs (args)
```

Parameters

args

See Also

DrawAnchorPointEventArgs Class
MindFusion.Diagramming Namespace

DrawAnchorPointEventArgs Properties

DrawAnchorPointEventArgs.anchorIndex Property

Gets the zero-based index of the anchor point related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get anchorIndex() {}
set anchorIndex(value) {}
```

Property Value

Number

The anchor point index.

See Also

DrawAnchorPointEventArgs Members
DrawAnchorPointEventArgs Class
MindFusion.Diagramming Namespace

DrawAnchorPointEventArgs.anchorPattern Property

Gets the anchor pattern containing the anchor point being drawn.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get anchorPattern() {}
set anchorPattern(value) {}
```

Property Value

AnchorPattern

The anchor pattern.

See Also

DrawAnchorPointEventArgs Members
DrawAnchorPointEventArgs Class
MindFusion.Diagramming Namespace

DrawAnchorPointEventArgs.location Property

Gets the location of the anchor point being drawn.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get location() {}
set location(value) {}
```

Property Value

Point

A Point instance.

See Also

DrawAnchorPointEventArgs Members
DrawAnchorPointEventArgs Class
MindFusion.Diagramming Namespace

DrawLinksBehavior Class

This class is used when the diagram behavior is set to Behavior.DrawLinks.

Syntax

JavaScript
`// class` `DrawLinksBehavior.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.DrawLinksBehavior

See Also

DrawLinksBehavior Members
MindFusion.Diagramming Namespace

DrawLinksBehavior Members

This class is used when the diagram behavior is set to Behavior.DrawLinks.

The following tables list the members exposed by the DrawLinksBehavior type.

Public Constructors

	Name	Description
	DrawLinksBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

DrawLinksBehavior Class
MindFusion.Diagramming Namespace

DrawLinksBehavior Constructor

Overload List

Name	Description
DrawLinksBehavior	Initializes a new instance of the DrawLinksBehavior class.
DrawLinksBehavior	Initializes a new instance of the DrawLinksBehavior class.

See Also

DrawLinksBehavior Class
MindFusion.Diagramming Namespace

DrawLinksBehavior Constructor

Initializes a new instance of the DrawLinksBehavior class.

Namespace: MindFusion.Diagramming
File: DrawLinksBehavior.js

Syntax

JavaScript

```
function DrawLinksBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

DrawLinksBehavior Constructor Overload List
DrawLinksBehavior Class
MindFusion.Diagramming Namespace

DrawLinksBehavior Constructor

Initializes a new instance of the DrawLinksBehavior class.

Namespace: MindFusion.Diagramming
File: DrawLinksBehavior.js

Syntax

JavaScript

```
function DrawLinksBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

DrawLinksBehavior Constructor Overload List
DrawLinksBehavior Class
MindFusion.Diagramming Namespace

DrawLinksBehavior Methods

DrawLinksBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: DrawLinksBehavior.js

Syntax

```
JavaScript
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

DrawLinksBehavior Members
DrawLinksBehavior Class
MindFusion.Diagramming Namespace

DrawNodesBehavior Class

The base class for behavior types that allow drawing of nodes, but not links.

Syntax

```
JavaScript
// class
DrawNodesBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.DrawNodesBehavior

See Also

DrawNodesBehavior Members
MindFusion.Diagramming Namespace

DrawNodesBehavior Members

The base class for behavior types that allow drawing of nodes, but not links.

The following tables list the members exposed by the DrawNodesBehavior type.

Public Constructors

	Name	Description
	DrawNodesBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

DrawNodesBehavior Class
MindFusion.Diagramming Namespace

DrawNodesBehavior Constructor

Overload List

Name	Description
DrawNodesBehavior	Initializes a new instance of the DrawNodesBehavior class.
DrawNodesBehavior	Initializes a new instance of the DrawNodesBehavior class.

See Also

DrawNodesBehavior Class
MindFusion.Diagramming Namespace

DrawNodesBehavior Constructor

Initializes a new instance of the DrawNodesBehavior class.

Namespace: MindFusion.Diagramming
File: DrawNodesBehavior.js

Syntax

```
JavaScript
function DrawNodesBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

DrawNodesBehavior Constructor Overload List
DrawNodesBehavior Class
MindFusion.Diagramming Namespace

DrawNodesBehavior Constructor

Initializes a new instance of the DrawNodesBehavior class.

Namespace: MindFusion.Diagramming
File: DrawNodesBehavior.js

Syntax

```
JavaScript
function DrawNodesBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

DrawNodesBehavior Constructor Overload List
DrawNodesBehavior Class
MindFusion.Diagramming Namespace

DrawNodesBehavior Methods

DrawNodesBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: DrawNodesBehavior.js

Syntax

```
JavaScript
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

DrawNodesBehavior Members
DrawNodesBehavior Class
MindFusion.Diagramming Namespace

DrawShapesBehavior Class

This class is used when the diagram behavior is set to Behavior.DrawShapes.

Syntax

JavaScript

```
// class
DrawShapesBehavior.prototype = {}
```

Inheritance Hierarchy

See Also

DrawShapesBehavior Members
MindFusion.Diagramming Namespace

DrawShapesBehavior Members

This class is used when the diagram behavior is set to Behavior.DrawShapes.

The following tables list the members exposed by the DrawShapesBehavior type.

Public Constructors

	Name	Description
	DrawShapesBehavior	Initializes a new instance of the DrawShapesBehavior class.

Public Methods

	Name	Description
	createNode	DrawNodesBehavior.CreateNode override.

See Also

DrawShapesBehavior Class
MindFusion.Diagramming Namespace

DrawShapesBehavior Constructor

Initializes a new instance of the DrawShapesBehavior class.

Namespace: MindFusion.Diagramming
File: DrawShapesBehavior.js

Syntax

```
JavaScript
function DrawShapesBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

DrawShapesBehavior Class
MindFusion.Diagramming Namespace

DrawShapesBehavior Methods

DrawShapesBehavior.createNode Method

DrawNodesBehavior.CreateNode override.

Namespace: MindFusion.Diagramming
File: DrawShapesBehavior.js

Syntax

```
JavaScript
function createNode ()
```

Return Value

An instance of the DiagramNode class.

See Also

DrawShapesBehavior Members
DrawShapesBehavior Class
MindFusion.Diagramming Namespace

DummyNode Class

DummyNode represents virtual nodes set as targets of disconnected links.

Syntax

JavaScript
`// class` `DummyNode.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.DummyNode

See Also

DummyNode Members
MindFusion.Diagramming Namespace

DummyNode Members

DummyNode represents virtual nodes set as targets of disconnected links.

The following tables list the members exposed by the **DummyNode** type.

Public Constructors

	Name	Description
	DummyNode	Initializes a new instance of the DummyNode class.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)

	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)

	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)

	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)

	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramItem.clone override. (Inherited from DiagramNode.)
	containsPoint	**DiagramItem.containsPoint** override. (Inherited from DiagramNode.)
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramItem.fromJson override. (Inherited from DiagramNode.)
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets the bounding rectangle of this node. (Inherited from DiagramNode.)

	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding **rectangle** of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramItem.loadFromXml override. Loads the node's content from an XML element. (Inherited from DiagramNode.)
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	DiagramItem.restoreState override. (Inherited from DiagramNode.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramItem.saveState override. (Inherited from DiagramNode.)
	saveToXml	Saves the item content into an XML element. (Inherited from DiagramNode.)
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramItem.toJson override. (Inherited from DiagramNode.)
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)

	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

See Also

DummyNode Class
MindFusion.Diagramming Namespace

DummyNode Constructor

Initializes a new instance of the DummyNode class.

Namespace: MindFusion.Diagramming
File: DummyNode.js

Syntax

JavaScript

```
function DummyNode ([parent])
```

Parameters

parent
Optional.
Diagram. A Diagram instance whose default node attributes are copied to this node.

See Also

DummyNode Class
MindFusion.Diagramming Namespace

EffectPhase Enumeration

Specifies the rendering phase during which an effect application is requested.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
EffectPhase = {}
```

Members

	Member name	Description
	AfterFill	Indicates that the NodeEffect.apply method is invoked after the target node is filled.
	AfterOutline	Indicates that the NodeEffect.apply method is invoked after the target node is stroked.

	BeforeFill	Indicates that the NodeEffect.apply method is invoked before the target node is filled.

See Also

MindFusion.Diagramming Namespace

Events Class

Obsolete. Defines all events raised by the Diagram component.

Syntax

JavaScript

```
// class
Events.prototype = {}
```

Remarks

Obsolete

Events are now raised by event emitters exposed as properties of the Diagram class. Adding event listeners through this class is supported for compatibility with old code.

To register an event handler function, call the addEventListener method of **Diagram**. Event handlers receive the diagram as first argument, and an instance of EventArgs subclass as second argument.

Example

The following code handles nodeCreated event, raised when the user draws a new node.

JavaScript

```
diagram.addEventListener(Events.nodeCreated, onNodeCreated);

function onNodeCreated(sender, args)
{
    var node = args.node;
    node.handlesStyle = HandlesStyle.HatchHandles3;
}
```

Inheritance Hierarchy

MindFusion.Diagramming.Events

See Also

Events Members
MindFusion.Diagramming Namespace
addEventListener Method
removeEventListener Method

Events Members

Obsolete. Defines all events raised by the Diagram component.

The following tables list the members exposed by the Events type.

Public Properties

	Name	Description
	actionRedone	Raised when an action is redone.
	actionUndone	Raised when an action is undone.
	animatedLayoutCompleted	Raised when an animated layout completes.
	cellTextEdited	Raised when the user edits the text of a table cell.
	cellTextEditing	Raised when the user starts to edit the text of a table cell.
	clicked	Raised when the user clicks the diagram at a point where no items are located.
	containerChildAdded	Raised when the user adds a node to a ContainerNode.
	containerChildAdding	A validation event raised to let you prevent users from adding child nodes to a container.
	containerChildRemoved	Raised when the user removes a node from a **ContainerNode**.
	containerChildRemoving	A validation event raised to let you prevent users from removing child nodes from a container.
	containerFolded	Raised when the user folds a container by clicking the arrow-up button in the container's caption bar.
	containerPageChanged	Raised when the user changes current page of a PagedContainerNode.
	containerUnfolded	Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar.
	controlLoaded	Raised when the control is loaded. For internal use by MindFusion ASP.NET WebForms / MVC controls.
	controlUnloaded	Raised when the control is unloaded.
	controlUnloading	Raised before the control is unloaded.
	createEditControl	Raised to let you create a custom inplace editor control.
	dataLoaded	Raised when the diagram data is loaded.
	deserializeTag	Raised when the Tag or Id of a diagram item must be deserialized from XML format.
	diagramChanged	Raised when the diagram property of a diagram view changes.
	documentChanged	Raised when a page is added or removed from a DiagramDocument instance.
	drawAdjustmentHandles	Raised to draw custom adjustment handles.

	drawAnchorPoint	Raised when an anchor point must be custom-drawn.
	enterInplaceEditMode	Raised when entering inplace-edit mode.
	expandButtonClicked	Raised if ExpandButtonAction is set to RaiseEvents and the [+/-]button of an Expandable node is clicked.
	headerResized	Raised when the user finished resizing a header.
	headerStartResizing	Raised when the user starts resizing a header.
	hitTestAdjustmentHandles	Raised to hit-test custom adjustment handles.
	hyperlinkClicked	Raised when the user clicks a hyperlink inside a node.
	initializeLink	Raised when the user starts drawing a new link, just after the DiagramLink instance is created.
	initializeNode	Raised when the user starts drawing a new node, just after the DiagramNode instance is created.
	itemAdded	Raised when an item is added to the diagram.
	itemRemoved	Raised when an item is removed from the diagram.
	laneGridCellTextEdited	Raised when the user edits the text of a grid cell.
	leaveInplaceEditMode	Raised when leaving inplace-edit mode.
	linkClicked	Raised when the user clicks a link.
	linkCreated	Raised when the user draws a new link.
	linkCreating	A validation event raised while the user is drawing a new link.
	linkDeleted	Raised when a link is deleted, either programmatically or by the user.
	linkDeleting	Raised when the user tries to delete a link. This event lets you cancel the operation.
	linkDeselected	Raised when a link is deselected, either programmatically or by the user.
	linkDoubleClicked	Raised when the user double-clicks a link.
	linkHovered	Raised when the mouse pointer hovers over a link.
	linkModified	Raised when the user moves any control point of a link.
	linkModifying	A validation event raised while the user is modifying a link.
	linkPasted	Raised when a link has been pasted from the clipboard.
	linkPointerEnter	Raised when the mouse pointer enters the boundaries of a link.
	linkPointerLeave	Raised when the mouse pointer leaves the boundaries of a link.
	linkRouted	Raised when a link is routed by the auto-routing algorithm. Event handlers receive a LinkEventArgs instance that contains data about this event.

	linkSelected	Raised when a link is selected, either programmatically or by the user.
	linkSelecting	Raised when the user tries to select a link. This event lets you cancel the operation.
	linkSplit	Raised when the user splits a link segment interactively.
	linkTextEdited	Raised when the user edits the text of a link.
	linkTextEditing	Raised when the user starts to edit the text of a link.
	mouseWheel	Raised when the user rotates the mouse wheel while the cursor is positioned over the diagram surface.
	nodeClicked	Raised when the user clicks a node.
	nodeCreated	Raised when the user draws a new node.
	nodeCreating	A validation event raised while the user is drawing a new node.
	nodeDeleted	Raised when a node is deleted, either programmatically or by the user.
	nodeDeleting	Raised when the user tries to delete a node. This event lets you cancel the operation.
	nodeDeselected	Raised when a node is deselected, either programmatically or by the user.
	nodeDomCreated	Raised when the content of a ControlNode is added to the page DOM.
	nodeDoubleClicked	Raised when the user double-clicks a node.
	nodeHovered	Raised when the mouse pointer hovers over a node.
	nodeModified	Raised when the user moves or resizes a node.
	nodeModifying	A validation event raised while the user is moving or resizing a node.
	nodePasted	Raised when a node has been pasted from the clipboard.
	nodePointerEnter	Raised when the mouse pointer enters the boundaries of a node.
	nodePointerLeave	Raised when the mouse pointer leaves the boundaries of a node.
	nodeSelected	Raised when a node is selected, either programmatically or by the user.
	nodeSelecting	Raised when the user tries to select a node. This event lets you cancel the operation.
	nodeTextEdited	Raised when the user edits the text of a node.
	nodeTextEditing	Raised when the user starts to edit the text of a node.
	queryView	Raised to enumerate diagram views.
	repaint	Raised when the diagram has been repainted.

	selectionChanged	Raised when the diagram selection has changed.
	selectionModifying	A validation event raised while the user is moving or resizing multiple items.
	serializeTag	Raised when the tag or id of a diagram item must be serialized in XML format.
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from shapeLibraryLocation.
	sizeChanged	Raised when the size of the diagram canvas has changed.
	tableColumnResized	Raised when the user finishes resizing a column in a table.
	tableColumnResizing	Raised when the user resizes a column in a table interactively.
	tableRowResized	Raised when the user finishes resizing a row in a table.
	tableRowResizing	Raised when the user resizes a row in a table interactively.
	tableSectionCollapsed	Raised when a table section is collapsed.
	tableSectionExpanded	Raised when a table section is expanded.
	treeCollapsed	Raised when a user collapses a tree branch by clicking the [-] button of an expandable node.
	treeExpanded	Raised when a user expands a tree branch by clicking the [+] button of an **expandable** node.
	treeItemTextEdited	Raised when the user edits the text of a tree view item.
	treeItemTextEditing	Raised when the user starts to edit the text of a tree view item.
	validateAnchorPoint	A validation event that lets you approve or reject connecting a link to an anchor point.
	validateLinkCrossing	A validation event that lets you approve or reject displaying a decoration at the intersection point of links.

See Also

Events Class
MindFusion.Diagramming Namespace

Events Properties

Events.actionRedone Property

Raised when an action is redone.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get actionRedone() {}
set actionRedone(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.actionUndone Property

Raised when an action is undone.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get actionUndone() {}
set actionUndone(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.animatedLayoutCompleted Property

Raised when an animated layout completes.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get animatedLayoutCompleted() {}
set animatedLayoutCompleted(value) {}
```

Property Value

String

A string containing the event name.

Remarks

When layout is applied on a ContainerNode, event handlers receive a NodeEventArgs instance as argument.

Example

This example handles the **animatedLayoutCompleted** event to resize the diagram.

JavaScript

```
diagram.addEventListener(Events.animatedLayoutCompleted, (sender, args) => {
    sender.resizeToFitItems(10);
});
diagram.arrangeAnimated(...);
```

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.cellTextEdited Property

Raised when the user edits the text of a table cell.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get cellTextEdited() {}
set cellTextEdited(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.cellTextEditing Property

Raised when the user starts to edit the text of a table cell.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get cellTextEditing() {}
set cellTextEditing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.clicked Property

Raised when the user clicks the diagram at a point where no items are located.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get clicked() {}
set clicked(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a DiagramEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
linkClicked Event
nodeClicked Event

Events.containerChildAdded Property

Raised when the user adds a node to a ContainerNode.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get containerChildAdded() {}
set containerChildAdded(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.containerChildAdding Property

A validation event raised to let you prevent users from adding child nodes to a container.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get containerChildAdding() {}
set containerChildAdding(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.containerChildRemoved Property

Raised when the user removes a node from a ContainerNode.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get containerChildRemoved() {}
set containerChildRemoved(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.containerChildRemoving Property

A validation event raised to let you prevent users from removing child nodes from a container.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get containerChildRemoving() {}
set containerChildRemoving(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.containerFolded Property

Raised when the user folds a container by clicking the arrow-up button in the container's caption bar.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get containerFolded() {}
set containerFolded(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.containerPageChanged Property

Raised when the user changes current page of a PagedContainerNode.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get containerPageChanged() {}
set containerPageChanged(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.containerUnfolded Property

Raised when the user unfolds a container by clicking the arrow-down button in the container's caption bar.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get containerUnfolded() {}
set containerUnfolded(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.controlLoaded Property

Raised when the control is loaded. For internal use by MindFusion ASP.NET WebForms / MVC controls.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get controlLoaded() {}
set controlLoaded(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Unlike most events listed here, the event listener must be added to DiagramView instead of Diagram, and respectively the *sender* argument of handler function is set to the view.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
addEventListener Method

Events.controlUnloaded Property

Raised when the control is unloaded.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get controlUnloaded() {}
set controlUnloaded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.controlUnloading Property

Raised before the control is unloaded.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get controlUnloading() {}
set controlUnloading(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.createEditControl Property

Raised to let you create a custom inplace editor control.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get createEditControl() {}
set createEditControl(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an InplaceEditEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.dataLoaded Property

Raised when the diagram data is loaded.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get dataLoaded() {}
set dataLoaded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class

Events.deserializeTag Property

Raised when the Tag or Id of a diagram item must be deserialized from XML format.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get deserializeTag() {}
set deserializeTag(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a SerializeTagEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.diagramChanged Property

Raised when the diagram property of a diagram view changes.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get diagramChanged() {}
set diagramChanged(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.documentChanged Property

Raised when a page is added or removed from a DiagramDocument instance.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get documentChanged() {}
set documentChanged(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.drawAdjustmentHandles Property

Raised to draw custom adjustment handles.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get drawAdjustmentHandles() {}
set drawAdjustmentHandles(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive either NodeEventArgs or LinkEventArgs instance as argument, depending on the item type.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.drawAnchorPoint Property

Raised when an anchor point must be custom-drawn.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get drawAnchorPoint() {}
set drawAnchorPoint(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Set the markStyle property of AnchorPoint to Custom to get this event raised. Handlers receive a DrawAnchorPointEventArgs instance as argument.

Example

JavaScript

```
diagram.addEventListener(
    Events.drawAnchorPoint, onDrawAnchorPoint);

function onDrawAnchorPoint(sender, e)
{
    var p = e.location;
    var ctx = e.context;
    ctx.fillStyle = "red";
    ctx.fillRect(p.x - 1, p.y - 1, 2, 2);
}
```

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.enterInplaceEditMode Property

Raised when entering inplace-edit mode.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get enterInplaceEditMode() {}
set enterInplaceEditMode(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an InplaceEditEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.expandButtonClicked Property

Raised if ExpandButtonAction is set to RaiseEvents and the [+/-]button of an Expandable node is clicked.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get expandButtonClicked() {}
set expandButtonClicked(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.headerResized Property

Raised when the user finished resizing a header.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get headerResized() {}
set headerResized(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a HeaderResizeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.headerStartResizing Property

Raised when the user starts resizing a header.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get headerStartResizing() {}
set headerStartResizing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a HeaderResizeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.hitTestAdjustmentHandles Property

Raised to hit-test custom adjustment handles.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get hitTestAdjustmentHandles() {}
set hitTestAdjustmentHandles(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive either NodeEventArgs or LinkEventArgs instance as argument, depending on the item type.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.hyperlinkClicked Property

Raised when the user clicks a hyperlink inside a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get hyperlinkClicked() {}
set hyperlinkClicked(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.initializeLink Property

Raised when the user starts drawing a new link, just after the DiagramLink instance is created.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get initializeLink() {}
set initializeLink(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.initializeNode Property

Raised when the user starts drawing a new node, just after the DiagramNode instance is created.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get initializeNode() {}
set initializeNode(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.itemAdded Property

Raised when an item is added to the diagram.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get itemAdded() {}
set itemAdded(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an ItemEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.itemRemoved Property

Raised when an item is removed from the diagram.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get itemRemoved() {}
set itemRemoved(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an ItemEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.laneGridCellTextEdited Property

Raised when the user edits the text of a grid cell.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get laneGridCellTextEdited() {}
set laneGridCellTextEdited(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.leaveInplaceEditMode Property

Raised when leaving inplace-edit mode.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get leaveInplaceEditMode() {}
set leaveInplaceEditMode(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an InplaceEditEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkClicked Property

Raised when the user clicks a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkClicked() {}
set linkClicked(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
linkHovered Event
linkDoubleClicked Event

Events.linkCreated Property

Raised when the user draws a new link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkCreated() {}
set linkCreated(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkCreating Property

A validation event raised while the user is drawing a new link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get linkCreating() {}
set linkCreating(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

Example

This example shows how to handle the **linkCreating** event to prevent cycles in the graph.

JavaScript

```
// assuming namespaced scripts
var Events = MindFusion.Diagramming.Events;
var PathFinder = MindFusion.Diagramming.PathFinder;

diagram.addEventListener(
    Events.linkCreating, onLinkCreating);

function onLinkCreating(diagram, args)
{
    if (args.destination == null)
    {
        // not pointing to a node yet
        return;
    }

    var pathFinder = new PathFinder(diagram);
    var path = pathFinder.findShortestPath(
        args.destination, args.origin);

    if (path != null)
    {
        // adding this new link would create a cycle
        // [origin]--[dest]--[path internal nodes]--[origin]

        args.cancel = true;
    }
}
```

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

allowSelfLoops Property
allowUnconnectedLinks Property

Events.linkDeleted Property

Raised when a link is deleted, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkDeleted() {}
set linkDeleted(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkDeleting Property

Raised when the user tries to delete a link. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkDeleting() {}
set linkDeleting(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkDeselected Property

Raised when a link is deselected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkDeselected() {}
set linkDeselected(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkDoubleClicked Property

Raised when the user double-clicks a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkDoubleClicked() {}
set linkDoubleClicked(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkHovered Property

Raised when the mouse pointer hovers over a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get linkHovered() {}
set linkHovered(value) {}
```

Property Value

String

A string containing the event name.

Remarks

This event fires when the mouse pointer rests over a link for hoverDelay milliseconds. Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
linkClicked Event
linkPointerEnter Event
linkPointerLeave Event

Events.linkModified Property

Raised when the user moves any control point of a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get linkModified() {}
set linkModified(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkModifying Property

A validation event raised while the user is modifying a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkModifying() {}
set linkModifying(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkPasted Property

Raised when a link has been pasted from the clipboard.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkPasted() {}
set linkPasted(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkPointerEnter Property

Raised when the mouse pointer enters the boundaries of a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get linkPointerEnter() {}
set linkPointerEnter(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance that contains data about this event.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
linkHovered Event
linkPointerLeave Event

Events.linkPointerLeave Property

Raised when the mouse pointer leaves the boundaries of a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get linkPointerLeave() {}
set linkPointerLeave(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance that contains data about this event.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
linkHovered Event
linkPointerEnter Event

Events.linkRouted Property

Raised when a link is routed by the auto-routing algorithm. Event handlers receive a LinkEventArgs instance that contains data about this event.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get linkRouted() {}
set linkRouted(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkSelected Property

Raised when a link is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkSelected() {}
set linkSelected(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkSelecting Property

Raised when the user tries to select a link. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkSelecting() {}
set linkSelecting(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkSplit Property

Raised when the user splits a link segment interactively.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkSplit() {}
set linkSplit(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
allowSplitLinks Property

Events.linkTextEdited Property

Raised when the user edits the text of a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkTextEdited() {}
set linkTextEdited(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.linkTextEditing Property

Raised when the user starts to edit the text of a link.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get linkTextEditing() {}
set linkTextEditing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a LinkEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.mouseWheel Property

Raised when the user rotates the mouse wheel while the cursor is positioned over the diagram surface.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get mouseWheel() {}
set mouseWheel(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a DiagramEventArgs instance as argument. Unlike most events listed here, the event listener must be added to DiagramView instead of Diagram, and respectively the *sender* argument of handler function is set to the view.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeClicked Property

Raised when the user clicks a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodeClicked() {}
set nodeClicked(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
nodeDoubleClicked Event
nodeHovered Event

Events.nodeCreated Property

Raised when the user draws a new node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodeCreated() {}
set nodeCreated(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

Example

This example handles the **nodeCreated** event to set new node's anchorPattern.

```
JavaScript
diagram.addEventListener(Events.nodeCreated, (sender, args) => {
    if (sender.defaultShape.id == "Rectangle")
```

```
        args.node.anchorPattern = pattern1;
    else
        args.node.anchorPattern = pattern2;
});
```

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeCreating Property

A validation event raised while the user is drawing a new node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeCreating() {}
set nodeCreating(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeDeleted Property

Raised when a node is deleted, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeDeleted() {}
set nodeDeleted(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeDeleting Property

Raised when the user tries to delete a node. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeDeleting() {}
set nodeDeleting(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeDeselected Property

Raised when a node is deselected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeDeselected() {}
set nodeDeselected(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeDomCreated Property

Raised when the content of a ControlNode is added to the page DOM.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeDomCreated() {}
set nodeDomCreated(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeDoubleClicked Property

Raised when the user double-clicks a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeDoubleClicked() {}
set nodeDoubleClicked(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeHovered Property

Raised when the mouse pointer hovers over a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeHovered() {}
set nodeHovered(value) {}
```

Property Value

String

A string containing the event name.

Remarks

This event fires when the mouse pointer rests over a node for hoverDelay milliseconds. Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
nodeClicked Event
nodePointerEnter Event
nodePointerLeave Event

Events.nodeModified Property

Raised when the user moves or resizes a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodeModified() {}
set nodeModified(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeModifying Property

A validation event raised while the user is moving or resizing a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodeModifying() {}
set nodeModifying(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodePasted Property

Raised when a node has been pasted from the clipboard.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodePasted() {}
set nodePasted(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodePointerEnter Property

Raised when the mouse pointer enters the boundaries of a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodePointerEnter() {}
set nodePointerEnter(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
nodeHovered Event
nodePointerLeave Event

Events.nodePointerLeave Property

Raised when the mouse pointer leaves the boundaries of a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodePointerLeave() {}
set nodePointerLeave(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance that contains data about this event.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
nodeHovered Event
nodePointerEnter Event

Events.nodeSelected Property

Raised when a node is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeSelected() {}
set nodeSelected(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeSelecting Property

Raised when the user tries to select a node. This event lets you cancel the operation.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeSelecting() {}
set nodeSelecting(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeTextEdited Property

Raised when the user edits the text of a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get nodeTextEdited() {}
set nodeTextEdited(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.nodeTextEditing Property

Raised when the user starts to edit the text of a node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get nodeTextEditing() {}
set nodeTextEditing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.queryView Property

Raised to enumerate diagram views.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get queryView() {}
set queryView(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.repaint Property

Raised when the diagram has been repainted.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get repaint() {}
set repaint(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an EventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.selectionChanged Property

Raised when the diagram selection has changed.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get selectionChanged() {}
set selectionChanged(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a SelectionChangedEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.selectionModifying Property

A validation event raised while the user is moving or resizing multiple items.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get selectionModifying() {}
set selectionModifying(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a SelectionEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.serializeTag Property

Raised when the tag or id of a diagram item must be serialized in XML format.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get serializeTag() {}
set serializeTag(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a SerializeTagEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.shapeLibraryLoaded Property

Raised when a ShapeLibrary is loaded from shapeLibraryLocation.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get shapeLibraryLoaded() {}
set shapeLibraryLoaded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.sizeChanged Property

Raised when the size of the diagram canvas has changed.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get sizeChanged() {}
set sizeChanged(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive an EventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.tableColumnResized Property

Raised when the user finishes resizing a column in a table.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get tableColumnResized() {}
set tableColumnResized(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
allowResizeColumns Property

Events.tableColumnResizing Property

Raised when the user resizes a column in a table interactively.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get tableColumnResizing() {}
set tableColumnResizing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
allowResizeColumns Property

Events.tableRowResized Property

Raised when the user finishes resizing a row in a table.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get tableRowResized() {}
set tableRowResized(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
allowResizeRows Property

Events.tableRowResizing Property

Raised when the user resizes a row in a table interactively.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get tableRowResizing() {}
set tableRowResizing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a CellEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace
allowResizeRows Property

Events.tableSectionCollapsed Property

Raised when a table section is collapsed.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get tableSectionCollapsed() {}
set tableSectionCollapsed(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.tableSectionExpanded Property

Raised when a table section is expanded.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

```
JavaScript
get tableSectionExpanded() {}
set tableSectionExpanded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.treeCollapsed Property

Raised when a user collapses a tree branch by clicking the [-] button of an expandable node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get treeCollapsed() {}
set treeCollapsed(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.treeExpanded Property

Raised when a user expands a tree branch by clicking the [+] button of an expandable node.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get treeExpanded() {}
set treeExpanded(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a NodeEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.treeItemTextEdited Property

Raised when the user edits the text of a tree view item.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get treeItemTextEdited() {}
set treeItemTextEdited(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a TreeItemEventArgs instance as argument.

Example

This example handles the **treeItemTextEdited** event to remove items without text.

JavaScript

```
diagram.addEventListener(Events.treeItemTextEdited, (sender, args) => {
    if (!args.newText)
        args.node.removeItem(args.item);
});
```

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.treeItemTextEditing Property

Raised when the user starts to edit the text of a tree view item.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get treeItemTextEditing() {}
set treeItemTextEditing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

Event handlers receive a TreeItemEventArgs instance as argument.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.validateAnchorPoint Property

A validation event that lets you approve or reject connecting a link to an anchor point.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get validateAnchorPoint() {}
set validateAnchorPoint(value) {}
```

Property Value

String

A string containing the event name.

Remarks

validateAnchorPoint event handlers receive an argument of type LinkValidationEventArgs.

This event is raised for each point of an AnchorPattern while a link's end point is being moved over a node. If none of the anchor points are validated, the mouse cursor icon changes to *NotAllowed*, indicating that the link cannot be connected to that node.

When the mouse button is released over a node, the event is raised again for the node's anchor points. From all validated anchor points, the one nearest to the dragged link end point is used to dock the link.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

Events.validateLinkCrossing Property

A validation event that lets you approve or reject displaying a decoration at the intersection point of links.

Namespace: MindFusion.Diagramming
File: Events.js

Syntax

JavaScript

```
get validateLinkCrossing() {}
set validateLinkCrossing(value) {}
```

Property Value

String

A string containing the event name.

Remarks

validateLinkCrossing event handlers receive an argument of type LinkCrossingEventArgs.

This event is raised for each intersection point of links when linkCrossings property is enabled.

See Also

Events Members
Events Class
MindFusion.Diagramming Namespace

ExpandButtonAction Enumeration

Specifies what action is triggered by the +/- buttons that are displayed near expandable nodes.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
ExpandButtonAction = {}
```

Members

	Member name	Description
	ExpandTreeBranch	Expands or collapses the children of a node in a tree-like diagram and raises the treeExpanded or treeCollapsed event.
	RaiseEvents	Raises the expandButtonClicked event.

Remarks

Members of this enumeration can be assigned to the expandButtonAction property of Diagram.

See Also

MindFusion.Diagramming Namespace

Factory Class

Provides shortcut methods for creating and adding instances of the standard item types.

Syntax

JavaScript
`// class` `Factory.prototype = {}`

Remarks

Obtain an instance of this class through the factory property of the Diagram class.

Inheritance Hierarchy

MindFusion.Diagramming.Factory

See Also

Factory Members
MindFusion.Diagramming Namespace

Factory Members

Provides shortcut methods for creating and adding instances of the standard item types.

The following tables list the members exposed by the Factory type.

Public Constructors

	Name	Description
	Factory	Initializes a new instance of the Factory class over the specified Diagram.

Public Methods

	Name	Description
	createContainerNode	Creates a new ContainerNode instance and adds it to the nodes collection of the underlying diagram.
	createControlNode	Creates a new ControlNode instance and adds it to the nodes collection of the underlying diagram.
	createDiagramLink	Overloaded.

	createPagedContainerNode	Creates a new PagedContainerNode instance and adds it to the nodes collection of the underlying diagram.
	createShapeNode	Creates a new ShapeNode instance and adds it to the nodes collection of the underlying diagram.
	createSvgNode	Creates a new SvgNode instance and adds it to the nodes collection of the underlying diagram.
	createTableNode	Creates a new TableNode instance and adds it to the nodes collection of the underlying diagram.
	createTreeViewNode	Creates a new TreeViewNode instance and adds it to the nodes collection of the underlying diagram.
	createVideoNode	Creates a new VideoNode instance and adds it to the nodes collection of the underlying diagram.

See Also

Factory Class
MindFusion.Diagramming Namespace

Factory Constructor

Initializes a new instance of the Factory class over the specified Diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function Factory (diagram)
```

Parameters

diagram
Diagram. The Diagram object for which to create the new factory.

See Also

Factory Class
MindFusion.Diagramming Namespace

Factory Methods

Factory.createContainerNode Method

Creates a new ContainerNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createContainerNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional. A Rect instance specifying the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

The newly created ContainerNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createControlNode Method

Creates a new ControlNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createControlNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional.
Rect. Specifies the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

ControlNode. The newly created ControlNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createDiagramLink Method

Overload List

Name	Description
Factory.createDiagramLink	Creates a new DiagramLink instance between the specified nodes and adds it to the links collection of the underlying diagram.
Factory.createDiagramLink	Creates a new **DiagramLink** instance between the specified nodes and adds it to the links collection of the underlying diagram.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createDiagramLink Method

Creates a new DiagramLink instance between the specified nodes and adds it to the links collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript
`function createDiagramLink ([origin, [destination]])`

Parameters

origin
Optional. The origin node.

destination
Optional. The destination **node**.

Return Value

The newly created DiagramLink instance.

Remarks

The method creates a link connecting the nodes specified by origin and destination.

See Also

createDiagramLink Method Overload List
Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createDiagramLink Method

Creates a new DiagramLink instance between the specified nodes and adds it to the links collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createDiagramLink (
    origin, [originRow], destination, [destRow])
```

Parameters

origin
Optional.
DiagramNode. The origin node.

originRow
Optional.
Number | TreeViewItem. The zero-based index of the origin table row or the origin tree item of the new link.

destination
Optional.
DiagramNode. The destination node.

destRow
Optional.
Number | **TreeviewItem**. The zero-based index of the destination table row or the destination tree item of the new link.

Return Value

The newly created DiagramLink instance.

Remarks

If the *origin* or *destination* argument refers to a TableNode, it can be optionally followed by a row index specifying the origin or destination row. If the *origin* or *destination* argument refers to a TreeViewNode, it can be optionally followed by a TreeViewItem argument specifying the origin or destination item. If index / item arguments are omitted, the link connects to the specified node as a target instead of specific element inside.

Example

The following script demonstrates various Factory method calls.

JavaScript

```
function buildDiagram(diagram)
{
    var node = diagram.factory.createShapeNode(10, 50, 40, 30);

    var table = diagram.factory.createTableNode(80, 10, 40, 60);
    table.text = "table";
    table.rowCount = 10;

    var treeView = diagram.factory.createTreeViewNode(150, 90, 40, 60);
    treeView.addItem(new TreeViewItem());
    treeView.addItem(new TreeViewItem());

    diagram.factory.createDiagramLink(node, table, 2);
    diagram.factory.createDiagramLink(table, 4, node);

    diagram.factory.createDiagramLink(treeView, treeView.rootItems[1], node);

    diagram.factory.createDiagramLink(table, 6, treeView, treeView.rootItems[0]);
}
```

See Also

createDiagramLink Method Overload List
Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createPagedContainerNode Method

Creates a new PagedContainerNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createPagedContainerNode ([bounds, [x, [y, [width, [height, [pageTitles]]]]]])
```

Parameters

bounds
Optional.
Rect. Specifies the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

pageTitles
Optional.
Array. An array of title strings for the initial pages.

Return Value

PagedContainerNode. The newly created PagedContainerNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createShapeNode Method

Creates a new ShapeNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createShapeNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional. A Rect instance specifying the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

The newly created ShapeNode instance.

Remarks

This method accepts either a single Rect object specifying the bounding rectangle of the new node, or a list of four numbers specifying the location and size.

The method creates a shape node with the specified arguments, if any.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createSvgNode Method

Creates a new SvgNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createSvgNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional. A Rect instance specifying the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

The newly created SvgNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createTableNode Method

Creates a new TableNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createTableNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional. A Rect instance specifying the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

The newly created TableNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createTreeViewNode Method

Creates a new TreeViewNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createTreeViewNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional.

Rect. Specifies the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

TreeViewNode. The newly created TreeViewNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

Factory.createVideoNode Method

Creates a new VideoNode instance and adds it to the nodes collection of the underlying diagram.

Namespace: MindFusion.Diagramming
File: Factory.js

Syntax

JavaScript

```
function createVideoNode ([bounds, [x, [y, [width, [height]]]]])
```

Parameters

bounds
Optional.
A Rect instance specifying the bounding rectangle of the new node.

x
Optional.
Number. The x-coordinate of the upper-left corner of the new node.

y
Optional.
Number. The y-coordinate of the upper-left corner of the new node.

width
Optional.
Number. The width of the new node.

height
Optional.
Number. The height of the new node.

Return Value

The newly created VideoNode instance.

See Also

Factory Members
Factory Class
MindFusion.Diagramming Namespace

FitSize Enumeration

Specifies how the resizeToFitText method should resize a node.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
// enum
FitSize = {}
```

Members

	Member name	Description
	KeepHeight	The original node height should remain intact.
	KeepRatio	Allow changing both the width and height of a node, but keep the original width/height ratio if possible.
	KeepWidth	The original node width should remain intact.

See Also

MindFusion.Diagramming Namespace

FoldContainerCommand Class

Handles undo / redo of container folding.

Syntax

```
// class
FoldContainerCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
MindFusion.Diagramming.FoldContainerCommand

See Also

FoldContainerCommand Members
MindFusion.Diagramming Namespace

FoldContainerCommand Members

Handles undo / redo of container folding.

The following tables list the members exposed by the FoldContainerCommand type.

Public Constructors

	Name	Description
	FoldContainerCommand	Initializes a new instance of the **FoldContainerCommand** class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

FoldContainerCommand Class
MindFusion.Diagramming Namespace

FoldContainerCommand Constructor

Initializes a new instance of the FoldContainerCommand class.

Namespace: MindFusion.Diagramming
File: FoldContainerCommand.js

Syntax

JavaScript

```
function FoldContainerCommand (diagram, container)
```

Parameters

diagram
The Diagram modified by this command.

container
The ContainerNode instance that will be modified.

See Also

FoldContainerCommand Class
MindFusion.Diagramming Namespace

FoldContainerCommand Methods

FoldContainerCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: FoldContainerCommand.js

Syntax

```
JavaScript
function execute ()
```

See Also

FoldContainerCommand Members
FoldContainerCommand Class
MindFusion.Diagramming Namespace

FoldContainerCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: FoldContainerCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

FoldContainerCommand Members
FoldContainerCommand Class
MindFusion.Diagramming Namespace

FoldContainerCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: FoldContainerCommand.js

Syntax

```
JavaScript
```

```
function undo ()
```

See Also

FoldContainerCommand Members
FoldContainerCommand Class
MindFusion.Diagramming Namespace

FreeFormNode Class

A **FreeFormNode** collects all points from users' mouse or touch input and displays them as node's outline.

Syntax

JavaScript

```
// class
FreeFormNode.prototype = {}
```

Remarks

To let users draw free-form nodes interactively, set behavior to DrawFreeForms or **LinkFreeForms**. Use the points property of **FreeFormNode** to get or set outline points programmatically. If the closed property is set, the node is drawn as a closed shape and its interior filled, or otherwise the node is drawn as a polyline. If the distance between first and last points drawn by user is shorter than autoCloseDistance, the node's **closed** property is automatically set to **true**. **autoCloseDistance** default value is Number.MAX_VALUE, so free-form nodes are always closed.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.FreeFormNode

See Also

FreeFormNode Members
MindFusion.Diagramming Namespace

FreeFormNode Members

A FreeFormNode collects all points from users' mouse or touch input and displays them as node's outline.

The following tables list the members exposed by the **FreeFormNode** type.

Public Constructors

	Name	Description
	FreeFormNode	Initializes a new instance of the **FreeFormNode** class.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	closed	Gets or sets whether the node's shape is closed.
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)

	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)

	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	points	Gets or sets the node's outline points.
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)

	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)

	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	DiagramItem.containsPoint override.
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	DiagramNode.endDrag override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	fromJson	DiagramNode.fromJson override.
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	DiagramNode.getOutline override. Gets the outline of this node.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	init	Returns a Builder object used to configure this **FreeFormNode** instance.
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramNode.LoadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)

	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	DiagramNode.restoreLocationState override.
	restoreState	DiagramNode.restoreState override.
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	DiagramNode.saveLocationState override.
	saveState	DiagramNode.saveState override.
	saveToXml	Saves the node's content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	DiagramNode.startDrag override.
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	DiagramNode.updateDrag override.
	updateFromPoints	Updates the node's internal state after its points have changed.
S	With	Returns a Builder object used to configure and create new **FreeFormNode** instances.

See Also

FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode Constructor

Initializes a new instance of the FreeFormNode class.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function FreeFormNode ([parent])
```

Parameters

parent
Optional.
A Diagram instance whose default node attributes are copied to this node.

See Also

FreeFormNode Methods

FreeFormNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function clone ()
```

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.containsPoint Method

DiagramItem.containsPoint override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function containsPoint (point)
```

Parameters

point
Point. The point to check.

Return Value

true it this node contains the specified point; otherwise, **false**.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.endDrag Method

DiagramNode.endDrag override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

```
JavaScript
function endDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.getOutline Method

DiagramNode.getOutline override. Gets the outline of this node.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

```
JavaScript
function getOutline ()
```

Return Value

Array. An array of points.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.init Method

Returns a Builder object used to configure this FreeFormNode instance.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function init ()
```

Return Value

A Builder instance.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.loadFromXml Method

DiagramNode.LoadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.restoreLocationState Method

DiagramNode.restoreLocationState override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function restoreLocationState (ist)
```

Parameters

ist
DiagramNodeState. A DiagramNodeState instance.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.restoreState Method

DiagramNode.restoreState override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.saveLocationState Method

DiagramNode.saveLocationState override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function saveLocationState ()
```

Return Value

DiagramNodeState. A DiagramNodeState instance.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.saveState Method

DiagramNode.saveState override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function saveState ()
```

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.startDrag Method

DiagramNode.startDrag override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function startDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function toJson ()
```

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.updateDrag Method

DiagramNode.updateDrag override.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function updateDrag (ist)
```

Parameters

ist

An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.updateFromPoints Method

Updates the node's internal state after its points have changed.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function updateFromPoints ()
```

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.With Method

Returns a Builder object used to configure and create new FreeFormNode instances.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

JavaScript

```
function With (diagram)
```

Parameters

diagram

Return Value

A Builder instance.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode Properties

FreeFormNode.closed Property

Gets or sets whether the node's shape is closed.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

```
JavaScript
get closed() {}
set closed(value) {}
```

Property Value

Boolean

true it this node has a closed shape, or **false** otherwise.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

FreeFormNode.points Property

Gets or sets the node's outline points.

Namespace: MindFusion.Diagramming
File: FreeFormNode.js

Syntax

```
JavaScript
get points() {}
set points(value) {}
```

Property Value

Array

An array of Point objects.

See Also

FreeFormNode Members
FreeFormNode Class
MindFusion.Diagramming Namespace

GlassEffect Class

Represents a glass-like visual effect.

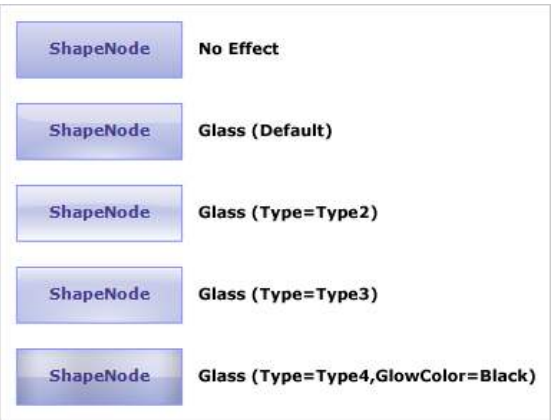
Syntax

```
JavaScript
// class
GlassEffect.prototype = {}
```

Remarks

This effect can be applied to all nodes in a Diagram by adding **GlassEffect** instance to the nodeEffects collection, or to individual nodes by adding it to their effects array.

The following image illustrates the Glass effect applied to an item as well as how the different properties affect the output:



Example

The following example demonstrates how to apply glass effect to the nodes of an existing diagram.

```
JavaScript
var GlassEffect = MindFusion.Diagramming.GlassEffect;
diagram.nodeEffects.push(new GlassEffect());
```

Inheritance Hierarchy

MindFusion.Diagramming.NodeEffect
  MindFusion.Diagramming.GlassEffect

See Also

GlassEffect Members
MindFusion.Diagramming Namespace
Node Effects
AeroEffect Class

GlassEffect Members

Represents a glass-like visual effect.

The following tables list the members exposed by the GlassEffect type.

Public Constructors

	Name	Description

	GlassEffect	Initializes a new instance of the GlassEffect class.

Public Properties

	Name	Description
	glowColor	Gets or sets the color of the glow region of the effect.
	reflectionColor	Gets or sets the color of the reflection region of the effect.
	type	Gets or sets the type of the glass effect.
	usePenAsGlow	Gets or sets whether the glow area of the effect should be painted using the stroke color of target node.

See Also

GlassEffect Class
MindFusion.Diagramming Namespace

GlassEffect Constructor

Initializes a new instance of the GlassEffect class.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
function GlassEffect ()
```

See Also

GlassEffect Class
MindFusion.Diagramming Namespace

GlassEffect Properties

GlassEffect.glowColor Property

Gets or sets the color of the glow region of the effect.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
get glowColor() {}
set glowColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value. The default value is "white".

See Also

GlassEffect Members
GlassEffect Class
MindFusion.Diagramming Namespace

GlassEffect.reflectionColor Property

Gets or sets the color of the reflection region of the effect.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
get reflectionColor() {}
set reflectionColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value. The default value is "white".

See Also

GlassEffect Members
GlassEffect Class
MindFusion.Diagramming Namespace

GlassEffect.type Property

Gets or sets the type of the glass effect.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
get type() {}
set type(value) {}
```

Property Value

GlassEffectType

A member of the GlassEffectType enumeration. The default is **Type1**.

See Also

GlassEffect Members
GlassEffect Class
MindFusion.Diagramming Namespace

GlassEffect.usePenAsGlow Property

Gets or sets whether the glow area of the effect should be painted using the stroke color of target node.

Namespace: MindFusion.Diagramming
File: NodeEffects.js

Syntax

JavaScript

```
get usePenAsGlow() {}
set usePenAsGlow(value) {}
```

Property Value

Boolean

true to ignore glowColor and use node's stroke color for the glow, or **false** otherwise.

See Also

GlassEffect Members
GlassEffect Class
MindFusion.Diagramming Namespace

GlassEffectType Enumeration

Specifies the type of a GlassEffect node effect.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
GlassEffectType = {}
```

Members

	Member name	Description
	Type1	Indicates a glass effect with soft radial glow at the bottom and a sharp reflection at the top.
	Type2	Indicates a glass effect with linear glow at the bottom and sharp reflection at the top.
	Type3	Indicates a glass effect with radial glow at the bottom and smooth reflection and emphasized outline at the top.

	Type4	Indicates a glass effect with sharp reflection at the top and soft glow at the left and right sides.

Remarks

Members of this enumeration can be assigned to the type property of GlassEffect.

See Also

MindFusion.Diagramming Namespace

GridColumn Class

Represents a column in a GridPanel.

Syntax

JavaScript

```
// class
GridColumn.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.GridColumn

See Also

GridColumn Members
MindFusion.Diagramming Namespace

GridColumn Members

Represents a column in a GridPanel.

The following tables list the members exposed by the GridColumn type.

Public Constructors

	Name	Description
	GridColumn	Initializes a new instance of the **GridColumn** class.

Public Properties

	Name	Description
	lengthType	Gets or sets how **GridPanel** determines the width of this column.
	width	Gets or sets fixed width for this column.

See Also

GridColumn Class
MindFusion.Diagramming Namespace

GridColumn Constructor

Initializes a new instance of the GridColumn class.

Namespace: MindFusion.Diagramming
File: GridColumn.js

Syntax

JavaScript

```
function GridColumn ()
```

See Also

GridColumn Class
MindFusion.Diagramming Namespace

GridColumn Properties

GridColumn.lengthType Property

Gets or sets how GridPanel determines the width of this column.

Namespace: MindFusion.Diagramming
File: GridColumn.js

Syntax

JavaScript

```
get lengthType() {}
set lengthType(value) {}
```

Property Value

LengthType

A member of the LengthType enumeration.

See Also

GridColumn Members
GridColumn Class
MindFusion.Diagramming Namespace

GridColumn.width Property

Gets or sets fixed width for this column.

Namespace: MindFusion.Diagramming
File: GridColumn.js

Syntax

```
JavaScript
get width() {}
set width(value) {}
```

Property Value

Number

A number specifying the column's width.

See Also

GridColumn Members
GridColumn Class
MindFusion.Diagramming Namespace

GridPanel Class

A layout container that arranges its child components in a grid.

Syntax

```
JavaScript
// class
GridPanel.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Container
  MindFusion.Diagramming.Panel
    MindFusion.Diagramming.StackPanel

See Also

GridPanel Members
MindFusion.Diagramming Namespace

GridPanel Members

A layout container that arranges its child components in a grid.

The following tables list the members exposed by the GridPanel type.

Public Constructors

	Name	Description
	GridPanel	Initializes a new instance of the GridPanel class.

Public Properties

	Name	Description
	columns	Gets a list of GridColumn objects specifying column attributes.
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	rows	Gets a list of GridRow objects specifying row attributes.
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	getComponents	Gets the child components of this panel. (Inherited from Panel.)

See Also

GridPanel Class
MindFusion.Diagramming Namespace

GridPanel Constructor

Initializes a new instance of the GridPanel class.

Namespace: MindFusion.Diagramming
File: GridPanel.js

Syntax

JavaScript
`function GridPanel ()`

See Also

GridPanel Class
MindFusion.Diagramming Namespace

GridPanel Properties

GridPanel.columns Property

Gets a list of GridColumn objects specifying column attributes.

Namespace: MindFusion.Diagramming
File: GridPanel.js

Syntax

```
JavaScript
get columns() {}
```

Property Value

Array

An array containing the grid columns.

See Also

GridPanel Members
GridPanel Class
MindFusion.Diagramming Namespace

GridPanel.rows Property

Gets a list of GridRow objects specifying row attributes.

Namespace: MindFusion.Diagramming
File: GridPanel.js

Syntax

```
JavaScript
get rows() {}
```

Property Value

Array

An array containing the grid rows.

See Also

GridPanel Members
GridPanel Class
MindFusion.Diagramming Namespace

GridRouter Class

Implements a link routing algorithm that uses a grid to represent the free and occupied regions in the diagram.

Syntax

JavaScript

```
// class
GridRouter.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Router
  MindFusion.Diagramming.GridRouter

See Also

GridRouter Members
MindFusion.Diagramming Namespace

GridRouter Members

Implements a link routing algorithm that uses a grid to represent the free and occupied regions in the diagram.

The following tables list the members exposed by the GridRouter type.

Public Constructors

	Name	Description
	GridRouter	Initializes a new instance of the GridRouter class.

Public Properties

	Name	Description
	avoidOverlaps	Gets or sets a value indicating whether to avoid overlapping links. (Inherited from Router.)
	crossingCost	Gets or sets a value added to the total cost of a route which causes a link to cross another link.
	gridSize	Gets or sets the size of the pieces of the routing grid.
	lengthCost	Gets or sets a value added to the total cost of a route for each piece of the routing grid occupied by the route.
	nodeVicinityCost	Gets or sets a value added to the total cost of a route if the route passes too close to a node.
	nodeVicinitySize	Gets or sets the size of buffer zone around nodes for which routes get a penalty cost if they pass through it.
	turnCost	Gets or sets a value added to the total cost of a route if the route passes too close to a node.

Public Methods

	Name	Description
	route	Called when a single link must be routed. (Inherited from Router.)

	routeInternal	Calculates actual route and returns it as an array of points. (Inherited from Router.)
	routeLinks	Called when more than one link must be routed. (Inherited from Router.)

See Also

GridRouter Class
MindFusion.Diagramming Namespace

GridRouter Constructor

Initializes a new instance of the GridRouter class.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

JavaScript

```
function GridRouter ([diagram])
```

Parameters

diagram
Optional.
Diagram. Gets or sets the diagram this router is associated with.

See Also

GridRouter Class
MindFusion.Diagramming Namespace

GridRouter Properties

GridRouter.crossingCost Property

Gets or sets a value added to the total cost of a route which causes a link to cross another link.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

JavaScript

```
get crossingCost() {}
set crossingCost(value) {}
```

Property Value

Number

A number specifying the crossing cost. The default is 4.

Remarks

This is a penalty value added to the total cost of an evaluated link route if it crosses another link. The routing algorithm aims to find paths that have a minimal cost, so assigning a greater value to **crossingCost** leads to fewer link intersections. This value is considered only when routing multiple links at once, e.g. by routeAllLinks method.

See Also

GridRouter Members
GridRouter Class
MindFusion.Diagramming Namespace

GridRouter.gridSize Property

Gets or sets the size of the pieces of the routing grid.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

JavaScript

```
get gridSize() {}
set gridSize(value) {}
```

Property Value

Number

A number specifying the size of the pieces of the routing grid. The default is 4.

See Also

GridRouter Members
GridRouter Class
MindFusion.Diagramming Namespace

GridRouter.lengthCost Property

Gets or sets a value added to the total cost of a route for each piece of the routing grid occupied by the route.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

JavaScript

```
get lengthCost() {}
set lengthCost(value) {}
```

Property Value

Number

A number specifying the length cost. The default is 2.

See Also

GridRouter Members
GridRouter Class
MindFusion.Diagramming Namespace

GridRouter.nodeVicinityCost Property

Gets or sets a value added to the total cost of a route if the route passes too close to a node.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

JavaScript

```
get nodeVicinityCost() {}
set nodeVicinityCost(value) {}
```

Property Value

Number

A number specifying the node vicinity cost. The default is 2.

See Also

GridRouter Members
GridRouter Class
MindFusion.Diagramming Namespace

GridRouter.nodeVicinitySize Property

Gets or sets the size of buffer zone around nodes for which routes get a penalty cost if they pass through it.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

JavaScript

```
get nodeVicinitySize() {}
set nodeVicinitySize(value) {}
```

Property Value

Number

A number specifying the size of the buffer area in which passing routes receive a penalty cost. The default is 12.

See Also

GridRouter Members
GridRouter Class
MindFusion.Diagramming Namespace

GridRouter.turnCost Property

Gets or sets a value added to the total cost of a route if the route passes too close to a node.

Namespace: MindFusion.Diagramming
File: GridRouter.js

Syntax

```
JavaScript
get turnCost() {}
set turnCost(value) {}
```

Property Value

Number

A number specifying the node vicinity cost. The default is 2.

See Also

GridRouter Members
GridRouter Class
MindFusion.Diagramming Namespace

GridRow Class

Represents a row in a GridPanel.

Syntax

```
JavaScript
// class
GridRow.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.GridRow

See Also

GridRow Members
MindFusion.Diagramming Namespace

GridRow Members

Represents a row in a GridPanel.

The following tables list the members exposed by the GridRow type.

Public Constructors

	Name	Description
	GridRow	Initializes a new instance of the **GridRow** class.

Public Properties

	Name	Description
	height	Gets or sets fixed height for this row.
	lengthType	Gets or sets how GridPanel determines the height of this row.

See Also

GridRow Class
MindFusion.Diagramming Namespace

GridRow Constructor

Initializes a new instance of the GridRow class.

Namespace: MindFusion.Diagramming
File: GridRow.js

Syntax

JavaScript

```
function GridRow ()
```

See Also

GridRow Class
MindFusion.Diagramming Namespace

GridRow Properties

GridRow.height Property

Gets or sets fixed height for this row.

Namespace: MindFusion.Diagramming
File: GridRow.js

Syntax

JavaScript

```
get height() {}
set height(value) {}
```

Property Value

Number

A number specifying the rows's height.

See Also

GridRow Members
GridRow Class
MindFusion.Diagramming Namespace

GridRow.lengthType Property

Gets or sets how GridPanel determines the height of this row.

Namespace: MindFusion.Diagramming
File: GridRow.js

Syntax

JavaScript

```
get lengthType() {}
set lengthType(value) {}
```

Property Value

LengthType

A member of the LengthType enumeration.

See Also

GridRow Members
GridRow Class
MindFusion.Diagramming Namespace

GridStyle Enumeration

Defines values that specify the visual style of the alignment grid.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
GridStyle = {}
```

Members

	Member name	Description
	Crosses	The grid is painted as a matrix of crosses.

	Lines	The grid is represented as a series of crossing horizontal and vertical lines.
	Points	The grid is painted as a matrix of points.

Remarks

Members of this enumeration can be assigned to the gridStyle property of Diagram.

See Also

MindFusion.Diagramming Namespace

HandlesStyle Enumeration

Enumerates possible visual styles for frames and handles drawn around selected nodes.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
HandlesStyle = {}
```

Members

	Member name	Description
	Custom	The control raises the drawAdjustmentHandles and hitTestAdjustmentHandles events to let your application perform custom drawing and hit testing of selection handles.
	DashFrame	If selected, a node is drawn with a dashed frame. The frame enables resizing the node. The center of the node allows moving it and the edge area enables creating links.
	EasyMove	Any point of the interior of a node enables moving the node, except a small area in the center that allows creating links. Best used with ModificationStart set to AutoHandles.
	HatchFrame	If selected, the item is drawn with hatched frame. Modifications can be done as with DashFrame.
	HatchHandles	If selected, a node is drawn with both hatched frame and modification handles. The frame enable moving the node, and the handles enable resizing. Links are created from any point of the interior.
	HatchHandles2	Looks like HatchHandles, but allows moving the node from the interior area. Links are created from the points near the edge.
	HatchHandles3	Behaves like HatchHandles2, but paints the selection frame using a denser brush pattern.

	Invisible	There is no visible difference between selected and unselected node.
	InvisibleMove	There aren't any selection handles displayed. The selected node can be only moved, but not resized.
	MoveOnly	If a node is selected, it can be only moved. Best used with ModificationStart set to AutoHandles.
	RoundAndSquare	Corner resize handles are round, remaining handles are square.
	RoundAndSquare2	Corner resize handles are round, remaining handles are square. In addition, there is a dashed frame drawn around selected nodes.
	SquareHandles	If selected, the node is drawn with square modification handles.
	SquareHandles2	Square resize handles are drawn around a selected node. There isn't a move handle in the center as with the SquareHandles style, but nodes can be moved by dragging their interior or caption areas. Links can be created by drawing from near the edges of a node.

Remarks

Members of this enumeration can be assigned to the handlesStyle property of DiagramNode.

See Also

MindFusion.Diagramming Namespace

HandlesVisualStyle Class

Defines the pen and brush objects used to draw items' adjustment handles.

Syntax

JavaScript

```
// class
HandlesVisualStyle.prototype = {}
```

Remarks

The component uses several different **HandlesVisualStyle** objects depending on the context. They can be accessed through the activeItemHandlesStyle, selectedItemHandlesStyle and disabledHandlesStyle properties of Diagram.

Inheritance Hierarchy

MindFusion.Diagramming.HandlesVisualStyle

See Also

HandlesVisualStyle Members

Defines the pen and brush objects used to draw items' adjustment handles.

The following tables list the members exposed by the HandlesVisualStyle type.

Public Constructors

	Name	Description
	HandlesVisualStyle	Initializes a new instance of the HandlesVisualStyle class.

Public Properties

	Name	Description
	controlPointBrush	Gets or sets the brush used to fill the interior of shape control points.
	dashStroke	Gets or sets the color used to stroke dashed selection frames.
	dashStrokeThickness	Gets or sets the line width applied when stroking dashed selection frames.
	handleBrush	Gets or sets the brush used to fill the interior of selection handles.
	handleStroke	Gets or sets the color used to stroke borders of adjustment handles.
	handleStrokeThickness	Gets or sets the line width applied when stroking borders of adjustment handles.
	hatchBrush	Gets or sets the brush used to draw hatch frames.
	patternBrush	Gets or sets the brush used to draw selection frames for the MoveOnly style.
	rotationHandleBrush	Gets or sets the brush used to fill the interior of the rotation handle.

See Also

HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle Constructor

Initializes a new instance of the HandlesVisualStyle class.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

```
JavaScript
function HandlesVisualStyle ()
```

See Also

HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle Properties

HandlesVisualStyle.controlPointBrush Property

Gets or sets the brush used to fill the interior of shape control points.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

```
JavaScript
get controlPointBrush() {}
set controlPointBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.dashStroke Property

Gets or sets the color used to stroke dashed selection frames.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

```
JavaScript
get dashStroke() {}
set dashStroke(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.dashStrokeThickness Property

Gets or sets the line width applied when stroking dashed selection frames.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

JavaScript

```
get dashStrokeThickness() {}
set dashStrokeThickness(value) {}
```

Property Value

Number

A number specifying stroke thickness.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.handleBrush Property

Gets or sets the brush used to fill the interior of selection handles.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

JavaScript

```
get handleBrush() {}
set handleBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.handleStroke Property

Gets or sets the color used to stroke borders of adjustment handles.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

```
JavaScript
get handleStroke() {}
set handleStroke(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.handleStrokeThickness Property

Gets or sets the line width applied when stroking borders of adjustment handles.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

```
JavaScript
get handleStrokeThickness() {}
set handleStrokeThickness(value) {}
```

Property Value

Number

A number specifying stroke thickness.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.hatchBrush Property

Gets or sets the brush used to draw hatch frames.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

JavaScript

```
get hatchBrush() {}
set hatchBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.patternBrush Property

Gets or sets the brush used to draw selection frames for the MoveOnly style.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

JavaScript

```
get patternBrush() {}
set patternBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HandlesVisualStyle.rotationHandleBrush Property

Gets or sets the brush used to fill the interior of the rotation handle.

Namespace: MindFusion.Diagramming
File: HandlesVisualStyle.js

Syntax

JavaScript

```
get rotationHandleBrush() {}
set rotationHandleBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

HandlesVisualStyle Members
HandlesVisualStyle Class
MindFusion.Diagramming Namespace

HeaderEventArgs Class

Contains the arguments passed to handlers of header-related events.

Syntax

JavaScript

```
// class
HeaderEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.EventArgs
  MindFusion.Diagramming.HeaderEventArgs
    MindFusion.Diagramming.HeaderResizeEventArgs

See Also

HeaderEventArgs Members
MindFusion.Diagramming Namespace
Events Class

HeaderEventArgs Members

Contains the arguments passed to handlers of header-related events.

The following tables list the members exposed by the HeaderEventArgs type.

Public Constructors

	Name	Description
	HeaderEventArgs	Initializes a new instance of the HeaderEventArgs class.

Public Properties

	Name	Description
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	header	Gets the header related to the event.

See Also

HeaderEventArgs Class
MindFusion.Diagramming Namespace

HeaderEventArgs Constructor

Initializes a new instance of the HeaderEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function HeaderEventArgs ()
```

See Also

HeaderEventArgs Class
MindFusion.Diagramming Namespace

HeaderEventArgs Properties

HeaderEventArgs.header Property

Gets the header related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get header() {}
```

Property Value

Header

The grid header.

See Also

HeaderEventArgs Members
HeaderEventArgs Class
MindFusion.Diagramming Namespace

HeaderResizeEventArgs Class

Contains the arguments passed to handlers of header resize events.

Syntax

JavaScript

```
// class
HeaderResizeEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.EventArgs
  MindFusion.Diagramming.HeaderEventArgs
    MindFusion.Diagramming.HeaderResizeEventArgs

See Also

HeaderResizeEventArgs Members
MindFusion.Diagramming Namespace
Events Class

HeaderResizeEventArgs Members

Contains the arguments passed to handlers of header resize events.

The following tables list the members exposed by the HeaderResizeEventArgs type.

Public Constructors

	Name	Description
	HeaderResizeEventArgs	Initializes a new instance of the HeaderEventArgs class.

Public Properties

	Name	Description
	cancel	Gets or sets a value indicating whether to allow the current operation.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	header	Gets the header related to the event. (Inherited from HeaderEventArgs.)
	horizontal	Gets a value indicating whether the user resizes the header horizontally.

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the resize operation.

See Also

HeaderResizeEventArgs Class
MindFusion.Diagramming Namespace

HeaderResizeEventArgs Constructor

Initializes a new instance of the HeaderEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function HeaderResizeEventArgs ()
```

See Also

HeaderResizeEventArgs Class
MindFusion.Diagramming Namespace

HeaderResizeEventArgs Methods

HeaderResizeEventArgs.cancelDrag Method

Immediately cancels the resize operation.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function cancelDrag ()
```

See Also

HeaderResizeEventArgs Members
HeaderResizeEventArgs Class
MindFusion.Diagramming Namespace

HeaderResizeEventArgs Properties

HeaderResizeEventArgs.cancel Property

Gets or sets a value indicating whether to allow the current operation.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get cancel() {}
set cancel(value) {}
```

Property Value

Boolean

true to cancel the operation, or **false** otherwise.

See Also

HeaderResizeEventArgs Members
HeaderResizeEventArgs Class
MindFusion.Diagramming Namespace

HeaderResizeEventArgs.horizontal Property

Gets a value indicating whether the user resizes the header horizontally.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get horizontal() {}
```

Property Value

Boolean

true if resizing horizontally, or **false** otherwise.

See Also

HeaderResizeEventArgs Members
HeaderResizeEventArgs Class
MindFusion.Diagramming Namespace

InplaceEditEventArgs Class

Contains the arguments passed to handlers of inplace edit related events.

Syntax

```
JavaScript
// class
InplaceEditEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.EventArgs
  MindFusion.Diagramming.InplaceEditEventArgs

See Also

InplaceEditEventArgs Members
MindFusion.Diagramming Namespace
Events Class

InplaceEditEventArgs Members

Contains the arguments passed to handlers of inplace edit related events.

The following tables list the members exposed by the InplaceEditEventArgs type.

Public Constructors

	Name	Description
	InplaceEditEventArgs	Initializes a new instance of the **InplaceEditEventArgs** class.

Public Properties

	Name	Description
	bounds	Gets the bounds of the item being edited.
	control	Gets or sets the control used to edit the item's text.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	item	Gets a reference to the item being edited.

See Also

InplaceEditEventArgs Class
MindFusion.Diagramming Namespace

InplaceEditEventArgs Constructor

Initializes a new instance of the InplaceEditEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function InplaceEditEventArgs ()
```

See Also

InplaceEditEventArgs Class
MindFusion.Diagramming Namespace

InplaceEditEventArgs Properties

InplaceEditEventArgs.bounds Property

Gets the bounds of the item being edited.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get bounds() {}
```

Property Value

Rect

A Rect instance containing item's coordinates.

See Also

InplaceEditEventArgs Members
InplaceEditEventArgs Class
MindFusion.Diagramming Namespace

InplaceEditEventArgs.control Property

Gets or sets the control used to edit the item's text.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get control() {}
set control(value) {}
```

Property Value

Element

An HTML DOM element to use as editor.

See Also

InplaceEditEventArgs Members
InplaceEditEventArgs Class
MindFusion.Diagramming Namespace

InplaceEditEventArgs.item Property

Gets a reference to the item being edited.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get item() {}
```

Property Value

DiagramItem

A DiagramItem instance.

See Also

InplaceEditEventArgs Members
InplaceEditEventArgs Class
MindFusion.Diagramming Namespace

Interaction Class

Encapsulates the current state of user interaction with the control.

Syntax

```
JavaScript
// class
Interaction.prototype = {}
```

Inheritance Hierarchy

See Also

Interaction Members
MindFusion.Diagramming Namespace

Interaction Members

Encapsulates the current state of user interaction with the control.

The following tables list the members exposed by the Interaction type.

Public Constructors

	Name	Description
	Interaction	Initializes a new instance of the **Interaction** class.

See Also

Interaction Class
MindFusion.Diagramming Namespace

Interaction Constructor

Initializes a new instance of the Interaction class.

Namespace: MindFusion.Diagramming
File: Interaction.js

Syntax

```
JavaScript
function Interaction ([diagram])
```

Parameters

diagram
Optional.

See Also

Interaction Class
MindFusion.Diagramming Namespace

InteractionState Class

Encapsulates the current state of user interaction with the control.

Syntax

```
JavaScript
// class
InteractionState.prototype = {}
```

Inheritance Hierarchy

See Also

InteractionState Members
MindFusion.Diagramming Namespace

InteractionState Members

Encapsulates the current state of user interaction with the control.

The following tables list the members exposed by the InteractionState type.

Public Constructors

	Name	Description

	InteractionState	Initializes a new instance of the **InteractionState** class.

Public Methods

	Name	Description
	cancel	Cancels the user interaction. Call this method to cancel interaction, usually in response to the user clicking the right mouse button while dragging.
	commit	Invoked to commit the changes made by the user.
	isAllowed	Checks whether the current interaction can be completed at the specified point.
	start	This method is invoked in the beginning of the interaction, usually when the user presses the left mouse button over a modification handle.
	update	This method is invoked to update the interaction state, usually in response to the user moving the mouse after the modification has started.

See Also

InteractionState Class
MindFusion.Diagramming Namespace

InteractionState Constructor

Initializes a new instance of the InteractionState class.

Namespace: MindFusion.Diagramming
File: InteractionState.js

Syntax

JavaScript

```
function InteractionState (modifiedItem, action, adjustmentHandle, point)
```

Parameters

modifiedItem
The DiagramItem to be created or modified.

action
One of the Action enumeration values, specifying what kind of action to perform.

adjustmentHandle
The index of the selection handle to be dragged with the mouse, defines what kind of modification to perform.

point
A Point specifying the mouse position where the user starts to draw or modify an item.

See Also

InteractionState Class
MindFusion.Diagramming Namespace

InteractionState Methods

InteractionState.cancel Method

Cancels the user interaction. Call this method to cancel interaction, usually in response to the user clicking the right mouse button while dragging.

Namespace: MindFusion.Diagramming
File: InteractionState.js

Syntax

JavaScript

```
function cancel (diagram)
```

Parameters

diagram
The Diagram, at which the action is performed.

Remarks

Call this method to cancel interaction, usually in response to the user clicking the right mouse button while dragging.

See Also

InteractionState Members
InteractionState Class
MindFusion.Diagramming Namespace

InteractionState.commit Method

Invoked to commit the changes made by the user.

Namespace: MindFusion.Diagramming
File: InteractionState.js

Syntax

JavaScript

```
function commit (point)
```

Parameters

point
A Point instance indicating the location of the cursor, in document coordinates.

See Also

InteractionState Members
InteractionState Class
MindFusion.Diagramming Namespace

InteractionState.isAllowed Method

Checks whether the current interaction can be completed at the specified point.

Namespace: MindFusion.Diagramming
File: InteractionState.js

Syntax

JavaScript

```
function isAllowed (point)
```

Parameters

point
A Point instance indicating the location of the cursor, in document coordinates.

Return Value

See Also

InteractionState Members
InteractionState Class
MindFusion.Diagramming Namespace

InteractionState.start Method

This method is invoked in the beginning of the interaction, usually when the user presses the left mouse button over a modification handle.

Namespace: MindFusion.Diagramming
File: InteractionState.js

Syntax

JavaScript

```
function start ()
```

See Also

InteractionState Members
InteractionState Class
MindFusion.Diagramming Namespace

InteractionState.update Method

This method is invoked to update the interaction state, usually in response to the user moving the mouse after the modification has started.

Namespace: MindFusion.Diagramming
File: InteractionState.js

Syntax

```
JavaScript
function update (point)
```

Parameters

point
A Point instance indicating the location of the cursor, in document coordinates.

See Also

InteractionState Members
InteractionState Class
MindFusion.Diagramming Namespace

ItemEventArgs Class

Contains the arguments passed to handlers of item-related events.

Syntax

```
JavaScript
// class
ItemEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
  MindFusion.Diagramming.ItemEventArgs
    MindFusion.Diagramming.LinkEventArgs
    MindFusion.Diagramming.NodeEventArgs

See Also

ItemEventArgs Members
MindFusion.Diagramming Namespace

ItemEventArgs Members

Contains the arguments passed to handlers of item-related events.

The following tables list the members exposed by the ItemEventArgs type.

Public Constructors

	Name	Description
	ItemEventArgs	Initializes a new instance of the ItemEventArgs class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user.
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	context	Gets the canvas rendering context.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	item	Gets the diagram item for which the event is raised.
	mouseButton	Gets which mouse button has been pressed.
	mousePosition	Gets the current mouse pointer position.
	newText	Gets the text that has just been entered by the user.
	oldText	Gets the text that had been displayed in the node before the user edited it.

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the creation or modification of the node.

See Also

ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs Constructor

Initializes a new instance of the ItemEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function ItemEventArgs (args)
```

Parameters

args

See Also

ItemEventArgs Methods

ItemEventArgs.cancelDrag Method

Immediately cancels the creation or modification of the node.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
function cancelDrag ()
```

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs Properties

ItemEventArgs.adjustmentHandle Property

Gets the adjustment handle being moved by the user.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get adjustmentHandle() {}
set adjustmentHandle(value) {}
```

Property Value

AdjustmentHandles

A member of the AdjustmentHandles enumeration.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs.context Property

Gets the canvas rendering context.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get context() {}
```

Property Value

CanvasRenderingContext2D

A rendering context used to draw on the Canvas element.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs.item Property

Gets the diagram item for which the event is raised.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get item() {}
```

Property Value

DiagramItem

An instance of the DiagramItem class.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs.mouseButton Property

Gets which mouse button has been pressed.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get mouseButton() {}
```

Property Value

MouseButton

A member of the MouseButton enumeration.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs.mousePosition Property

Gets the current mouse pointer position.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get mousePosition() {}
```

Property Value

Point

A Point instance containing mouse pointer coordinates.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs.newText Property

Gets the text that has just been entered by the user.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get newText() {}
```

Property Value

String

A string containing newly entered text.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemEventArgs.oldText Property

Gets the text that had been displayed in the node before the user edited it.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get oldText() {}
```

Property Value

String

A string containing item's original text.

See Also

ItemEventArgs Members
ItemEventArgs Class
MindFusion.Diagramming Namespace

ItemLabel Class

Represents a label displayed by a DiagramItem.

Syntax

JavaScript

```
// class
ItemLabel.prototype = {}
```

Remarks

This class represents supplementary labels displayed by an item, in addition to its main text property. Call the addLabel method to create a label, and customize it through various members of type-specific LinkLabel and NodeLabel classes.

A label can also display an image assigned to the image or imageLocation property. This lets you draw multiple icons per node, or along a link's geometry.

Inheritance Hierarchy

MindFusion.Diagramming.ItemLabel
MindFusion.Diagramming.LinkLabel
MindFusion.Diagramming.NodeLabel

See Also

ItemLabel Members
MindFusion.Diagramming Namespace

ItemLabel Members

Represents a label displayed by a DiagramItem.

The following tables list the members exposed by the ItemLabel type.

Public Constructors

	Name	Description
	ItemLabel	Initializes a new instance of the **ItemLabel** class.

Public Properties

	Name	Description
	brush	Gets or sets the brush used to draw the label's background.
	contentPadding	Gets or sets the distance between image and text when both are specified.
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled.
	enableWrap	Gets or sets a value indicating whether text wrapping is enabled.
	font	Gets or sets the font used to render label's text.
	horizontalAlign	Gets or sets the label's horizontal alignment.
	horizontalOffset	Gets or sets label's horizontal offset from its pin point.
	image	Gets or sets the image displayed by this label.
	imageAlign	Gets or sets the alignment of label's image relative to text.
	imageContent	Gets or sets a Base64-encoded image to display inside this label.
	imageLocation	Gets or sets URL of the image to display inside this label.
	imageSize	Gets or sets a size that overrides the native image size.
	locked	Gets or sets a value indicating whether users are allowed to edit this label.
	margin	Gets or sets the label's margin.
	maxWidth	Gets or sets the maximum width of this label.
	text	Gets or sets the label's text.

	textColor	Gets or sets the text color.
	verticalAlign	Gets or sets the label's vertical alignment.
	verticalOffset	Gets or sets label's vertical offset from its pin point.

See Also

ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel Constructor

Initializes a new instance of the ItemLabel class.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
function ItemLabel (item, text)
```

Parameters

item
DiagramItem. The DiagramItem that displays the label.

text
String. The label's text.

See Also

ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel Properties

ItemLabel.brush Property

Gets or sets the brush used to draw the label's background.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.contentPadding Property

Gets or sets the distance between image and text when both are specified.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get contentPadding() {}
set contentPadding(value) {}
```

Property Value

Number

A Number specifying the distance.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.enableStyledText Property

Gets or sets a value indicating whether styled text rendering is enabled.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get enableStyledText() {}
set enableStyledText(value) {}
```

Property Value

Boolean

true to parse the label's text in order to find formatting tags, or **false** otherwise.

Remarks

Styled text recognizes the following format tags:

Tag	Description
text	Indicates italic text.
text	Indicates bold text.
<u>text</u>	Indicates underlined text.
_{text}	Indicates subscript.
^{text}	Indicates superscript.
<color=#RRGGBB>text</color>	Changes the color of the text. The red, green and blue color channels are specified as two-digit hexadecimal numbers.
 	Indicates line break.
<a=url>text	Renders specified text as a hyperlink and raises hyperlinkClicked event when link has been clicked.
<size=N>text</size>	Changes the font size.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.enableWrap Property

Gets or sets a value indicating whether text wrapping is enabled.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get enableWrap() {}
set enableWrap(value) {}
```

Property Value

Boolean

true to wrap the label's text, or **false** otherwise.

Remarks

In order to wrap text, you must also set the maxWidth property to specify maximum line length.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.font Property

Gets or sets the font used to render label's text.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get font() {}
set font(value) {}
```

Property Value

Font

An instance of the Font class.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.horizontalAlign Property

Gets or sets the label's horizontal alignment.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get horizontalAlign() {}
set horizontalAlign(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.horizontalOffset Property

Gets or sets label's horizontal offset from its pin point.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get horizontalOffset() {}
set horizontalOffset(value) {}
```

Property Value

Number

A number specifying label's horizontal position, specified in diagram's measureUnit.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.image Property

Gets or sets the image displayed by this label.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get image() {}
set image(value) {}
```

Property Value

HTMLImageElement

An image object to render inside the label.

Remarks

If a label displays both image and text, the image position relative to text depends on the imageAlign property.

If the label is only intended to display an icon image and no text, you could also set its locked property to prevent users from entering text.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace
imageSize Property
imageLocation Property

ItemLabel.imageAlign Property

Gets or sets the alignment of label's image relative to text.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get imageAlign() {}
set imageAlign(value) {}
```

Property Value

ImageAlign

A member of the ImageAlign enumeration. The default value is **MiddleLeft**.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace
image Property
imageLocation Property

ItemLabel.imageContent Property

Gets or sets a Base64-encoded image to display inside this label.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get imageContent() {}
set imageContent(value) {}
```

Property Value

String

A Base64-encoded representation of an image.

Remarks

If a label displays both image and text, the image position relative to text depends on the imageAlign property.

If the label is only intended to display an icon image and no text, you could also set its locked property to prevent users from entering text.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.imageLocation Property

Gets or sets URL of the image to display inside this label.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get imageLocation() {}
set imageLocation(value) {}
```

Property Value

String

A URL string that identifies the image to be loaded.

Remarks

If a label displays both image and text, the image position relative to text depends on the imageAlign property.

If the label is only intended to display an icon image and no text, you could also set its locked property to prevent users from entering text.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace
image Property

ItemLabel.imageSize Property

Gets or sets a size that overrides the native image size.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get imageSize() {}
set imageSize(value) {}
```

Property Value

Size

A Size instance.

Remarks

By default, image is rendered using bitmap's intrinsic size. Set this value to increase or reduce the size of the rendered image.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace
image Property
imageLocation Property

ItemLabel.locked Property

Gets or sets a value indicating whether users are allowed to edit this label.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get locked() {}
set locked(value) {}
```

Property Value

Boolean

true to prevent users from editing this label, or **false** otherwise.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.margin Property

Gets or sets the label's margin.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get margin() {}
set margin(value) {}
```

Property Value

Thickness

A Thickness instance specifying the margin space on all sides of label.

See Also

ItemLabel Members
ItemLabel Class

ItemLabel.maxWidth Property

Gets or sets the maximum width of this label.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get maxWidth() {}
set maxWidth(value) {}
```

Property Value

Number

The maximum width.

Remarks

This value is used when enableWrap is set to **true**.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.text Property

Gets or sets the label's text.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

A string containing the label's text.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.textColor Property

Gets or sets the text color.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string value identifying the text color.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.verticalAlign Property

Gets or sets the label's vertical alignment.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

JavaScript

```
get verticalAlign() {}
set verticalAlign(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemLabel.verticalOffset Property

Gets or sets label's vertical offset from its pin point.

Namespace: MindFusion.Diagramming
File: ItemLabel.js

Syntax

```
JavaScript
get verticalOffset() {}
set verticalOffset(value) {}
```

Property Value

Number

A number specifying label's vertical position, specified in diagram's measureUnit.

See Also

ItemLabel Members
ItemLabel Class
MindFusion.Diagramming Namespace

ItemListView Class

A list view control that hosts DiagramItems and supports drag-and-drop to DiagramView.

Syntax

```
JavaScript
// class
ItemListView.prototype = {}
```

Remarks

Call the addItem method to add an item to the view.

Drag-and-drop operations create a copy of the dragged item, and set its size to newInstanceSize. The copy is created by calling item's clone method. You can override **clone** to copy property values of custom item subclasses.

Icons in the list are drawn by scaling **newInstanceSize** down to iconSize. E.g. if a list icon is half the size of the diagram item it creates, its graphic elements such as cells, labels, strokes would be rendered at half scale too (as opposed to rendering with same label / stroke sizes but in smaller bounding rectangle). The scaling code assumes measureUnit of the target diagram and the list view have matching values.

Inheritance Hierarchy

MindFusion.Controls.CanvasControl
  MindFusion.Diagramming.ListViewBase
    MindFusion.Diagramming.ItemListView

See Also

ItemListView Members
MindFusion.Diagramming Namespace
Palette Class

ItemListView Members

A list view control that hosts DiagramItems and supports drag-and-drop to DiagramView.

The following tables list the members exposed by the ItemListView type.

Public Constructors

	Name	Description
	ItemListView	Creates an instance of **ItemListView**.

Public Properties

	Name	Description
	autoConnectDistance	Gets or sets the maximum distance to a node when auto connecting links.
	autoConnectLinks	Gets or sets whether links should be automatically connected to the nearest node.
	backColor	Gets or sets the background color of the list view. (Inherited from ListViewBase.)
	columns	Gets or sets the number of columns in control's layout.
	draggedItem	Gets the currently dragged item. (Inherited from ListViewBase.)
	element	Returns a reference to the control's DOM element. (Inherited from Control.)
	enabled	Gets or sets whether mouse events are enabled. (Inherited from Control.)
	iconSize	Gets or sets the size of items displayed in this list view. (Inherited from ListViewBase.)
	items	Gets the items contained in this list view. (Inherited from ListViewBase.)
	labelPosition	Gets or sets a value indicating the position of labels relative to item icons.
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	measureUnit	Gets the unit of measure used for logical coordinates. (Inherited from CanvasControl.)
	newInstanceSize	Gets or sets the default size of items created by drag-and-drop from this list view. (Inherited from ListViewBase.)
	orientation	Gets or sets the orientation of the view layout. (Inherited from ListViewBase.)
	padding	Gets or sets the padding space between item's slot bounds and its contents. (Inherited from ListViewBase.)
	shapeLibraryLocation	Gets the location of a shape library file containing custom shape definitions. It can be set to a full URL of the file, or a relative file path. (Inherited from ListViewBase.)

	textColor	Gets or sets the color used to paint item captions. (Inherited from ListViewBase.)
	tooltipDelay	Gets or sets the delay before a tooltip is shown. (Inherited from ListViewBase.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
	addItem	Adds an item to the list view. (Inherited from ListViewBase.)
	clearAll	Removes all items from the list view. (Inherited from ListViewBase.)
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)
	clientToDoc	Transforms a point from client to document coordinates. (Inherited from CanvasControl.)
S	create	Creates and initializes a new **ItemListView** from the specified element. This method is static and can be called without creating an instance of the class.
	docToClient	Transforms a point from document to client coordinates. (Inherited from CanvasControl.)
S	find	Returns the specified **ItemListView** object. This member is static and can be invoked without creating an instance of the class.
S	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	getItemAt	Gets the item at the specified location. (Inherited from ListViewBase.)
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data. (Inherited from Control.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
	removeItem	Removes an item from the list view. (Inherited from ListViewBase.)
	selectItem	Selects the specified item. (Inherited from ListViewBase.)
S	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description
⚡	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
⚡	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
⚡	controlUnloading	Raised when the control is unloading. (Inherited from Control.)
⚡	linkSelected	Raised when a link is selected, either programmatically or by the user. (Inherited from ListViewBase.)
⚡	nodeSelected	Raised when a node is selected, either programmatically or by the user. (Inherited from ListViewBase.)
⚡	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from **shapeLibraryLocation**. (Inherited from ListViewBase.)

See Also

ItemListView Class
MindFusion.Diagramming Namespace

ItemListView Constructor

Creates an instance of ItemListView.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

JavaScript

```
function ItemListView (element)
```

Parameters

element
HTMLCanvasElement. The Canvas DOM Element this ItemListView is associated with.

See Also

ItemListView Class
MindFusion.Diagramming Namespace

ItemListView Methods

ItemListView.create Method

Creates and initializes a new ItemListView from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

```
JavaScript
function create (element)
```

Parameters

element
HTMLCanvasElement. The DOM element that the ItemListView should be attached to.

Return Value

ItemListView. An **ItemListView** object that represents the newly created view.

See Also

ItemListView Members
ItemListView Class
MindFusion.Diagramming Namespace

ItemListView.find Method

Returns the specified ItemListView object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

```
JavaScript
function find (id)
```

Parameters

id
String. The id of the ItemListView's DOM element.

Return Value

ItemListView. An ItemListView object with the specified id, if found; otherwise, null.

See Also

ItemListView Members
ItemListView Class
MindFusion.Diagramming Namespace

ItemListView Properties

ItemListView.autoConnectDistance Property

Gets or sets the maximum distance to a node when auto connecting links.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

```
JavaScript
get autoConnectDistance() {}
set autoConnectDistance(value) {}
```

Property Value

Number

A number specifying auto-connect distance.

See Also

ItemListView Members
ItemListView Class
MindFusion.Diagramming Namespace

ItemListView.autoConnectLinks Property

Gets or sets whether links should be automatically connected to the nearest node.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

```
JavaScript
get autoConnectLinks() {}
set autoConnectLinks(value) {}
```

Property Value

Boolean

true to automatically connect links, or **false** otherwise. The default value is **true**.

Remarks

If this property is enabled and there are nodes within autoConnectDistance when a link is created via drag-and-drop, the link automatically connects to the nodes. Otherwise it is created as unconnected link.

See Also

ItemListView Members
ItemListView Class
MindFusion.Diagramming Namespace

ItemListView.columns Property

Gets or sets the number of columns in control's layout.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

```
JavaScript
get columns() {}
set columns(value) {}
```

Property Value

Number

An integer value specifying the number of columns. The default value is 1.

See Also

ItemListView Members
ItemListView Class
MindFusion.Diagramming Namespace

ItemListView.labelPosition Property

Gets or sets a value indicating the position of labels relative to item icons.

Namespace: MindFusion.Diagramming
File: ItemListView.js

Syntax

```
JavaScript
get labelPosition() {}
set labelPosition(value) {}
```

Property Value

LabelPosition

A member of the LabelPosition enumeration. The default value is **Auto**.

Remarks

The default Auto value places labels to the right of item icons if using one-column layout, and below icons if showing multiple columns.

See Also

ItemListView Members
ItemListView Class
MindFusion.Diagramming Namespace

ItemListView Events

ListViewBase.linkSelected Event

Raised when a link is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
listView.linkSelected.addEventListener(onLinkSelected);
```

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.nodeSelected Event

Raised when a node is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
listView.nodeSelected.addEventListener(onNodeSelected);
```

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.shapeLibraryLoaded Event

Raised when a ShapeLibrary is loaded from shapeLibraryLocation.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
listView.shapeLibraryLoaded.addEventListener(onShapeLibraryLoaded);
```

See Also

LabelPosition Enumeration

Specifies the position of labels in a ListView control.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
LabelPosition = {}
```

Members

	Member name	Description
	Auto	Labels are positioned depending on the column count.
	Bottom	Labels are positioned below the icon.
	Left	Labels are on the left side of the icon.
	Right	Labels are on the right side of the icon.

See Also

MindFusion.Diagramming Namespace

LengthType Enumeration

Indicates how GridPanel determines dimensions of its rows and columns.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
LengthType = {}
```

Members

	Member name	Description
	Auto	The row or column is auto-sized to fit its child components.
	Relative	The size of relative elements is calculated from available space in the GridPanel proportionally to the number of other

		relative elements.

Remarks

Members of this enumeration can be assigned to the lengthType property of GridRow and GridColumn.

See Also

MindFusion.Diagramming Namespace

LinkCrossingEventArgs Class

Contains the arguments passed to handlers of the validateLinkCrossing event.

Syntax

JavaScript

```
// class
LinkCrossingEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
MindFusion.Diagramming.LinkCrossingEventArgs

See Also

LinkCrossingEventArgs Members
MindFusion.Diagramming Namespace

LinkCrossingEventArgs Members

Contains the arguments passed to handlers of the validateLinkCrossing event.

The following tables list the members exposed by the LinkCrossingEventArgs type.

Public Constructors

	Name	Description
	LinkCrossingEventArgs	Initializes a new instance of the LinkCrossingEventArgs class.

Public Properties

	Name	Description
	bottomLink	The link at lower Z position.
	bottomSegmentIndex	Index of bottom link's segment that contains the intersection point.
	intersectionPoint	Intersection point that will show crossing decoration if the event is validated.

	topLink	The link at higher Z position.
	topSegmentIndex	Index of top link's segment that contains the intersection point.

See Also

LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossingEventArgs Constructor

Initializes a new instance of the LinkCrossingEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function LinkCrossingEventArgs (args)
```

Parameters

args

See Also

LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossingEventArgs Properties

LinkCrossingEventArgs.bottomLink Property

The link at lower Z position.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get bottomLink() {}
set bottomLink(value) {}
```

Property Value

DiagramLink

An instance of the DiagramLink class.

See Also

LinkCrossingEventArgs Members
LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossingEventArgs.bottomSegmentIndex Property

Index of bottom link's segment that contains the intersection point.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get bottomSegmentIndex() {}
set bottomSegmentIndex(value) {}
```

Property Value

Number

An integer index of link's segment containing intersectionPoint.

See Also

LinkCrossingEventArgs Members
LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossingEventArgs.intersectionPoint Property

Intersection point that will show crossing decoration if the event is validated.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get intersectionPoint() {}
set intersectionPoint(value) {}
```

Property Value

Point

An instance of the Point class containing intersection coordinates.

See Also

LinkCrossingEventArgs Members
LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossingEventArgs.topLink Property

The link at higher Z position.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get topLink() {}
set topLink(value) {}
```

Property Value

DiagramLink

An instance of the DiagramLink class.

See Also

LinkCrossingEventArgs Members
LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossingEventArgs.topSegmentIndex Property

Index of top link's segment that contains the intersection point.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get topSegmentIndex() {}
set topSegmentIndex(value) {}
```

Property Value

Number

An integer index of link's segment containing intersectionPoint.

See Also

LinkCrossingEventArgs Members
LinkCrossingEventArgs Class
MindFusion.Diagramming Namespace

LinkCrossings Enumeration

Defines values that indicate how to render link intersection points.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
LinkCrossings = {}
```

Members

	Member name	Description
	Arcs	An arc is drawn at the intersection point of two links. It connects the segments of the link at higher Z position and goes over the link at lower Z.
	Cut	The intersection segments of links at lower Z position are cut by links at higher Z position.
	Straight	No special indication for link intersection points.

Remarks

Members of this enumeration can be assigned to the linkCrossings property of Diagram.

See Also

MindFusion.Diagramming Namespace

LinkEventArgs Class

Contains the arguments passed to handlers of link-related events.

Syntax

```
JavaScript
// class
LinkEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ItemEventArgs
  MindFusion.Diagramming.LinkEventArgs
    MindFusion.Diagramming.LinkValidationEventArgs

See Also

LinkEventArgs Members
MindFusion.Diagramming Namespace
Events Class

LinkEventArgs Members

Contains the arguments passed to handlers of link-related events.

The following tables list the members exposed by the LinkEventArgs type.

Public Constructors

	Name	Description
	LinkEventArgs	Initializes a new instance of the **LinkEventArgs** class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user. (Inherited from ItemEventArgs.)
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	context	Gets the canvas rendering context. (Inherited from ItemEventArgs.)
	destination	Gets the candidate destination node; reported to validation events.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	item	Gets the diagram item for which the event is raised. (Inherited from ItemEventArgs.)
	label	Gets the link label for which the event is raised.
	link	Gets the link for which the event is raised.
	mouseButton	Gets which mouse button has been pressed. (Inherited from ItemEventArgs.)
	mousePosition	Gets the current mouse pointer position. (Inherited from ItemEventArgs.)
	newText	Gets the text that has just been entered by the user. (Inherited from ItemEventArgs.)
	oldText	Gets the text that had been displayed in the node before the user edited it. (Inherited from ItemEventArgs.)
	origin	Gets the candidate origin node; reported to validation events.

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the creation or modification of the node. (Inherited from ItemEventArgs.)

See Also

LinkEventArgs Class
MindFusion.Diagramming Namespace

LinkEventArgs Constructor

Initializes a new instance of the LinkEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
function LinkEventArgs (args)
```

Parameters

args

See Also

LinkEventArgs Class
MindFusion.Diagramming Namespace

LinkEventArgs Properties

LinkEventArgs.destination Property

Gets the candidate destination node; reported to validation events.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get destination() {}
set destination(value) {}
```

Property Value

DiagramNode

A DiagramNode instance.

Remarks

When handling linkCreating and linkModifying validation events, origin and **destination** values of LinkEventArgs are generally different from link's current origin and destination. If you do not cancel the operation, the former are assigned to the latter once the user stops drawing the link.

See Also

LinkEventArgs Members
LinkEventArgs Class

LinkEventArgs.label Property

Gets the link label for which the event is raised.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get label() {}
```

Property Value

LinkLabel

A LinkLabel instance.

See Also

LinkEventArgs Members
LinkEventArgs Class
MindFusion.Diagramming Namespace

LinkEventArgs.link Property

Gets the link for which the event is raised.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get link() {}
```

Property Value

DiagramLink

A DiagramLink instance.

See Also

LinkEventArgs Members
LinkEventArgs Class
MindFusion.Diagramming Namespace

LinkEventArgs.origin Property

Gets the candidate origin node; reported to validation events.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get origin() {}
set origin(value) {}
```

Property Value

DiagramNode

A DiagramNode instance.

Remarks

When handling linkCreating and linkModifying validation events, **origin** and destination values of LinkEventArgs are generally different from link's current origin and destination. If you do not cancel the operation, the former are assigned to the latter once the user stops drawing the link.

See Also

LinkEventArgs Members
LinkEventArgs Class
MindFusion.Diagramming Namespace
linkCreating Event
linkModifying Event

LinkLabel Class

Represents a label displayed by a DiagramLink. New labels can be added to a link by calling its addLabel method.

Syntax

```
JavaScript
// class
LinkLabel.prototype = {}
```

Remarks

The position of a label is determined by where it is pinned along the link's geometry, and offset from the pin point, as set using setControlPointPosition, setSegmentPosition and setLinkLengthPosition methods. In addition, the horizontalAlign and verticalAlign properties specify whether the label is centered on its pin point or aligned to it.

Set the autoArrange property to let the diagram set label's position automatically, avoiding overlaps with nodes and other link labels.

You can let users arrange labels manually by setting the behavior property of DiagramView to MoveLabels.

Inheritance Hierarchy

MindFusion.Diagramming.ItemLabel
MindFusion.Diagramming.LinkLabel

See Also

LinkLabel Members
MindFusion.Diagramming Namespace

LinkLabel Members

Represents a label displayed by a DiagramLink. New labels can be added to a link by calling its addLabel method.

The following tables list the members exposed by the LinkLabel type.

Public Constructors

	Name	Description
	LinkLabel	Initializes a new instance of the LinkLabel class.

Public Properties

	Name	Description
	autoArrange	Gets or sets whether to automatically arrange the label to avoid overlapping nodes.
	brush	Gets or sets the brush used to draw the label's background.
	contentPadding	Gets or sets the distance between image and text when both are specified. (Inherited from ItemLabel.)
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled. (Inherited from ItemLabel.)
	enableWrap	Gets or sets a value indicating whether text wrapping is enabled. (Inherited from ItemLabel.)
	font	Gets or sets the font used to render label's text.
	horizontalAlign	Gets or sets the label's horizontal alignment.
	horizontalOffset	Gets or sets label's horizontal offset from its pin point. (Inherited from ItemLabel.)
	image	Gets or sets the image displayed by this label. (Inherited from ItemLabel.)
	imageAlign	Gets or sets the alignment of label's image relative to text. (Inherited from ItemLabel.)
	imageContent	Gets or sets a Base64-encoded image to display inside this label. (Inherited from ItemLabel.)
	imageLocation	Gets or sets URL of the image to display inside this label. (Inherited from ItemLabel.)
	imageSize	Gets or sets a size that overrides the native image size. (Inherited from ItemLabel.)

	locked	Gets or sets a value indicating whether users are allowed to edit this label. (Inherited from ItemLabel.)
	margin	Gets or sets the label's margin.
	maxWidth	Gets or sets the maximum width of this label. (Inherited from ItemLabel.)
	text	Gets or sets the label's text.
	textColor	Gets or sets the text color.
	verticalAlign	Gets or sets the label's vertical alignment.
	verticalOffset	Gets or sets label's vertical offset from its pin point. (Inherited from ItemLabel.)

Public Methods

	Name	Description
	setControlPointPosition	Positions this label relatively to the specified control point.
	setLinkLengthPosition	Positions this label relatively to the link length.
	setSegmentPosition	Positions this label relatively to the specified link segment.

See Also

LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel Constructor

Initializes a new instance of the LinkLabel class.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
function LinkLabel (link, text)
```

Parameters

link
DiagramLink. The DiagramLink that displays the label.

text
String. The label's text.

See Also

LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel Methods

LinkLabel.setControlPointPosition Method

Positions this label relatively to the specified control point.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

```
JavaScript
function setControlPointPosition (controlPoint, dx, dy)
```

Parameters

controlPoint
Number. The control point index.

dx
Number. Horizontal offset from the control point.

dy
Number. Vertical offset from the control point.

Remarks

This method sets the specified control point as label's pin point and assigns specified offsets to horizontalOffset and verticalOffset properties.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.setLinkLengthPosition Method

Positions this label relatively to the link length.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

```
JavaScript
function setLinkLengthPosition (lengthFactor)
```

Parameters

lengthFactor
Number. A value between 0 and 1 specifying the position of the label relatively to the link's length.

Remarks

This method sets the point at specified distance along link's geometry as pin point and assigns specified offsets to horizontalOffset and verticalOffset properties.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.setSegmentPosition Method

Positions this label relatively to the specified link segment.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
function setSegmentPosition (segment, dx, dy)
```

Parameters

segment
Number. The segment index.

dx
Number. Horizontal offset from the segment center.

dy
Number. Vertical offset from the segment center.

Remarks

This method sets the middle of specified segment as label's pin point and assigns specified offsets to horizontalOffset and verticalOffset properties.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel Properties

LinkLabel.autoArrange Property

Gets or sets whether to automatically arrange the label to avoid overlapping nodes.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get autoArrange() {}
set autoArrange(value) {}
```

Property Value

Boolean

true if the label should be auto-arranged, or **false** otherwise.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.brush Property

Gets or sets the brush used to draw the label's background.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.font Property

Gets or sets the font used to render label's text.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get font() {}
set font(value) {}
```

Property Value

Font

An instance of the Font class.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.horizontalAlign Property

Gets or sets the label's horizontal alignment.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get horizontalAlign() {}
set horizontalAlign(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.margin Property

Gets or sets the label's margin.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get margin() {}
set margin(value) {}
```

Property Value

Thickness

A Thickness instance specifying the margin space on all sides of label.

See Also

LinkLabel Members
LinkLabel Class

LinkLabel.text Property

Gets or sets the label's text.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

A string containing the label's text.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.textColor Property

Gets or sets the text color.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

JavaScript

```
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string value identifying the text color.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkLabel.verticalAlign Property

Gets or sets the label's vertical alignment.

Namespace: MindFusion.Diagramming
File: LinkLabel.js

Syntax

```
JavaScript
get verticalAlign() {}
set verticalAlign(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

LinkLabel Members
LinkLabel Class
MindFusion.Diagramming Namespace

LinkNodesBehavior Class

The base class for behavior types that allow drawing both nodes and links.

Syntax

```
JavaScript
// class
LinkNodesBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.LinkNodesBehavior

See Also

LinkNodesBehavior Members
MindFusion.Diagramming Namespace

LinkNodesBehavior Members

The base class for behavior types that allow drawing both nodes and links.

The following tables list the members exposed by the LinkNodesBehavior type.

Public Constructors

	Name	Description
	LinkNodesBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

LinkNodesBehavior Class
MindFusion.Diagramming Namespace

LinkNodesBehavior Constructor

Overload List

Name	Description
LinkNodesBehavior	Initializes a new instance of the LinkNodesBehavior class.
LinkNodesBehavior	Initializes a new instance of the LinkNodesBehavior class.
LinkNodesBehavior	Initializes a new instance of the LinkNodesBehavior class.

See Also

LinkNodesBehavior Class
MindFusion.Diagramming Namespace

LinkNodesBehavior Constructor

Initializes a new instance of the LinkNodesBehavior class.

Namespace: MindFusion.Diagramming
File: LinkNodesBehavior.js

Syntax

JavaScript
`function LinkNodesBehavior ()`

See Also

LinkNodesBehavior Constructor Overload List
LinkNodesBehavior Class
MindFusion.Diagramming Namespace

LinkNodesBehavior Constructor

Initializes a new instance of the LinkNodesBehavior class.

Namespace: MindFusion.Diagramming
File: LinkNodesBehavior.js

Syntax

JavaScript

```
function LinkNodesBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

LinkNodesBehavior Constructor Overload List
LinkNodesBehavior Class
MindFusion.Diagramming Namespace

LinkNodesBehavior Constructor

Initializes a new instance of the LinkNodesBehavior class.

Namespace: MindFusion.Diagramming
File: LinkNodesBehavior.js

Syntax

JavaScript

```
function LinkNodesBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

LinkNodesBehavior Constructor Overload List
LinkNodesBehavior Class
MindFusion.Diagramming Namespace

LinkNodesBehavior Methods

LinkNodesBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: LinkNodesBehavior.js

Syntax

JavaScript

```
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

LinkNodesBehavior Members
LinkNodesBehavior Class
MindFusion.Diagramming Namespace

LinkShape Enumeration

Specifies available styles for the segments of links.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
LinkShape = {}
```

Members

	Member name	Description
	Bezier	The segments of a link are Bézier curves.
	Cascading	The segments of a link are alternating horizontal and vertical lines, orthogonal to each other.
	Polyline	The segments of a link are straight lines.
	Spline	The segments of a link are curves forming an interpolating spline that passes through all control points.

Remarks

Members of this enumeration can be assigned to the shape property of DiagramLink.

See Also

MindFusion.Diagramming Namespace

LinkShapesBehavior Class

This class is used when view's behavior is set to Behavior.LinkShapes.

Syntax

JavaScript
`// class` `LinkShapesBehavior.prototype = {}`

Inheritance Hierarchy

See Also

LinkShapesBehavior Members
MindFusion.Diagramming Namespace

LinkShapesBehavior Members

This class is used when view's behavior is set to Behavior.LinkShapes.

The following tables list the members exposed by the LinkShapesBehavior type.

Public Constructors

	Name	Description
	LinkShapesBehavior	Initializes a new instance of the **LinkShapesBehavior** class.

Public Methods

	Name	Description
	createNode	LinkNodesBehavior.CreateNode override.

See Also

LinkShapesBehavior Class
MindFusion.Diagramming Namespace

LinkShapesBehavior Constructor

Initializes a new instance of the LinkShapesBehavior class.

Namespace: MindFusion.Diagramming
File: LinkShapesBehavior.js

Syntax

```
JavaScript
function LinkShapesBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

LinkShapesBehavior Class
MindFusion.Diagramming Namespace

LinkShapesBehavior Methods

LinkShapesBehavior.createNode Method

LinkNodesBehavior.CreateNode override.

Namespace: MindFusion.Diagramming
File: LinkShapesBehavior.js

Syntax

```
JavaScript
function createNode ()
```

Return Value

An instance of the DiagramNode class.

See Also

LinkShapesBehavior Members
LinkShapesBehavior Class
MindFusion.Diagramming Namespace

LinkTextStyle Enumeration

Specifies the position and alignment of links' labels.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
LinkTextStyle = {}
```

Members

	Member name	Description
	Center	The text is displayed horizontally above the middle link segment or control point, depending on whetherthere are an odd or even number of segments.
	Follow	The label text starts from the first point of the link and follows the path defined by the link segments. This style is useful for displaying long text.
	MiddleSegment	The text is displayed horizontally above the middle link segment. If there are an even number of segments, the label is drawn over the longer from the two segments adjacent to the middle control point.
	MiddleSegmentRotated	The text is displayed rotated at the same angle as the segment where the text is placed. A long enough segmentfrom the middle link segments is chosen for that purpose.
	OverLongestSegment	The text follows the longest link segment and is displayed centered at the segment's middle point.
	Rotate	The text is displayed rotated at the same angle as the segment where the text is placed.A long enough segment from the middle link segments is chosen for that purpose.

Remarks

Members of this enumeration can be assigned to the textStyle property of DiagramLink.

See Also

MindFusion.Diagramming Namespace

LinkValidationEventArgs Class

Contains the arguments passed to handlers of link validation events.

Syntax

JavaScript

```
// class
LinkValidationEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
  MindFusion.Diagramming.ItemEventArgs
    MindFusion.Diagramming.LinkEventArgs
      MindFusion.Diagramming.LinkValidationEventArgs

See Also

LinkValidationEventArgs Members

Contains the arguments passed to handlers of link validation events.

The following tables list the members exposed by the LinkValidationEventArgs type.

Public Constructors

	Name	Description
	LinkValidationEventArgs	Initializes a new instance of the LinkValidationEventArgs class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user. (Inherited from ItemEventArgs.)
	anchorIndex	Gets the zero-based index of the anchor point related to the event.
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	context	Gets the canvas rendering context. (Inherited from ItemEventArgs.)
	destination	Gets the candidate destination node; reported to validation events. (Inherited from LinkEventArgs.)
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	item	Gets the diagram item for which the event is raised. (Inherited from ItemEventArgs.)
	label	Gets the link label for which the event is raised. (Inherited from LinkEventArgs.)
	link	Gets the link for which the event is raised. (Inherited from LinkEventArgs.)
	mouseButton	Gets which mouse button has been pressed. (Inherited from ItemEventArgs.)
	mousePosition	Gets the current mouse pointer position. (Inherited from ItemEventArgs.)
	newText	Gets the text that has just been entered by the user. (Inherited from ItemEventArgs.)
	node	Gets the node related to the event.
	oldText	Gets the text that had been displayed in the node before the user edited it. (Inherited from ItemEventArgs.)

	origin	Gets the candidate origin node; reported to validation events. (Inherited from LinkEventArgs.)
	tableRow	Gets the zero-based index of the table row related to the event.

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the creation or modification of the node. (Inherited from ItemEventArgs.)

See Also

LinkValidationEventArgs Class
MindFusion.Diagramming Namespace

LinkValidationEventArgs Constructor

Initializes a new instance of the LinkValidationEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function LinkValidationEventArgs (args)
```

Parameters

args

See Also

LinkValidationEventArgs Class
MindFusion.Diagramming Namespace

LinkValidationEventArgs Properties

LinkValidationEventArgs.anchorIndex Property

Gets the zero-based index of the anchor point related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get anchorIndex() {}
set anchorIndex(value) {}
```

Property Value

Number

The anchor point index.

See Also

LinkValidationEventArgs Members
LinkValidationEventArgs Class
MindFusion.Diagramming Namespace

LinkValidationEventArgs.node Property

Gets the node related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get node() {}
set node(value) {}
```

Property Value

DiagramNode

A DiagramNode instance.

See Also

LinkValidationEventArgs Members
LinkValidationEventArgs Class
MindFusion.Diagramming Namespace

LinkValidationEventArgs.tableRow Property

Gets the zero-based index of the table row related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get tableRow() {}
set tableRow(value) {}
```

Property Value

Number

The table row index.

See Also

ListViewBase Class

The base class of ItemListView and NodeListView controls.

Syntax

JavaScript

```
// class
ListViewBase.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CanvasControl
  MindFusion.Diagramming.ListViewBase
    MindFusion.Diagramming.ItemListView
    MindFusion.Diagramming.NodeListView

See Also

ListViewBase Members
MindFusion.Diagramming Namespace

ListViewBase Members

The base class of ItemListView and NodeListView controls.

The following tables list the members exposed by the ListViewBase type.

Public Properties

	Name	Description
	backColor	Gets or sets the background color of the list view.
	draggedItem	Gets the currently dragged item.
	element	Returns a reference to the control's DOM element. (Inherited from Control.)
	enabled	Gets or sets whether mouse events are enabled. (Inherited from Control.)
	iconSize	Gets or sets the size of items displayed in this list view.
	items	Gets the items contained in this list view.
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	measureUnit	Gets the unit of measure used for logical coordinates. (Inherited from CanvasControl.)

	newInstanceSize	Gets or sets the default size of items created by drag-and-drop from this list view.
	orientation	Gets or sets the orientation of the view layout.
	padding	Gets or sets the padding space between item's slot bounds and its contents.
	shapeLibraryLocation	Gets the location of a shape library file containing custom shape definitions. It can be set to a full URL of the file, or a relative file path.
	textColor	Gets or sets the color used to paint item captions.
	tooltipDelay	Gets or sets the delay before a tooltip is shown.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
	addItem	Adds an item to the list view.
	clearAll	Removes all items from the list view.
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)
	clientToDoc	Transforms a point from client to document coordinates. (Inherited from CanvasControl.)
	docToClient	Transforms a point from document to client coordinates. (Inherited from CanvasControl.)
S	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	getItemAt	Gets the item at the specified location.
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data. (Inherited from Control.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
	removeItem	Removes an item from the list view.
	selectItem	Selects the specified item.
S	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description

	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
	controlUnloading	Raised when the control is unloading. (Inherited from Control.)
	linkSelected	Raised when a link is selected, either programmatically or by the user.
	nodeSelected	Raised when a node is selected, either programmatically or by the user.
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from **shapeLibraryLocation**.

See Also

ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase Methods

ListViewBase.addItem Method

Adds an item to the list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
function addItem (item, [caption, [locked, [inactive]]])
```

Parameters

item
DiagramItem. The item to add.

caption
Optional.
String. The caption to display for the item.

locked
Optional.
Boolean. Indicates whether the item can be dragged.

inactive
Optional.
Boolean. Indicates whether the item can be selected.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

items Property
removeItem Method
clearAll Method

ListViewBase.clearAll Method

Removes all items from the list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
function clearAll ()
```

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace
addItem Method
removeItem Method

ListViewBase.getItemAt Method

Gets the item at the specified location.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
function getItemAt (point)
```

Parameters

point
Point. The location where to look.

Return Value

DiagramItem. The item found.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.removeItem Method

Removes an item from the list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

```
JavaScript
function removeItem (item)
```

Parameters

item
DiagramItem. The item to remove.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace
addItem Method
clearAll Method

ListViewBase.selectItem Method

Selects the specified item.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

```
JavaScript
function selectItem (item)
```

Parameters

item
DiagramItem. The item to select.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase Properties

ListViewBase.backColor Property

Gets or sets the background color of the list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

```
JavaScript
get backColor() {}
set backColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.draggedItem Property

Gets the currently dragged item.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

```
JavaScript
get draggedItem() {}
set draggedItem(value) {}
```

Property Value

DiagramItem

The dragged item.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.iconSize Property

Gets or sets the size of items displayed in this list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

```
JavaScript
get iconSize() {}
set iconSize(value) {}
```

Property Value

Size

An instance of the Size class specifying item size.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace
newInstanceSize Property
measureUnit Property

ListViewBase.items Property

Gets the items contained in this list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
get items() {}
set items(value) {}
```

Property Value

Array

An array of DiagramItem objects to show in the list view.

Remarks

You must call addItem to add items to the list view, instead of modifying this array directly.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace
addItem Method
removeItem Method
clearAll Method

ListViewBase.newInstanceSize Property

Gets or sets the default size of items created by drag-and-drop from this list view.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
get newInstanceSize() {}
set newInstanceSize(value) {}
```

Property Value

Size

An instance of the Size class specifying item size.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace
iconSize Property
measureUnit Property

ListViewBase.orientation Property

Gets or sets the orientation of the view layout.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
get orientation() {}
set orientation(value) {}
```

Property Value

Orientation

A member of the Orientation enumeration.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.padding Property

Gets or sets the padding space between item's slot bounds and its contents.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
get padding() {}
set padding(value) {}
```

Property Value

Number

A number that specifies the padding space size.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.shapeLibraryLocation Property

Gets the location of a shape library file containing custom shape definitions. It can be set to a full URL of the file, or a relative file path.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
get shapeLibraryLocation() {}
set shapeLibraryLocation(value) {}
```

Property Value

String

URL of a shape library file containing custom shape definitions.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.textColor Property

Gets or sets the color used to paint item captions.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

JavaScript

```
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase.tooltipDelay Property

Gets or sets the delay before a tooltip is shown.

Namespace: MindFusion.Diagramming
File: ListViewBase.js

Syntax

```
JavaScript
get tooltipDelay() {}
set tooltipDelay(value) {}
```

Property Value

Number

The tooltip delay in milliseconds.

See Also

ListViewBase Members
ListViewBase Class
MindFusion.Diagramming Namespace

ListViewBase Events

MagnifyBehavior Class

This class is used when view's behavior is set to Behavior.Magnify.

Syntax

```
JavaScript
// class
MagnifyBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.MagnifyBehavior

See Also

MagnifyBehavior Members
MindFusion.Diagramming Namespace

MagnifyBehavior Members

This class is used when view's behavior is set to Behavior.Magnify.

The following tables list the members exposed by the MagnifyBehavior type.

Public Constructors

	Name	Description
	MagnifyBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	Creates a controller appropriate for current context and pointer position. (Inherited from BehaviorBase.)
	onMouseDown	BehaviorBase.onMouseDown override.
	onMouseMove	BehaviorBase.onMouseMove override.
	onMouseUp	BehaviorBase.onMouseUp override.

See Also

MagnifyBehavior Class
MindFusion.Diagramming Namespace

MagnifyBehavior Constructor

Overload List

Name	Description
MagnifyBehavior	Initializes a new instance of the MagnifyBehavior class.
MagnifyBehavior	Initializes a new instance of the MagnifyBehavior class.

See Also

MagnifyBehavior Class
MindFusion.Diagramming Namespace

MagnifyBehavior Constructor

Initializes a new instance of the MagnifyBehavior class.

Namespace: MindFusion.Diagramming
File: MagnifyBehavior.js

Syntax

JavaScript

```
function MagnifyBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

MagnifyBehavior Constructor Overload List
MagnifyBehavior Class
MindFusion.Diagramming Namespace

MagnifyBehavior Constructor

Initializes a new instance of the MagnifyBehavior class.

Namespace: MindFusion.Diagramming
File: MagnifyBehavior.js

Syntax

JavaScript

```
function MagnifyBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

MagnifyBehavior Constructor Overload List
MagnifyBehavior Class
MindFusion.Diagramming Namespace

MagnifyBehavior Methods

MagnifyBehavior.onMouseDown Method

BehaviorBase.onMouseDown override.

Namespace: MindFusion.Diagramming
File: MagnifyBehavior.js

Syntax

```
JavaScript
function onMouseDown (e)
```

Parameters

e
A **MouseButtonEventArgs** that contains the event data.

Return Value

Boolean. true if interaction starts.

See Also

MagnifyBehavior Members
MagnifyBehavior Class
MindFusion.Diagramming Namespace

MagnifyBehavior.onMouseMove Method

BehaviorBase.onMouseMove override.

Namespace: MindFusion.Diagramming
File: MagnifyBehavior.js

Syntax

```
JavaScript
function onMouseMove (e)
```

Parameters

e
A **MouseEventArgs** that contains the event data.

See Also

MagnifyBehavior Members
MagnifyBehavior Class
MindFusion.Diagramming Namespace

MagnifyBehavior.onMouseUp Method

BehaviorBase.onMouseUp override.

Namespace: MindFusion.Diagramming
File: MagnifyBehavior.js

Syntax

```
JavaScript
```

```
function onMouseUp (e)
```

Parameters

e
A **MouseButtonEventArgs** that contains the event data.

See Also

MagnifyBehavior Members
MagnifyBehavior Class
MindFusion.Diagramming Namespace

Manipulator Class

Manipulator is an abstract base class which defines the behavior of miscellaneous UI widgets attached to nodes.

Syntax

JavaScript

```
// class
Manipulator.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Container
MindFusion.Diagramming.Manipulator

See Also

Manipulator Members
MindFusion.Diagramming Namespace

Manipulator Members

Manipulator is an abstract base class which defines the behavior of miscellaneous UI widgets attached to nodes.

The following tables list the members exposed by the Manipulator type.

Public Constructors

	Name	Description
	Manipulator	Initializes a new instance of the Manipulator class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)

	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

See Also

Manipulator Class
MindFusion.Diagramming Namespace

Manipulator Constructor

Initializes a new instance of the Manipulator class.

Namespace: MindFusion.Diagramming
File: Manipulator.js

Syntax

JavaScript

```
function Manipulator ([node])
```

Parameters

node
Optional.
DiagramNode. The DiagramNode containing this manipulator.

See Also

Manipulator Class
MindFusion.Diagramming Namespace

MarkStyle Enumeration

Specifies the visual style of anchor point marks.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
MarkStyle = {}
```

Members

	Member name	Description
	Circle	Anchor point marked with a circle.
	Cross	Anchor point marked with a cross.
	Custom	Raises drawAnchorPoint event, which lets you draw custom graphics using the Canvas API.
	None	No visual indication for the anchor point.
	Rectangle	Anchor point marked with a rectangle.
	X	Anchor point marked with an 'X'.

Remarks

Members of this enumeration can be assigned to the markStyle property of AnchorPoint.

See Also

MindFusion.Diagramming Namespace

ModificationStart Enumeration

Specifies how the user can start modifying an item.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
ModificationStart = {}
```

Members

	Member name	Description
	AutoHandles	Adjustment handles appear automatically around the object under the mouse; it can be modified without selecting it first.
	SelectedOnly	Only selected objects can be modified.

Remarks

Members of this enumeration can be assigned to the modificationStart property of DiagramView.

See Also

MindFusion.Diagramming Namespace

ModifierKeyAction Enumeration

Defines the actions that can be assigned to a modifier key such as CTRL or ALT.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
ModifierKeyAction = {}
```

Members

	Member name	Description
	Magnify	Magnifies the diagram objects under the mouse while the modifier key is pressed down.
	None	Specifies that a modifier key does not have any specific function assigned to it.
	OverrideBehavior	While the modifier key is pressed down, dragging the mouse starts creating a new item or drawing a selection rectangle instead of modifying a selected item. Otherwise, dragging the mouse could start modification of the selected item.
	Pan	Pan the view if the mouse is dragged while the modifier key is pressed down.
	Select	Start drawing a selection rectangle if the mouse is dragged while a modifier key is pressed down. Toggle the selection if an item is clicked while the key is down.
	Zoom	Start drawing a zoom-lasso if the mouse is dragged while respective modifier key is pressed down.

Remarks

Members of this enumeration can be assigned to the properties of view's modifierKeyActions.

See Also

MindFusion.Diagramming Namespace

ModifierKeyActions Class

Maps keyboard modifier keys to special actions in MindFusion.Diagramming such as panning and drawing selection rectangle.

Syntax

JavaScript

```
// class
ModifierKeyActions.prototype = {}
```

Remarks

The modifierKeyActions property of DiagramView returns an object from this class, letting you specify the modifier key functions for each view instance.

Assign member of the ModifierKeyAction enumeration to the alt, control and shift properties to specify what function is assigned to the ALT, CTRL and SHIFT keys.

Inheritance Hierarchy

MindFusion.Diagramming.ModifierKeyActions

See Also

ModifierKeyActions Members
MindFusion.Diagramming Namespace

ModifierKeyActions Members

Maps keyboard modifier keys to special actions in MindFusion.Diagramming such as panning and drawing selection rectangle.

The following tables list the members exposed by the ModifierKeyActions type.

Public Constructors

	Name	Description
	ModifierKeyActions	Initializes a new instance of the ModifierKeyActions class.

Public Properties

	Name	Description
	alt	Gets or sets the action triggered by the ALT key.
	control	Gets or sets the action triggered by the CTRL key.
	shift	Gets or sets the action triggered by the SHIFT key.

See Also

ModifierKeyActions Class
MindFusion.Diagramming Namespace

ModifierKeyActions Constructor

Initializes a new instance of the ModifierKeyActions class.

Namespace: MindFusion.Diagramming
File: ModifierKeyActions.js

Syntax

JavaScript

```
function ModifierKeyActions ()
```

See Also

ModifierKeyActions Class
MindFusion.Diagramming Namespace

ModifierKeyActions Properties

ModifierKeyActions.alt Property

Gets or sets the action triggered by the ALT key.

Namespace: MindFusion.Diagramming
File: ModifierKeyActions.js

Syntax

JavaScript

```
get alt() {}
set alt(value) {}
```

Property Value

ModifierKeyAction

A member of the ModifierKeyAction enumeration.

Remarks

The value of this property specifies what happens when a user holds down the ALT key and drags the mouse. The default value Pan makes the control enter pan mode, letting the user scroll the diagram contents by dragging it with the mouse. You can change the function of ALT by assigning a different value to this property.

See Also

ModifierKeyActions Members
ModifierKeyActions Class
MindFusion.Diagramming Namespace

ModifierKeyActions.control Property

Gets or sets the action triggered by the CTRL key.

Namespace: MindFusion.Diagramming
File: ModifierKeyActions.js

Syntax

JavaScript

```
get control() {}
set control(value) {}
```

Property Value

ModifierKeyAction

A member of the ModifierKeyAction enumeration.

Remarks

The value of this property specifies what happens when a user holds down the CTRL key and drags the mouse. The default value Select makes the control enter selection mode, letting the user draw a lasso rectangle around diagram items in order to select them. You can change the function of CTRL by assigning a different value to this property.

See Also

ModifierKeyActions Members
ModifierKeyActions Class
MindFusion.Diagramming Namespace

ModifierKeyActions.shift Property

Gets or sets the action triggered by the SHIFT key.

Namespace: MindFusion.Diagramming
File: ModifierKeyActions.js

Syntax

JavaScript

```
get shift() {}
set shift(value) {}
```

Property Value

ModifierKeyAction

A member of the ModifierKeyAction enumeration.

Remarks

The value of this property specifies what happens when a user holds down the SHIFT key and drags the mouse. The default value OverrideBehavior makes the control ignore item's selection handles, thus not allowing modifying a selected item but starts the creation of a new item. You can change the function of SHIFT by assigning a different value to this property.

See Also

ModifierKeyActions Members
ModifierKeyActions Class
MindFusion.Diagramming Namespace

ModifyBehavior Class

This class is used when view's behavior is set to Behavior.Modify.

Syntax

JavaScript

```
// class
ModifyBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.ModifyBehavior

See Also

ModifyBehavior Members
MindFusion.Diagramming Namespace

ModifyBehavior Members

This class is used when view's behavior is set to Behavior.Modify.

The following tables list the members exposed by the ModifyBehavior type.

Public Constructors

	Name	Description
	ModifyBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

ModifyBehavior Class
MindFusion.Diagramming Namespace

ModifyBehavior Constructor

Overload List

Name	Description
ModifyBehavior	Initializes a new instance of the ModifyBehavior class.
ModifyBehavior	Initializes a new instance of the ModifyBehavior class.

See Also

ModifyBehavior Class
MindFusion.Diagramming Namespace

ModifyBehavior Constructor

Initializes a new instance of the ModifyBehavior class.

Namespace: MindFusion.Diagramming
File: ModifyBehavior.js

Syntax

JavaScript

```
function ModifyBehavior (parent)
```

Parameters

parent
The Diagram this behavior will be associated with.

See Also

ModifyBehavior Constructor Overload List
ModifyBehavior Class
MindFusion.Diagramming Namespace

ModifyBehavior Constructor

Initializes a new instance of the ModifyBehavior class.

Namespace: MindFusion.Diagramming
File: ModifyBehavior.js

Syntax

JavaScript

```
function ModifyBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

ModifyBehavior Constructor Overload List
ModifyBehavior Class
MindFusion.Diagramming Namespace

ModifyBehavior Methods

ModifyBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: ModifyBehavior.js

Syntax

JavaScript

```
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

ModifyBehavior Members
ModifyBehavior Class
MindFusion.Diagramming Namespace

ModifyLinkController Class

A controller used to modify existing links in the diagram.

Syntax

JavaScript

```
// class
ModifyLinkController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
  MindFusion.Diagramming.ModifyLinkController

See Also

ModifyLinkController Members
MindFusion.Diagramming Namespace

ModifyLinkController Members

A controller used to modify existing links in the diagram.

The following tables list the members exposed by the ModifyLinkController type.

Public Constructors

	Name	Description
	ModifyLinkController	Initializes a new instance of the ModifyLinkController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	SinglePointerController.commit override.
	drawInteraction	Called to draw current state of user interaction. (Inherited from SinglePointerController.)
	move	SinglePointerController.move override.
	start	SinglePointerController.start override.
	validate	SinglePointerController.validate override.

See Also

ModifyLinkController Class
MindFusion.Diagramming Namespace

ModifyLinkController Constructor

Initializes a new instance of the ModifyLinkController class.

Namespace: MindFusion.Diagramming
File: ModifyLinkController.js

Syntax

JavaScript

```
function ModifyLinkController (link, adjustmentHandle)
```

Parameters

link
DiagramLink. The link that will be modified by this controller.

adjustmentHandle
Object. An object representing the adjustment handle used to modify the link.

See Also

ModifyLinkController Methods

ModifyLinkController.commit Method

SinglePointerController.commit override.

Namespace: MindFusion.Diagramming
File: ModifyLinkController.js

Syntax

```
JavaScript
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifyLinkController Members
ModifyLinkController Class
MindFusion.Diagramming Namespace

ModifyLinkController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: ModifyLinkController.js

Syntax

```
JavaScript
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifyLinkController Members
ModifyLinkController Class
MindFusion.Diagramming Namespace

ModifyLinkController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: ModifyLinkController.js

Syntax

```
JavaScript
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifyLinkController Members
ModifyLinkController Class
MindFusion.Diagramming Namespace

ModifyLinkController.validate Method

SinglePointerController.validate override.

Namespace: MindFusion.Diagramming
File: ModifyLinkController.js

Syntax

```
JavaScript
function validate (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

Return Value

Boolean. true to accept the operation, or false otherwise.

See Also

ModifyLinkController Members
ModifyLinkController Class
MindFusion.Diagramming Namespace

ModifyNodeController Class

A controller used to modify existing nodes in the diagram.

Syntax

```
JavaScript
// class
ModifyNodeController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
  MindFusion.Diagramming.ModifyNodeController

See Also

ModifyNodeController Members
MindFusion.Diagramming Namespace

ModifyNodeController Members

A controller used to modify existing nodes in the diagram.

The following tables list the members exposed by the ModifyNodeController type.

Public Constructors

	Name	Description
	ModifyNodeController	Initializes a new instance of the ModifyNodeController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	SinglePointerController.commit override.
	drawInteraction	Called to draw current state of user interaction. (Inherited from SinglePointerController.)
	move	SinglePointerController.move override.
	start	SinglePointerController.start override.
	validate	SinglePointerController.validate override.

See Also

ModifyNodeController Class
MindFusion.Diagramming Namespace

ModifyNodeController Constructor

Initializes a new instance of the ModifyNodeController class.

Namespace: MindFusion.Diagramming
File: ModifyNodeController.js

Syntax

```
JavaScript
function ModifyNodeController (node, adjustmentHandle)
```

Parameters

node
DiagramNode. The node that will be modified by this controller.

adjustmentHandle
Object. An object representing the adjustment handle used to modify the node.

See Also

ModifyNodeController Class
MindFusion.Diagramming Namespace

ModifyNodeController Methods

ModifyNodeController.commit Method

SinglePointerController.commit override.

Namespace: MindFusion.Diagramming
File: ModifyNodeController.js

Syntax

```
JavaScript
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifyNodeController Members
ModifyNodeController Class
MindFusion.Diagramming Namespace

ModifyNodeController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: ModifyNodeController.js

Syntax

JavaScript

```
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifyNodeController Members
ModifyNodeController Class
MindFusion.Diagramming Namespace

ModifyNodeController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: ModifyNodeController.js

Syntax

JavaScript

```
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifyNodeController Members
ModifyNodeController Class
MindFusion.Diagramming Namespace

ModifyNodeController.validate Method

SinglePointerController.validate override.

Namespace: MindFusion.Diagramming
File: ModifyNodeController.js

Syntax

JavaScript

```
function validate (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

Return Value

Boolean. true to accept the operation, or false otherwise.

See Also

ModifyNodeController Members
ModifyNodeController Class
MindFusion.Diagramming Namespace

ModifySelectionController Class

A controller used to modify multiple selected items in the diagram.

Syntax

JavaScript
`// class` `ModifySelectionController.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
MindFusion.Diagramming.ModifySelectionController

See Also

ModifySelectionController Members
MindFusion.Diagramming Namespace

ModifySelectionController Members

A controller used to modify multiple selected items in the diagram.

The following tables list the members exposed by the ModifySelectionController type.

Public Constructors

	Name	Description
	ModifySelectionController	Initializes a new instance of the ModifySelectionController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	SinglePointerController.commit override.

	drawInteraction	Called to draw current state of user interaction. (Inherited from SinglePointerController.)
	move	SinglePointerController.move override.
	start	SinglePointerController.start override.
	validate	SinglePointerController.validate override.

See Also

ModifySelectionController Class
MindFusion.Diagramming Namespace

ModifySelectionController Constructor

Initializes a new instance of the ModifySelectionController class.

Namespace: MindFusion.Diagramming
File: ModifySelectionController.js

Syntax

JavaScript

```
function ModifySelectionController (selection, adjustmentHandle)
```

Parameters

selection
Selection. The selection that will be modified by this controller.

adjustmentHandle
Object. An object representing the adjustment handle used to modify the selection.

See Also

ModifySelectionController Class
MindFusion.Diagramming Namespace

ModifySelectionController Methods

ModifySelectionController.commit Method

SinglePointerController.commit override.

Namespace: MindFusion.Diagramming
File: ModifySelectionController.js

Syntax

JavaScript

```
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifySelectionController Members
ModifySelectionController Class
MindFusion.Diagramming Namespace

ModifySelectionController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: ModifySelectionController.js

Syntax

JavaScript

```
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifySelectionController Members
ModifySelectionController Class
MindFusion.Diagramming Namespace

ModifySelectionController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: ModifySelectionController.js

Syntax

JavaScript

```
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

ModifySelectionController Members
ModifySelectionController Class
MindFusion.Diagramming Namespace

ModifySelectionController.validate Method

SinglePointerController.validate override.

Namespace: MindFusion.Diagramming
File: ModifySelectionController.js

Syntax

JavaScript

```
function validate (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

Return Value

Boolean. true to accept the operation, or false otherwise.

See Also

ModifySelectionController Members
ModifySelectionController Class
MindFusion.Diagramming Namespace

MouseButtonActions Enumeration

Specifies what actions can be associated with the middle and right mouse buttons.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
MouseButtonActions = {}
```

Members

	Member name	Description
	Cancel	Cancels the current drawing operation.
	Draw	Draw new items or modify existing items.
	Magnify	Magnifies the items under the mouse while the button is pressed down.
	None	Do not associate any action with the button.
	Pan	Pans the view if the mouse is dragged while the button is pressed down.

	Select	Starts a multiple selection operation.
	Zoom	Draw a zoom-lasso if the mouse is dragged with specified button pressed down.

Remarks

Members of this enumeration can be assigned to the middleButtonActions and rightButtonActions properties of DiagramView.

See Also

MindFusion.Diagramming Namespace

MouseInputDispatcher Class

Processes mouse events and dispatches them to controller objects appropriate for current context.

Syntax

JavaScript

```
// class
MouseInputDispatcher.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.MouseInputDispatcher

See Also

MouseInputDispatcher Members
MindFusion.Diagramming Namespace

MouseInputDispatcher Members

Processes mouse events and dispatches them to controller objects appropriate for current context.

The following tables list the members exposed by the MouseInputDispatcher type.

Public Constructors

	Name	Description
	MouseInputDispatcher	Overloaded.

Public Methods

	Name	Description
	onMouseDown	Called when the user presses down a mouse button.
	onMouseMove	Called when the user moves the mouse.

	onMouseUp	Called when the user releases a mouse button.

See Also

MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseInputDispatcher Constructor

Overload List

Name	Description
MouseInputDispatcher	Initializes a new instance of the MouseInputDispatcher class.
MouseInputDispatcher	Initializes a new instance of the MouseInputDispatcher class.

See Also

MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseInputDispatcher Constructor

Initializes a new instance of the MouseInputDispatcher class.

Namespace: MindFusion.Diagramming
File: MouseInputDispatcher.js

Syntax

JavaScript

```
function MouseInputDispatcher (diagram)
```

Parameters

diagram
The Diagram instance that should receive mouse input.

See Also

MouseInputDispatcher Constructor Overload List
MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseInputDispatcher Constructor

Initializes a new instance of the MouseInputDispatcher class.

Namespace: MindFusion.Diagramming
File: MouseInputDispatcher.js

Syntax

```
JavaScript
function MouseInputDispatcher (diagramView)
```

Parameters

diagramView
DiagramView. The DiagramView instance that should receive mouse input.

See Also

MouseInputDispatcher Constructor Overload List
MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseInputDispatcher Methods

MouseInputDispatcher.onMouseDown Method

Called when the user presses down a mouse button.

Namespace: MindFusion.Diagramming
File: MouseInputDispatcher.js

Syntax

```
JavaScript
function onMouseDown (position, button)
```

Parameters

position
Point. A point specified in diagram coordinates.

button
Number. 0 for left button, 1 for middle button, 2 for right button.

See Also

MouseInputDispatcher Members
MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseInputDispatcher.onMouseMove Method

Called when the user moves the mouse.

Namespace: MindFusion.Diagramming
File: MouseInputDispatcher.js

Syntax

JavaScript

```
function onMouseMove (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

MouseInputDispatcher Members
MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseInputDispatcher.onMouseUp Method

Called when the user releases a mouse button.

Namespace: MindFusion.Diagramming
File: MouseInputDispatcher.js

Syntax

JavaScript

```
function onMouseUp (position, button)
```

Parameters

position
Point. A point specified in diagram coordinates.

button
Number. 0 for left button, 1 for middle button, 2 for right button.

See Also

MouseInputDispatcher Members
MouseInputDispatcher Class
MindFusion.Diagramming Namespace

MouseWheelAction Enumeration

Specifies the type of action performed when the user rotates the mouse wheel.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
MouseWheelAction = {}
```

Members

	Member name	Description
	None	The control does not handle wheel rotation internally, but raises mouseWheel event to allow custom handling.
	Scroll	Indicates that the mouse wheel should scroll the view.
	Zoom	Indicates that the mouse wheel should zoom the view.

Remarks

Members of this enumeration can be assigned to the mouseWheelAction property of DiagramView.

See Also

MindFusion.Diagramming Namespace

MoveNodesBehavior Class

This class is used when view's behavior is set to Behavior.MoveNodes.

Syntax

JavaScript

```
// class
MoveNodesBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.MoveNodesBehavior

See Also

MoveNodesBehavior Members
MindFusion.Diagramming Namespace

MoveNodesBehavior Members

This class is used when view's behavior is set to Behavior.MoveNodes.

The following tables list the members exposed by the MoveNodesBehavior type.

Public Constructors

	Name	Description
	MoveNodesBehavior	Initializes a new instance of the MoveNodesBehavior class.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

MoveNodesBehavior Class
MindFusion.Diagramming Namespace

MoveNodesBehavior Constructor

Initializes a new instance of the MoveNodesBehavior class.

Namespace: MindFusion.Diagramming
File: MoveNodesBehavior.js

Syntax

JavaScript

```
function MoveNodesBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

MoveNodesBehavior Class
MindFusion.Diagramming Namespace

MoveNodesBehavior Methods

MoveNodesBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: MoveNodesBehavior.js

Syntax

JavaScript

```
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

MindFusion.Diagramming.SinglePointerController. An instance of SinglePointerController -derived class.

See Also

MoveNodesBehavior Members
MoveNodesBehavior Class
MindFusion.Diagramming Namespace

NodeConstraints Class

Defines specific constraints that can be applied to the size and location of nodes.

Syntax

JavaScript
`// class` `NodeConstraints.prototype = {}`

Remarks

Instances of this class can be assigned to the constraints property of nodes.

Inheritance Hierarchy

MindFusion.Diagramming.NodeConstraints

See Also

NodeConstraints Members
MindFusion.Diagramming Namespace

NodeConstraints Members

Defines specific constraints that can be applied to the size and location of nodes.

The following tables list the members exposed by the NodeConstraints type.

Public Constructors

	Name	Description
	NodeConstraints	Initializes a new instance of the **NodeConstraints** class.

Public Properties

	Name	Description

	keepRatio	Gets or sets a value indicating whether the initial width/height ratio of a node should be preserved.
	maxHeight	Gets or sets the maximal node height allowed.
	maxWidth	Gets or sets the maximal node width allowed.
	minHeight	Gets or sets the minimal node height allowed.
	minWidth	Gets or sets the minimal node width allowed.
	moveDirection	Gets or sets constraints on movement direction.

See Also

NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints Constructor

Initializes a new instance of the NodeConstraints class.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
function NodeConstraints ()
```

See Also

NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints Properties

NodeConstraints.keepRatio Property

Gets or sets a value indicating whether the initial width/height ratio of a node should be preserved.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
get keepRatio() {}
set keepRatio(value) {}
```

Property Value

Boolean

true to preserve the initial width/height ratio of a node, or **false** otherwise. The default is **false**.

Remarks

If you enable this property from a custom node class' constructor or an initialization-event handler, the node will keep an aspect ratio of 1:1. Thus you could let users draw perfect square or circle shapes by setting **keepRatio** to **true** from the initializeNode handler.

See Also

NodeConstraints Members
NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints.maxHeight Property

Gets or sets the maximal node height allowed.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
get maxHeight() {}
set maxHeight(value) {}
```

Property Value

Number

A number specifying the maximal node height.

Remarks

0 = no height constraints; any other value specifies the maximal node height allowed.

See Also

NodeConstraints Members
NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints.maxWidth Property

Gets or sets the maximal node width allowed.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
get maxWidth() {}
set maxWidth(value) {}
```

Property Value

Number

A number specifying the maximal node width.

Remarks

0 = no width constraints; any other value specifies the maximal node width allowed.

See Also

NodeConstraints Members
NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints.minHeight Property

Gets or sets the minimal node height allowed.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
get minHeight() {}
set minHeight(value) {}
```

Property Value

Number

A number specifying the minimal node height.

Remarks

0 = no height constraints; any other value specifies the minimal node height allowed.

See Also

NodeConstraints Members
NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints.minWidth Property

Gets or sets the minimal node width allowed.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
get minWidth() {}
set minWidth(value) {}
```

Property Value

Number

A number specifying the minimal node width.

Remarks

0 = no width constraints; any other value specifies the minimal node width allowed.

See Also

NodeConstraints Members
NodeConstraints Class
MindFusion.Diagramming Namespace

NodeConstraints.moveDirection Property

Gets or sets constraints on movement direction.

Namespace: MindFusion.Diagramming
File: NodeConstraints.js

Syntax

JavaScript

```
get moveDirection() {}
set moveDirection(value) {}
```

Property Value

DirectionConstraint

A member of the DirectionConstraint enumeration. The default value is **None**.

See Also

NodeConstraints Members
NodeConstraints Class
MindFusion.Diagramming Namespace

NodeEffect Class

Represents the base class of all visual effects that can be applied to the nodes in a Diagram.

Syntax

JavaScript

```
// class
NodeEffect.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.NodeEffect
MindFusion.Diagramming.AeroEffect

MindFusion.Diagramming.GlassEffect

See Also

NodeEffect Members
MindFusion.Diagramming Namespace

NodeEffect Members

Represents the base class of all visual effects that can be applied to the nodes in a Diagram.

The following tables list the members exposed by the NodeEffect type.

See Also

NodeEffect Class
MindFusion.Diagramming Namespace

NodeEventArgs Class

Contains the arguments passed to handlers of node-related events.

Syntax

JavaScript

```
// class
NodeEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ItemEventArgs
  MindFusion.Diagramming.NodeEventArgs
    MindFusion.Diagramming.ComponentEventArgs
    MindFusion.Diagramming.DrawAnchorPointEventArgs

See Also

NodeEventArgs Members
MindFusion.Diagramming Namespace
Events Class

NodeEventArgs Members

Contains the arguments passed to handlers of node-related events.

The following tables list the members exposed by the NodeEventArgs type.

Public Constructors

	Name	Description
	NodeEventArgs	Initializes a new instance of the **NodeEventArgs** class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user. (Inherited from ItemEventArgs.)
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	container	Gets the container a node has been added to or removed from.
	context	Gets the canvas rendering context. (Inherited from ItemEventArgs.)
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	hyperlink	Gets the hyperlink that was clicked inside a node.
	item	Gets the diagram item for which the event is raised. (Inherited from ItemEventArgs.)
	label	Gets the node label for which the event is raised.
	mouseButton	Gets which mouse button has been pressed. (Inherited from ItemEventArgs.)
	mousePosition	Gets the current mouse pointer position. (Inherited from ItemEventArgs.)
	newText	Gets the text that has just been entered by the user. (Inherited from ItemEventArgs.)
	node	Gets the node for which the event is raised.
	oldText	Gets the text that had been displayed in the node before the user edited it. (Inherited from ItemEventArgs.)

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the creation or modification of the node. (Inherited from ItemEventArgs.)

See Also

NodeEventArgs Class
MindFusion.Diagramming Namespace

NodeEventArgs Constructor

Initializes a new instance of the NodeEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
function NodeEventArgs (args)
```

Parameters

args

See Also

NodeEventArgs Class
MindFusion.Diagramming Namespace

NodeEventArgs Properties

NodeEventArgs.container Property

Gets the container a node has been added to or removed from.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get container() {}
```

Property Value

ContainerNode

A ContainerNode instance.

See Also

NodeEventArgs Members
NodeEventArgs Class
MindFusion.Diagramming Namespace

NodeEventArgs.hyperlink Property

Gets the hyperlink that was clicked inside a node.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get hyperlink() {}
```

Property Value

String

A string containing clicked hyperlink.

Remarks

This value is valid when handling hyperlinkClicked event.

See Also

NodeEventArgs Members
NodeEventArgs Class
MindFusion.Diagramming Namespace

NodeEventArgs.label Property

Gets the node label for which the event is raised.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get label() {}
set label(value) {}
```

Property Value

NodeLabel

A NodeLabel instance.

See Also

NodeEventArgs Members
NodeEventArgs Class
MindFusion.Diagramming Namespace

NodeEventArgs.node Property

Gets the node for which the event is raised.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get node() {}
```

Property Value

DiagramNode

An instance of the DiagramNode class.

See Also

NodeLabel Class

Represents a supplementary label displayed by a DiagramNode. New labels can be added to a node by calling its addLabel method.

Syntax

```
JavaScript
// class
NodeLabel.prototype = {}
```

Remarks

The position of a label is determined by where it is pinned relative to node's bounding box, and offset from the pin point, as set using setCornerPosition, setEdgePosition and setCenterPosition methods. In addition, the horizontalAlign and verticalAlign properties specify whether the label is centered on its pin point or aligned to it.

You can let users arrange labels manually by setting the behavior property of DiagramView to MoveLabels.

Inheritance Hierarchy

MindFusion.Diagramming.ItemLabel
MindFusion.Diagramming.NodeLabel

See Also

NodeLabel Members
MindFusion.Diagramming Namespace

NodeLabel Members

Represents a supplementary label displayed by a DiagramNode. New labels can be added to a node by calling its addLabel method.

The following tables list the members exposed by the NodeLabel type.

Public Constructors

	Name	Description
	NodeLabel	Initializes a new instance of the **NodeLabel** class.

Public Properties

	Name	Description
	brush	Gets or sets the brush used to draw the label's background. (Inherited from ItemLabel.)

	contentPadding	Gets or sets the distance between image and text when both are specified. (Inherited from ItemLabel.)
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled. (Inherited from ItemLabel.)
	enableWrap	Gets or sets a value indicating whether text wrapping is enabled. (Inherited from ItemLabel.)
	font	Gets or sets the font used to render label's text. (Inherited from ItemLabel.)
	horizontalAlign	Gets or sets the label's horizontal alignment. (Inherited from ItemLabel.)
	horizontalOffset	Gets or sets label's horizontal offset from its pin point. (Inherited from ItemLabel.)
	image	Gets or sets the image displayed by this label. (Inherited from ItemLabel.)
	imageAlign	Gets or sets the alignment of label's image relative to text. (Inherited from ItemLabel.)
	imageContent	Gets or sets a Base64-encoded image to display inside this label. (Inherited from ItemLabel.)
	imageLocation	Gets or sets URL of the image to display inside this label. (Inherited from ItemLabel.)
	imageSize	Gets or sets a size that overrides the native image size. (Inherited from ItemLabel.)
	locked	Gets or sets a value indicating whether users are allowed to edit this label. (Inherited from ItemLabel.)
	margin	Gets or sets the label's margin. (Inherited from ItemLabel.)
	maxWidth	Gets or sets the maximum width of this label. (Inherited from ItemLabel.)
	text	Gets or sets the label's text. (Inherited from ItemLabel.)
	textColor	Gets or sets the text color. (Inherited from ItemLabel.)
	verticalAlign	Gets or sets the label's vertical alignment. (Inherited from ItemLabel.)
	verticalOffset	Gets or sets label's vertical offset from its pin point. (Inherited from ItemLabel.)

Public Methods

	Name	Description
	setCenterPosition	Positions this label relatively to the node center.
	setCornerPosition	Positions this label relatively to the specified corner point.
	setEdgePosition	Positions this label relatively to the specified node edge.

See Also

NodeLabel Class
MindFusion.Diagramming Namespace

NodeLabel Constructor

Initializes a new instance of the NodeLabel class.

Namespace: MindFusion.Diagramming
File: NodeLabel.js

Syntax

JavaScript

```
function NodeLabel (node, text)
```

Parameters

node
DiagramNode. The DiagramNode that displays the label.

text
String. The label's text.

See Also

NodeLabel Class
MindFusion.Diagramming Namespace

NodeLabel Methods

NodeLabel.setCenterPosition Method

Positions this label relatively to the node center.

Namespace: MindFusion.Diagramming
File: NodeLabel.js

Syntax

JavaScript

```
function setCenterPosition (dx, dy)
```

Parameters

dx
Number. Horizontal offset from the center point.

dy
Number. Vertical offset from the center point.

Remarks

This method sets the node's center as label's pin point and assigns specified offsets to horizontalOffset and verticalOffset properties.

See Also

NodeLabel Members
NodeLabel Class
MindFusion.Diagramming Namespace

NodeLabel.setCornerPosition Method

Positions this label relatively to the specified corner point.

Namespace: MindFusion.Diagramming
File: NodeLabel.js

Syntax

JavaScript

```
function setCornerPosition (cornerPoint, dx, dy)
```

Parameters

cornerPoint
Number. The corner point index.

dx
Number. Horizontal offset from the corner point.

dy
Number. Vertical offset from the corner point.

Remarks

This method sets node's corner as label's pin point and assigns specified offsets to horizontalOffset and verticalOffset properties. The *cornerPoint* index starts counting from top-left corner in clockwise direction:

- 0 desigates top-left corner;
- 1 desigates top-right corner;
- 2 desigates bottom-right corner;
- 3 desigates bottom-left corner;

See Also

NodeLabel Members
NodeLabel Class
MindFusion.Diagramming Namespace

NodeLabel.setEdgePosition Method

Positions this label relatively to the specified node edge.

Namespace: MindFusion.Diagramming
File: NodeLabel.js

Syntax

JavaScript

```
function setEdgePosition (edge, dx, dy)
```

Parameters

edge
Number. The edge index.

dx
Number. Horizontal offset from the edge center.

dy
Number. Vertical offset from the edge center.

Remarks

This method sets the middle point of node's edge as label's pin point and assigns specified offsets to horizontalOffset and verticalOffset properties. The *edge* index starts counting from top side in clockwise direction:

- 0 desigates center of top edge;
- 1 desigates center of right edge;
- 2 desigates center of bottom edge;
- 3 desigates center of left edge;

See Also

NodeLabel Members
NodeLabel Class
MindFusion.Diagramming Namespace

NodeListView Class

A list view control that hosts DiagramNodes and supports drag-and-drop to DiagramView.

Syntax

JavaScript

```
// class
NodeListView.prototype = {}
```

Remarks

Obsolete

This component has been superseded by ItemListView, which can also display DiagramLink objects and supports multi-column layouts.

Call the addNode method to add a node to the view.

Drag-and-drop operations create a clone of the dragged node by calling its clone method. You can override clone to copy property values of custom node subclasses.

Inheritance Hierarchy

MindFusion.Controls.CanvasControl
  MindFusion.Diagramming.ListViewBase
    MindFusion.Diagramming.NodeListView

See Also

NodeListView Members

A list view control that hosts DiagramNodes and supports drag-and-drop to DiagramView.

The following tables list the members exposed by the NodeListView type.

Public Constructors

	Name	Description
	NodeListView	Creates an instance of NodeListView

Public Properties

	Name	Description
	backColor	Gets or sets the background color of the control.
	defaultNodeSize	Gets or sets the default size of nodes created after dragging from this list view.
	draggedItem	Gets the currently dragged item. (Inherited from ListViewBase.)
	draggedNode	Gets the currently dragged node from this list view.
	element	Returns a reference to the control's DOM element. (Inherited from Control.)
	enabled	Gets or sets whether mouse events are enabled. (Inherited from Control.)
	iconSize	Gets or sets the size of nodes displayed in this list view.
	items	Gets the items contained in this list view. (Inherited from ListViewBase.)
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	measureUnit	Gets the unit of measure used for logical coordinates. (Inherited from CanvasControl.)
	newInstanceSize	Gets or sets the default size of items created by drag-and-drop from this list view. (Inherited from ListViewBase.)
	nodes	Gets the nodes contained in this NodeListView.
	orientation	Gets or sets the orientation of the view layout.
	padding	Gets or sets the padding space between the NodeListView control and its contents.
	shapeLibraryLocation	Gets the location of a shape library file containing custom shape definitions.It can be set to a full URL of the file, or a relative file path.

	textColor	Gets or sets the color used to paint text in the NodeListView control.
	tooltipDelay	Gets or sets the delay before a tooltip is shown.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
	addItem	Adds an item to the list view. (Inherited from ListViewBase.)
	addNode	Adds a node to the NodeListView.
	clearAll	Removes all items from the **NodeListView**.
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)
	clientToDoc	Transforms a point from client to document coordinates. (Inherited from CanvasControl.)
S	create	Creates and initializes a new **NodeListView** from the specified element. This method is static and can be called without creating an instance of the class.
	docToClient	Transforms a point from document to client coordinates. (Inherited from CanvasControl.)
S	find	Returns the specified **NodeListView** object. This member is static and can be invoked without creating an instance of the class.
S	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	getItemAt	Gets the item at the specified location. (Inherited from ListViewBase.)
	getNodeAt	Gets the item at the specified location.
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data. (Inherited from Control.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
	removeItem	Removes an item from the list view. (Inherited from ListViewBase.)
	removeNode	Removes a node from the list view.
	selectItem	Selects the specified item.
	selectItem	Selects the specified item. (Inherited from ListViewBase.)

	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description
	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
	controlUnloading	Raised when the control is unloading. (Inherited from Control.)
	linkSelected	Raised when a link is selected, either programmatically or by the user. (Inherited from ListViewBase.)
	nodeSelected	Raised when a node is selected, either programmatically or by the user.
	shapeLibraryLoaded	Raised when a ShapeLibrary is loaded from **shapeLibraryLocation**.

See Also

NodeListView Class
MindFusion.Diagramming Namespace

NodeListView Constructor

Creates an instance of NodeListView

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
function NodeListView (element)
```

Parameters

element
HTMLCanvasElement. The Canvas DOM Element this NodeListView is associated with.

See Also

NodeListView Class
MindFusion.Diagramming Namespace

NodeListView Methods

NodeListView.addNode Method

Adds a node to the NodeListView.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
function addNode (node, [caption])
```

Parameters

node
DiagramNode. The node to add.

caption
Optional.
String. The caption to display for the node.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace
nodes Property
removeNode Method
clearAll Method

NodeListView.clearAll Method

Removes all items from the NodeListView.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
function clearAll ()
```

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace
addNode Method
removeNode Method

NodeListView.create Method

Creates and initializes a new NodeListView from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
function create (element)
```

Parameters

element
HTMLCanvasElement. The DOM element that the NodeListView should be attached to.

Return Value

A NodeListView object that represents the newly created list view.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.find Method

Returns the specified NodeListView object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
function find (id)
```

Parameters

id
String. The id of the NodeListView's element.

Return Value

A NodeListView object that contains the list view requested by id, if found, or **null** otherwise.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.getNodeAt Method

Gets the item at the specified location.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
function getNodeAt (point)
```

Parameters

point
Point. The location where to look.

Return Value

DiagramNode. The item found.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.removeNode Method

Removes a node from the list view.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
function removeNode (node)
```

Parameters

node
DiagramNode. The node to remove.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace
addNode Method
clearAll Method

NodeListView.selectItem Method

Selects the specified item.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
function selectItem (item)
```

Parameters

item
DiagramNode. The item to select.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView Properties

NodeListView.backColor Property

Gets or sets the background color of the control.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
get backColor() {}
set backColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.defaultNodeSize Property

Gets or sets the default size of nodes created after dragging from this list view.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
```

```
get defaultNodeSize() {}
set defaultNodeSize(value) {}
```

Property Value

Size

An instance of the Size class specifying node size.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.draggedNode Property

Gets the currently dragged node from this list view.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
get draggedNode() {}
```

Property Value

DiagramNode

The dragged diagram node.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.iconSize Property

Gets or sets the size of nodes displayed in this list view.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
get iconSize() {}
```

Property Value

Size

An instance of the Size class specifying node size.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.nodes Property

Gets the nodes contained in this NodeListView.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
get nodes() {}
```

Property Value

Array

An array of DiagramNode objects to show in the list view.

Remarks

You must call addNode to add nodes to the list view, instead of modifying this array directly.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.orientation Property

Gets or sets the orientation of the view layout.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
get orientation() {}
set orientation(value) {}
```

Property Value

Orientation

A member of the Orientation enumeration.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.padding Property

Gets or sets the padding space between the NodeListView control and its contents.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
get padding() {}
set padding(value) {}
```

Property Value

Number

A number that specifies the padding space size.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.shapeLibraryLocation Property

Gets the location of a shape library file containing custom shape definitions.It can be set to a full URL of the file, or a relative file path.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

JavaScript

```
get shapeLibraryLocation() {}
set shapeLibraryLocation(value) {}
```

Property Value

String

URL of a shape library file containing custom shape definitions.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.textColor Property

Gets or sets the color used to paint text in the NodeListView control.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.tooltipDelay Property

Gets or sets the delay before a tooltip is shown.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
get tooltipDelay() {}
set tooltipDelay(value) {}
```

Property Value

Number

The tooltip delay in milliseconds.

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView Events

NodeListView.nodeSelected Event

Raised when a node is selected, either programmatically or by the user.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
nodeList.nodeSelected.addEventListener(onNodeSelected);
```

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

NodeListView.shapeLibraryLoaded Event

Raised when a ShapeLibrary is loaded from shapeLibraryLocation.

Namespace: MindFusion.Diagramming
File: NodeListView.js

Syntax

```
JavaScript
nodeList.shapeLibraryLoaded.addEventListener(onShapeLibraryLoaded);
```

See Also

NodeListView Members
NodeListView Class
MindFusion.Diagramming Namespace

Orientation Enumeration

Specifies the orientation of user interface elements.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
Orientation = {}
```

Members

	Member name	Description
	Auto	Select orientation automatically.
	Horizontal	Specifies horizontal orientation.

	Vertical	Specifies vertical orientation.

See Also

MindFusion.Diagramming Namespace

Overview Class

The **Overview** control provides a scaled-down view of Diagram contents and allows scrolling and zooming a DiagramView.

Syntax

JavaScript

```
// class
Overview.prototype = {}
```

Remarks

To specify which DiagramView instance is controlled by the overview, set the diagramView property. Set the scaleMode property to specify if overview should zoom-to-fit the whole diagram or display it at a fixed scale.

Inheritance Hierarchy

MindFusion.Controls.CanvasControl
MindFusion.Diagramming.Overview

See Also

Overview Members
MindFusion.Diagramming Namespace

Overview Members

The Overview control provides a scaled-down view of Diagram contents and allows scrolling and zooming a DiagramView.

The following tables list the members exposed by the **Overview** type.

Public Constructors

	Name	Description
	Overview	Initializes a new instance of the Overview class.

Public Properties

	Name	Description

	allowZoom	Gets or sets a value indicating whether users are allowed to zoom the diagramby resizing the overview's viewport tracker rectangle.
	autoScrollAmount	Gets or sets the amount by which to auto-scroll the view when the mouse leaves the view boundaries while drawing.
	backColor	Gets or sets the background color of the overview control.
	diagramView	Gets or sets the **DiagramView** shown by this **Overview**.
	dimColor	Gets or sets a color used to blend with the part of the diagram that is not visible in target DiagramView.
	element	Returns a reference to the control's DOM element. (Inherited from Control.)
	enabled	Gets or sets whether mouse events are enabled. (Inherited from Control.)
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	measureUnit	Gets the unit of measure used for logical coordinates. (Inherited from CanvasControl.)
	scaleFactor	Gets or sets a fixed scale factor to apply when FitAll is disabled.
	scaleMode	Gets or sets a value indicating whether to fit the whole diagram inside the overview control.
	scrollX	Gets the overview's horizontal scroll position.
	scrollY	Gets the overview's vertical scroll position.
	viewport	Returns the visible portion of the diagram area.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)
	clientToDoc	Transforms a point from client to document coordinates. (Inherited from CanvasControl.)
S	create	Creates and initializes a new **Overview** from the specified element. This method is static and can be called without creating an instance of the class.
	docToClient	Transforms a point from document to client coordinates. (Inherited from CanvasControl.)
S	find	Returns the specified **Overview** object. This member is static and can be invoked without creating an instance of

		the class.
	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Control's parent html form to flush postback data. (Inherited from Control.)
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description
	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
	controlUnloading	Raised when the control is unloading. (Inherited from Control.)

See Also

Overview Class
MindFusion.Diagramming Namespace

Overview Constructor

Initializes a new instance of the Overview class.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript
`function Overview (element)`

Parameters

element
HTMLCanvasElement. The Canvas DOM Element this Overview is associated with.

See Also

Overview Class
MindFusion.Diagramming Namespace

Overview Methods

Overview.create Method

Creates and initializes a new Overview from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript

```
function create (element)
```

Parameters

element
HTMLCanvasElement. The DOM element that the overview should be attached to.

Return Value

An Overview object that represents the newly created overview.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.find Method

Returns the specified Overview object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript

```
function find (id)
```

Parameters

id
String. The id of the Overview's element.

Return Value

An **Overview** object with the specified id, if found, or **null** otherwise.

See Also

Overview Members
Overview Class

Overview Properties

Overview.allowZoom Property

Gets or sets a value indicating whether users are allowed to zoom the diagramby resizing the overview's viewport tracker rectangle.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript

```
get allowZoom() {}
set allowZoom(value) {}
```

Property Value

Boolean

true to let users zoom by resizing the tracker rectangle, or **false** otherwise.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.autoScrollAmount Property

Gets or sets the amount by which to auto-scroll the view when the mouse leaves the view boundaries while drawing.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript

```
get autoScrollAmount() {}
set autoScrollAmount(value) {}
```

Property Value

Number

A number specifying the amount by which to auto-scroll.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.backColor Property

Gets or sets the background color of the overview control.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

```
JavaScript
get backColor() {}
set backColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.diagramView Property

Gets or sets the DiagramView shown by this Overview.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

```
JavaScript
get diagramView() {}
set diagramView(value) {}
```

Property Value

DiagramView

The DiagramView shown by the overview.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.dimColor Property

Gets or sets a color used to blend with the part of the diagram that is not visible in target DiagramView.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

```
JavaScript
get dimColor() {}
set dimColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.scaleFactor Property

Gets or sets a fixed scale factor to apply when FitAll is disabled.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

```
JavaScript
get scaleFactor() {}
set scaleFactor(value) {}
```

Property Value

Number

A number specifying the scale as percentage.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.scaleMode Property

Gets or sets a value indicating whether to fit the whole diagram inside the overview control.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

```
JavaScript
get scaleMode() {}
set scaleMode(value) {}
```

Property Value

ScaleMode

A member of the ScaleMode enumeration. The default value is **FitAll**.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.scrollX Property

Gets the overview's horizontal scroll position.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript

```
get scrollX() {}
set scrollX(value) {}
```

Property Value

Number

A number specifying the horizontal scroll position in current unit of measure.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.scrollY Property

Gets the overview's vertical scroll position.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

JavaScript

```
get scrollY() {}
set scrollY(value) {}
```

Property Value

Number

A number specifying the vertical scroll position in current unit of measure.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview.viewport Property

Returns the visible portion of the diagram area.

Namespace: MindFusion.Diagramming
File: Overview.js

Syntax

```
JavaScript
get viewport() {}
set viewport(value) {}
```

Property Value

Rect

A Rect instance containing the viewport coordinates.

See Also

Overview Members
Overview Class
MindFusion.Diagramming Namespace

Overview Events

PageContainerCommand Class

Implements page navigation of PagedContainerNode.

Syntax

```
JavaScript
// class
PageContainerCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.PageContainerCommand

See Also

PageContainerCommand Members
MindFusion.Diagramming Namespace

PageContainerCommand Members

Implements page navigation of PagedContainerNode.

The following tables list the members exposed by the PageContainerCommand type.

Public Constructors

	Name	Description
	PageContainerCommand	Initializes a new instance of the PageContainerCommand class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

PageContainerCommand Class
MindFusion.Diagramming Namespace

PageContainerCommand Constructor

Initializes a new instance of the PageContainerCommand class.

Namespace: MindFusion.Diagramming
File: PageContainerCommand.js

Syntax

JavaScript

```
function PageContainerCommand (container)
```

Parameters

container
PagedContainerNode. The PagedContainerNode modified by this command.

See Also

PageContainerCommand Class
MindFusion.Diagramming Namespace

PageContainerCommand Methods

PageContainerCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: PageContainerCommand.js

Syntax

```
JavaScript
function execute ()
```

See Also

PageContainerCommand Members
PageContainerCommand Class
MindFusion.Diagramming Namespace

PageContainerCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: PageContainerCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

PageContainerCommand Members
PageContainerCommand Class
MindFusion.Diagramming Namespace

PageContainerCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: PageContainerCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

PageContainerCommand Members
PageContainerCommand Class
MindFusion.Diagramming Namespace

PagedContainerNode Class

A container that can arrange its children in pages.

Syntax

```
JavaScript
// class
PagedContainerNode.prototype = {}
```

Remarks

PagedContainerNode is a ContainerNode subclass that organizes its child nodes into a collection of pages, represented by ContainerPage objects. This allows for the creation of more orgainzed diagrams where nodes can be grouped into logical views within a single container. Only the nodes belonging to the currently active page, specified by currentPage property, are rendered and participate in user interactions.

Navigation between pages is handled by scroll arrows in the caption bar, allowing users to cycle through the pages sequentially. For faster access, users can also double-click the container's caption to open an in-place combo box listing available page titles, enabling direct navigation to any page. The node's text automatically displays the title of newly activated **ContainerPage**.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.CaptionedNode
      MindFusion.Diagramming.ContainerNode
        MindFusion.Diagramming.PagedContainerNode

See Also

PagedContainerNode Members
MindFusion.Diagramming Namespace

PagedContainerNode Members

A container that can arrange its children in pages.

The following tables list the members exposed by the PagedContainerNode type.

Public Constructors

	Name	Description
	PagedContainerNode	Initializes a new instance of the PagedContainerNode class.

Public Properties

	Name	Description
	allowAddChildren	Gets or sets whether users are allowed to add child nodes to the container. (Inherited from ContainerNode.)
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	allowRemoveChildren	Gets or sets whether users are allowed to remove child nodes from the container. (Inherited from ContainerNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	autoDeleteChildren	Gets or sets whether whether deleting this container should also remove its child items from the diagram. (Inherited from ContainerNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	captionBackBrush	Gets or sets an object specifying how to fill the caption bar. (Inherited from ContainerNode.)
	captionFont	Gets or sets the font used to render the node's caption text. (Inherited from CaptionedNode.)
	captionHeight	Gets or sets the height of the node's caption bar. (Inherited from ContainerNode.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	children	Gets the child nodes of this container. (Inherited from ContainerNode.)
	clipChildren	Gets or sets whether child items should be clipped by container boundaries. (Inherited from ContainerNode.)
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent **ContainerNode** containing this node. (Inherited from DiagramItem.)
	contentPath	Creates a path corresponding to the content area rectangle, excluding caption bar. (Inherited from ContainerNode.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)

	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	currentPage	Gets or sets the current page.
	dividerStroke	Gets or sets the color used to stroke the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	dividerStrokeDashArray	Gets or sets an array of values specifying the dash pattern of the node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from CaptionedNode.)
	dividerStrokeDashStyle	Gets or sets the line dash pattern applied when stroking node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeThickness	Gets or sets the width of the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveDividerStrokeThickness	Gets the effective line thickness that should be applied when stroking this node's divider line. (Inherited from CaptionedNode.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)

	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	enableStyledText	Gets or sets a value indicating whether styled caption text rendering is enabled. (Inherited from ContainerNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	foldable	Gets or sets a value indicating whether users are allowed to fold the container. (Inherited from ContainerNode.)
	folded	Gets or sets a value indicating whether the container is folded. (Inherited from ContainerNode.)
	foldedSize	Gets or sets the size of the container in folded state. (Inherited from ContainerNode.)
	foldIconSize	Gets or sets a value indicating the size of the fold/unfold icon. (Inherited from ContainerNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)

	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	pages	Gets the pages of this container.
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the shape of the node's outline. (Inherited from ContainerNode.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)

	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	unfoldedSize	Gets the size of the container in unfolded state. (Inherited from ContainerNode.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	add	Adds a node to this container. (Inherited from ContainerNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)

	addPage	Adds a page to the container.
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	arrange	Arranges the container's children with the given layout. (Inherited from ContainerNode.)
	arrangeAnimated	Arranges the container's children with the given layout, and animates items moving to their new positions. (Inherited from ContainerNode.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	Checks if the **ContainerNode** contains the specified point. (Inherited from ContainerNode.)
	containsRecursively	Determines whether the specified node is contained within this container, either directly or inside child containers. (Inherited from ContainerNode.)
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	drawChildren	ContainerNode.drawChildren override.
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	fromJson	DiagramNode.fromJson override.
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getEditRect	DiagramNode.getEditRect override.
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets an array of points that define the outline of this node shape. (Inherited from ContainerNode.)
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)

getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
hitTest	Returns the diagram item that contains the specified point. (Inherited from ContainerNode.)
hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
insertPage	Inserts a page at the specified index.
invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
loadFromXml	DiagramNode.loadFromXml override.
loadFromXml	DiagramNode.LoadFromXml override. Loads the node's content from an XML element. (Inherited from ContainerNode.)
movePage	Moves a page to a new position in the page list.
onRemove	DiagramItem.onRemove override. (Inherited from ContainerNode.)
raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
remove	Removes a node from this container. (Inherited from ContainerNode.)
removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
removePage	Removes a page from the container.
resizeToFitChildren	Resizes the container making it big enough to contain its child nodes. (Inherited from ContainerNode.)
resizeToFitText	Makes the container node caption big enough to display its text without clipping. (Inherited from ContainerNode.)
restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
restoreState	DiagramNode.restoreState override. (Inherited from ContainerNode.)
saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
saveState	DiagramNode.saveState override. (Inherited from ContainerNode.)
saveToXml	DiagramNode.saveToXml override.

	saveToXml	Saves the node's content into an XML element. (Inherited from ContainerNode.)
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)
	visitNodes	Visits the diagram nodes in reverse Z order. (Inherited from ContainerNode.)

See Also

PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode Constructor

Initializes a new instance of the PagedContainerNode class.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript

```
function PagedContainerNode ([parent, [pageTitles]])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the node properties.

pageTitles
Optional.
An array of title strings for the initial pages.

See Also

PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode Methods

PagedContainerNode.addPage Method

Adds a page to the container.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

```
JavaScript
function addPage (page)
```

Parameters

page
ContainerPage. The page to add.

Remarks

Container's current pages are accessible via pages property.

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace
removePage Method

PagedContainerNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.drawChildren Method

ContainerNode.drawChildren override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript

```
function drawChildren ()
```

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.getEditRect Method

DiagramNode.getEditRect override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript

```
function getEditRect ()
```

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.insertPage Method

Inserts a page at the specified index.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript
`function insertPage (index, page)`

Parameters

index
Number. The zero-based index at which the page should be inserted.

page
ContainerPage. The page to insert.

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.loadFromXml Method

DiagramNode.loadFromXml override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript
`function loadFromXml ()`

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.movePage Method

Moves a page to a new position in the page list.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript
`function movePage (newIndex, page)`

Parameters

newIndex
Number. The new index of the page.

page
ContainerPage. The page to move.

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.removePage Method

Removes a page from the container.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript

```
function removePage (page)
```

Parameters

page
ContainerPage. The page to remove.

Remarks

Container's current pages are accessible via pages property.

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace
addPage Method

PagedContainerNode.saveToXml Method

DiagramNode.saveToXml override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

JavaScript

```
function saveToXml ()
```

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode Properties

PagedContainerNode.currentPage Property

Gets or sets the current page.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

```
JavaScript
get currentPage() {}
set currentPage(value) {}
```

Property Value

ContainerPage

A ContainerPage reference.

Remarks

PagedContainerNode displays only child items associated with this page. Users can change the current page by clicking on arrow icons in the caption bar.

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PagedContainerNode.pages Property

Gets the pages of this container.

Namespace: MindFusion.Diagramming
File: PagedContainerNode.js

Syntax

```
JavaScript
get pages() {}
```

Property Value

Array

An array of ContainerPage objects.

Remarks

Call addPage and removePage methods to modify this collection.

See Also

PagedContainerNode Members
PagedContainerNode Class
MindFusion.Diagramming Namespace

PanBehavior Class

Pans the diagram when the users drags the mouse.

Syntax

```
JavaScript
// class
PanBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.PanBehavior

See Also

PanBehavior Members
MindFusion.Diagramming Namespace

PanBehavior Members

Pans the diagram when the users drags the mouse.

The following tables list the members exposed by the PanBehavior type.

Public Constructors

	Name	Description
	PanBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

PanBehavior Class
MindFusion.Diagramming Namespace

PanBehavior Constructor

Overload List

Name	Description
PanBehavior	Initializes a new instance of the PanBehavior class.
PanBehavior	Initializes a new instance of the PanBehavior class.

See Also

PanBehavior Class
MindFusion.Diagramming Namespace

PanBehavior Constructor

Initializes a new instance of the PanBehavior class.

Namespace: MindFusion.Diagramming
File: PanBehavior.js

Syntax

JavaScript
`function PanBehavior (parent)`

Parameters

parent
The Diagram this behavior will be associated with.

See Also

PanBehavior Constructor Overload List
PanBehavior Class
MindFusion.Diagramming Namespace

PanBehavior Constructor

Initializes a new instance of the PanBehavior class.

Namespace: MindFusion.Diagramming
File: PanBehavior.js

Syntax

JavaScript

```
function PanBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

PanBehavior Constructor Overload List
PanBehavior Class
MindFusion.Diagramming Namespace

PanBehavior Methods

PanBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: PanBehavior.js

Syntax

JavaScript

```
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

PanBehavior Members
PanBehavior Class
MindFusion.Diagramming Namespace

PanController Class

A controller used to pan the DiagramView.

Syntax

JavaScript

```
// class
PanController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
  MindFusion.Diagramming.PanController

See Also

PanController Members
MindFusion.Diagramming Namespace

PanController Members

A controller used to pan the DiagramView.

The following tables list the members exposed by the PanController type.

Public Constructors

	Name	Description
	PanController	Overloaded.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation. (Inherited from SinglePointerController.)
	commit	Called when the user stops drawing or modifying a diagram element, accepting the operation. (Inherited from SinglePointerController.)
	drawInteraction	Called to draw current state of user interaction. (Inherited from SinglePointerController.)
	move	SinglePointerController.move override.

	start	SinglePointerController.start override.
	validate	Called to validate the operation at current pointer position. (Inherited from SinglePointerController.)

See Also

PanController Class
MindFusion.Diagramming Namespace

PanController Constructor

Overload List

Name	Description
PanController	Initializes a new instance of the PanController class.
PanController	Initializes a new instance of the PanController class.

See Also

PanController Class
MindFusion.Diagramming Namespace

PanController Constructor

Initializes a new instance of the PanController class.

Namespace: MindFusion.Diagramming
File: PanController.js

Syntax

JavaScript
`function PanController (diagram)`

Parameters

diagram
The diagram that will get scrolled by this controller.

See Also

PanController Constructor Overload List
PanController Class
MindFusion.Diagramming Namespace

PanController Constructor

Initializes a new instance of the PanController class.

Namespace: MindFusion.Diagramming
File: PanController.js

Syntax

```
JavaScript
function PanController (diagramView)
```

Parameters

diagramView
DiagramView. The diagramView that will get scrolled by this controller.

See Also

PanController Constructor Overload List
PanController Class
MindFusion.Diagramming Namespace

PanController Methods

PanController.move Method

SinglePointerController.move override.

Namespace: MindFusion.Diagramming
File: PanController.js

Syntax

```
JavaScript
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

PanController Members
PanController Class
MindFusion.Diagramming Namespace

PanController.start Method

SinglePointerController.start override.

Namespace: MindFusion.Diagramming
File: PanController.js

Syntax

```
```

JavaScript

```
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

PanController Members
PanController Class
MindFusion.Diagramming Namespace

Panel Class

Represents layout containers in CompositeNode visual tree.

Syntax

JavaScript

```
// class
Panel.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Container
  MindFusion.Diagramming.Panel
    MindFusion.Diagramming.GridPanel
    MindFusion.Diagramming.SimplePanel
    MindFusion.Diagramming.StackPanel

See Also

Panel Members
MindFusion.Diagramming Namespace
Composite Nodes

Panel Members

Represents layout containers in CompositeNode visual tree.

The following tables list the members exposed by the Panel type.

Public Constructors

	Name	Description
	Panel	Initializes a new instance of the Panel class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	getComponents	Gets the child components of this panel.

See Also

Panel Class
MindFusion.Diagramming Namespace

Panel Constructor

Initializes a new instance of the Panel class.

Namespace: MindFusion.Diagramming
File: Panel.js

Syntax

JavaScript
`function Panel ()`

See Also

Panel Class
MindFusion.Diagramming Namespace

Panel Methods

Panel.getComponents Method

Gets the child components of this panel.

Namespace: MindFusion.Diagramming
File: Panel.js

Syntax

JavaScript
`function getComponents ()`

Return Value

Array. An array containing the child components of this panel.

See Also

Panel Members
Panel Class
MindFusion.Diagramming Namespace

Path Class

Represents a sequence of connected DiagramNode and DiagramLink elements.

Syntax

JavaScript
`// class` `Path.prototype = {}`

Remarks

Path instances are returned by methods of the PathFinder class.

Inheritance Hierarchy

MindFusion.Diagramming.Path

See Also

Path Members
MindFusion.Diagramming Namespace

Path Members

Represents a sequence of connected DiagramNode and DiagramLink elements.

The following tables list the members exposed by the Path type.

Public Constructors

	Name	Description
	Path	Initializes a new instance of the **Path** class.

Public Fields

	Name	Description

	items	An array of all DiagramItem elements participating in the path.
	links	An array of all **DiagramLink** elements participating in the path.
	nodes	An array of all **DiagramNode** elements participating in the path.

Public Methods

	Name	Description
	getLength	Gets the length of the **Path** expressed as the number of **links** participating in it.
	getWeight	Calculates the weight of this **Path** by accumulating the Weight of all participating diagram elements.

See Also

Path Class
MindFusion.Diagramming Namespace

Path Fields

Path.items Field

An array of all DiagramItem elements participating in the path.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

JavaScript
`var items`

Remarks

The collection contains consecutive DiagramNode and DiagramLink objects.

See Also

Path Members
Path Class
MindFusion.Diagramming Namespace

Path.links Field

An array of all DiagramLink elements participating in the path.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
var links
```

Remarks

Use nodes to get all DiagramNode elements in the Path.

See Also

Path Members
Path Class
MindFusion.Diagramming Namespace

Path.nodes Field

An array of all DiagramNode elements participating in the path.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
var nodes
```

Remarks

Use links to get all DiagramLink elements in the Path.

See Also

Path Members
Path Class
MindFusion.Diagramming Namespace

Path Constructor

Initializes a new instance of the Path class.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function Path ()
```

See Also

Path Methods

Path.getLength Method

Gets the length of the Path expressed as the number of links participating in it.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function getLength ()
```

Return Value

Number. An integer value that represents the number of edges (links) in the path.

See Also

Path Members
Path Class
MindFusion.Diagramming Namespace

Path.getWeight Method

Calculates the weight of this Path by accumulating the Weight of all participating diagram elements.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function getWeight ([incNodes, [incLinks]])
```

Parameters

incNodes
Optional.
A boolean value specifying whether the Weight of DiagramNode elements should be considered.

incLinks
Optional.
A boolean value specifying whether the Weight of DiagramLink elements should be considered.

Return Value

A number that represents the total weight of the elements in this Path according to the *incNodes* and *incLinks* parameters.

See Also

PathFinder Class

Provides methods that help you find paths and cycles in a graph.

Syntax

```
// class
PathFinder.prototype = {}
```

Remarks

The direction of links is taken into account when looking for paths or cycles.

A path-finder object caches graph information between method calls in order to achieve better performance when path and cycle -finding methods are called on the same diagram more than once. If the diagram structure changes, you must create a new PathFinder instance in order to use the new structure of diagram's graph.

Inheritance Hierarchy

MindFusion.Diagramming.PathFinder

See Also

PathFinder Members
MindFusion.Diagramming Namespace

PathFinder Members

Provides methods that help you find paths and cycles in a graph.

The following tables list the members exposed by the PathFinder type.

Public Constructors

	Name	Description
	PathFinder	Initializes a new instance of the PathFinder class.

Public Methods

	Name	Description
	findAllCycles	Finds all cycles in the underlying diagram.
	findAllPaths	Finds and returns all paths that exist between two DiagramNode objects.

	findCycle	Detects whether the specified **DiagramNode** participates in a cycle.
	findLongestPath	Finds the longest path between the specified **DiagramNode** objects.
	findShortestPath	Finds the shortest path between two **DiagramNode** elements, optionally considering the Weight of nodes and / or links.
	findShortestRoute	Calculates a route that passes through all nodes of the diagram, using only existing links. This implementation of the Traveling Salesperson Problem uses the nearest-neighbor heuristic.
	generateShortestRoute	Calculates a route that passes through all nodes of the diagram. This implementation of the Traveling Salesperson Problem uses the nearest-neighbor heuristic.

See Also

PathFinder Class
MindFusion.Diagramming Namespace

PathFinder Constructor

Initializes a new instance of the PathFinder class.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

JavaScript

```
function PathFinder (diagram)
```

Parameters

diagram
An instance of the Diagram class in which to look for paths or cycles.

See Also

PathFinder Class
MindFusion.Diagramming Namespace

PathFinder Methods

PathFinder.findAllCycles Method

Finds all cycles in the underlying diagram.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function findAllCycles ()
```

Return Value

An array of Path objects; if no cycles are detected, the array is empty.

See Also

PathFinder Members
PathFinder Class
MindFusion.Diagramming Namespace
findCycle Method

PathFinder.findAllPaths Method

Finds and returns all paths that exist between two DiagramNode objects.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function findAllPaths (from, to, [maxPaths])
```

Parameters

from
The first DiagramNode in the Path.

to
The last **DiagramNode** in the **Path**.

maxPaths
Optional.
Number. The maximum number of paths to find.

Return Value

An array of **Path** objects; if no paths are found, the array is empty.

Remarks

The returned Path instances include the *from* and *to* nodes.

See Also

PathFinder Members
PathFinder Class
MindFusion.Diagramming Namespace

PathFinder.findCycle Method

Detects whether the specified DiagramNode participates in a cycle.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function findCycle (participant)
```

Parameters

participant
A DiagramNode element which should participate in the found cycle.

Return Value

A Path instance that represents the found cycle; **null** if no cycle is detected.

See Also

PathFinder Members
PathFinder Class
MindFusion.Diagramming Namespace
findAllCycles Method

PathFinder.findLongestPath Method

Finds the longest path between the specified DiagramNode objects.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

```
JavaScript
function findLongestPath ([from, [to]])
```

Parameters

from
Optional.
The first DiagramNode element in the Path.

to
Optional.
The last **DiagramNode** element in the **Path**.

Return Value

A **Path** instance that represents the longest path found; **null** if there is no **Path** connecting *from* and *to*.

See Also

PathFinder Members
PathFinder Class

PathFinder.findShortestPath Method

Finds the shortest path between two DiagramNode elements, optionally considering the Weight of nodes and / or links.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

JavaScript

```
function findShortestPath (from, to, [useNodeWeights, [useLinkWeights]])
```

Parameters

from
The first DiagramNode in the Path.

to
The last **DiagramNode** in the **Path**.

useNodeWeights
Optional.
If **true**, the **Path** with minimal total weight is found, summing the Weight of nodes in the **Path**.

useLinkWeights
Optional.
If **true**, the **Path** with minimal total weight is found, summing the Weight of links in the **Path**.

Return Value

A **Path** instance that represents the shortest **Path** found or **null** if there is no **Path** connecting *from* and *to*.

See Also

PathFinder Members
PathFinder Class
MindFusion.Diagramming Namespace
findAllPaths Method

PathFinder.findShortestRoute Method

Calculates a route that passes through all nodes of the diagram, using only existing links. This implementation of the Traveling Salesperson Problem uses the nearest-neighbor heuristic.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

JavaScript

```
function findShortestRoute ()
```

Return Value

A Path object with its nodes, links and items fields populated, representing the approximate shortest route found by the heuristic.

Remarks

This method expects to be called on a connected graph and builds a path consisting of existing diagram links. The weight of the links is taken from their weight property.

See Also

PathFinder Members
PathFinder Class
MindFusion.Diagramming Namespace
generateShortestRoute Method

PathFinder.generateShortestRoute Method

Calculates a route that passes through all nodes of the diagram. This implementation of the Traveling Salesperson Problem uses the nearest-neighbor heuristic.

Namespace: MindFusion.Diagramming
File: PathFinder.js

Syntax

JavaScript

```
function generateShortestRoute ([weight])
```

Parameters

weight
Optional.
function. A function that takes two DiagramNode arguments and returns a number that represents the distance between them. If not provided, the geometric distance between the nodes' centers is used.

Return Value

A Path object with its nodes and items fields populated, representing approximate shortest route found by the heuristic.

Remarks

This method does not consider current links in the diagram. Instead, it uses geometrical distance between nodes or the result of *weight* function to suggest the sequence of nodes you should connect to generate approximate shortest path passing between them.

See Also

PathFinder Members
PathFinder Class
MindFusion.Diagramming Namespace
findShortestRoute Method

PatternRouter Class

Routes links by applying patterns of segment and turn sequences and accepting them when resulting paths do not cross nodes.

Syntax

JavaScript

```
// class
PatternRouter.prototype = {}
```

Remarks

You can add custom patterns to the patterns property, or replace the standard patterns altogether. Patterns are defined using RoutePattern and RouteStep objects.

Example

The following code defines a Z-like link route that will be accepted if it does not cross obstacle nodes, nor passes by them within minimumDistance.

JavaScript

```
var rightZigzag = new RoutePattern();
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.AdvanceRelative, 0.5));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.TurnRight));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.AdvanceRelative, 1));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.TurnLeft));
rightZigzag.steps.push(
    new RouteStep(RouteStepKind.AdvanceRelative, 0.5));
router.patterns.push(rightZigzag);

// this can be defined in shorter form using a pattern string
// var rightZigzag = new RoutePattern(
//     "ADR0.5 TRR ADR1 TRL ADR0.5");
```

Inheritance Hierarchy

MindFusion.Diagramming.Router
  MindFusion.Diagramming.PatternRouter

See Also

PatternRouter Members
MindFusion.Diagramming Namespace

PatternRouter Members

Routes links by applying patterns of segment and turn sequences and accepting them when resulting paths do not cross nodes.

The following tables list the members exposed by the PatternRouter type.

Public Constructors

	Name	Description
	PatternRouter	Initializes a new instance of the PatternRouter class.

Public Properties

	Name	Description
	avoidOverlaps	Gets or sets a value indicating whether to avoid overlapping links. (Inherited from Router.)
	minimumDistance	Gets or sets the minimum allowed distance from obstacles for a route to be accepted.
	patterns	Gets or sets a list of RoutePattern objects representing the routing patterns.
	preferredDistance	Gets or sets the preferred distance from link's bend points to the link's nodes.

Public Methods

	Name	Description
	route	Called when a single link must be routed. (Inherited from Router.)
	routeInternal	Overrides Router.routeInternal.
	routeLinks	Called when more than one link must be routed. (Inherited from Router.)

See Also

PatternRouter Class
MindFusion.Diagramming Namespace

PatternRouter Constructor

Initializes a new instance of the PatternRouter class.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

JavaScript
`function PatternRouter ([diagram])`

Parameters

diagram
Optional.
Diagram. The diagram this router is associated with.

See Also

PatternRouter Class
MindFusion.Diagramming Namespace

PatternRouter Methods

PatternRouter.routeInternal Method

Overrides Router.routeInternal.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

```
JavaScript
function routeInternal (link)
```

Parameters

link
DiagramLink. The link to route.

Return Value

Array. An array of points representing the link's new shape, or null if routing fails.

See Also

PatternRouter Members
PatternRouter Class
MindFusion.Diagramming Namespace

PatternRouter Properties

PatternRouter.minimumDistance Property

Gets or sets the minimum allowed distance from obstacles for a route to be accepted.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

```
JavaScript
get minimumDistance() {}
set minimumDistance(value) {}
```

Property Value

Number

A number specifying minimum distance. The default value is 4.

See Also

PatternRouter Members
PatternRouter Class
MindFusion.Diagramming Namespace

PatternRouter.patterns Property

Gets or sets a list of RoutePattern objects representing the routing patterns.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

```
JavaScript
get patterns() {}
set patterns(value) {}
```

Property Value

Array

An array of RoutePattern objects defining link route templates.

See Also

PatternRouter Members
PatternRouter Class
MindFusion.Diagramming Namespace

PatternRouter.preferredDistance Property

Gets or sets the preferred distance from link's bend points to the link's nodes.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

```
JavaScript
get preferredDistance() {}
set preferredDistance(value) {}
```

Property Value

Number

A number specifying specifying preferred distance. The default value is 10.

Remarks

The value of this property can be used in route templates using an AdvancePreferred RouteStep.

See Also

PatternRouter Members
PatternRouter Class

RelativeToLink Enumeration

Specifies how the link labels are positioned.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
RelativeToLink = {}
```

Members

	Member name	Description
	ControlPoint	The link labels are positioned relative to a specified control point.
	LinkLength	The link labels are positioned relative to the link.
	Segment	The link labels are positioned relative to a specified segment.

Remarks

Members of this enumeration can be passed as arguments to LinkLabel positioning methods.

See Also

MindFusion.Diagramming Namespace

RelativeToNode Enumeration

Specifies how the node labels are positioned.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

JavaScript

```
// enum
RelativeToNode = {}
```

Members

	Member name	Description
	Center	The node labels are positioned relative to the node center.

	Corner	The node labels are positioned relative to a specified corner point.
	Edge	The node labels are positioned relative to a specified edge.

See Also

MindFusion.Diagramming Namespace

RemoveItemCommand Class

Implements deletion of diagram items.

Syntax

JavaScript

```
// class
RemoveItemCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.RemoveItemCommand

See Also

RemoveItemCommand Members
MindFusion.Diagramming Namespace

RemoveItemCommand Members

Implements deletion of diagram items.

The following tables list the members exposed by the RemoveItemCommand type.

Public Constructors

	Name	Description
	RemoveItemCommand	Initializes a new instance of the **RemoveItemCommand** class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

RemoveItemCommand Class
MindFusion.Diagramming Namespace

RemoveItemCommand Constructor

Initializes a new instance of the RemoveItemCommand class.

Namespace: MindFusion.Diagramming
File: RemoveItemCommand.js

Syntax

```
JavaScript
function RemoveItemCommand (diagram, item)
```

Parameters

diagram
The Diagram modified by this command.

item
A DiagramItem that should be removed from the **Diagram**.

See Also

RemoveItemCommand Class
MindFusion.Diagramming Namespace

RemoveItemCommand Methods

RemoveItemCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: RemoveItemCommand.js

Syntax

```
JavaScript
function execute ()
```

See Also

RemoveItemCommand Members
RemoveItemCommand Class
MindFusion.Diagramming Namespace

RemoveItemCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: RemoveItemCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

RemoveItemCommand Members
RemoveItemCommand Class
MindFusion.Diagramming Namespace

RemoveItemCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: RemoveItemCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

RemoveItemCommand Members
RemoveItemCommand Class
MindFusion.Diagramming Namespace

ResizeTableColumnCommand Class

Implements resizing of table's columns.

Syntax

```
JavaScript
// class
ResizeTableColumnCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.ResizeTableColumnCommand

See Also

ResizeTableColumnCommand Members
MindFusion.Diagramming Namespace

ResizeTableColumnCommand Members

Implements resizing of table's columns.

The following tables list the members exposed by the ResizeTableColumnCommand type.

Public Constructors

	Name	Description
	ResizeTableColumnCommand	Initializes a new instance of the ResizeTableColumnCommand class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

ResizeTableColumnCommand Class
MindFusion.Diagramming Namespace

ResizeTableColumnCommand Constructor

Initializes a new instance of the ResizeTableColumnCommand class.

Namespace: MindFusion.Diagramming
File: ResizeTableColumnCommand.js

Syntax

JavaScript

```
function ResizeTableColumnCommand (diagram, table, column)
```

Parameters

diagram
Diagram. The Diagram modified by this command.

table
TableNode. The TableNode instance that will be modified.

column
Object. The column object that will be resized.

See Also

ResizeTableColumnCommand Class
MindFusion.Diagramming Namespace

ResizeTableColumnCommand Methods

ResizeTableColumnCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: ResizeTableColumnCommand.js

Syntax

```
JavaScript
function execute ()
```

See Also

ResizeTableColumnCommand Members
ResizeTableColumnCommand Class
MindFusion.Diagramming Namespace

ResizeTableColumnCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: ResizeTableColumnCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

ResizeTableColumnCommand Members
ResizeTableColumnCommand Class
MindFusion.Diagramming Namespace

ResizeTableColumnCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: ResizeTableColumnCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

ResizeTableColumnCommand Members
ResizeTableColumnCommand Class
MindFusion.Diagramming Namespace

ResizeTableRowCommand Class

Implements resizing of table's rows.

Syntax

JavaScript

```
// class
ResizeTableRowCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.ResizeTableRowCommand

See Also

ResizeTableRowCommand Members
MindFusion.Diagramming Namespace

ResizeTableRowCommand Members

Implements resizing of table's rows.

The following tables list the members exposed by the ResizeTableRowCommand type.

Public Constructors

	Name	Description
	ResizeTableRowCommand	Initializes a new instance of the ResizeTableRowCommand class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

ResizeTableRowCommand Class
MindFusion.Diagramming Namespace

ResizeTableRowCommand Constructor

Initializes a new instance of the ResizeTableRowCommand class.

Namespace: MindFusion.Diagramming
File: ResizeTableRowCommand.js

Syntax

```
JavaScript
function ResizeTableRowCommand (diagram, table, row)
```

Parameters

diagram
Diagram. The Diagram modified by this command.

table
TableNode. The TableNode instance that will be modified.

row
Object. The row object that will be resized.

See Also

ResizeTableRowCommand Class
MindFusion.Diagramming Namespace

ResizeTableRowCommand Methods

ResizeTableRowCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: ResizeTableRowCommand.js

Syntax

```
JavaScript
function execute ()
```

See Also

ResizeTableRowCommand Members
ResizeTableRowCommand Class
MindFusion.Diagramming Namespace

ResizeTableRowCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: ResizeTableRowCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

ResizeTableRowCommand Members
ResizeTableRowCommand Class
MindFusion.Diagramming Namespace

ResizeTableRowCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: ResizeTableRowCommand.js

Syntax

```
JavaScript
function undo ()
```

See Also

ResizeTableRowCommand Members
ResizeTableRowCommand Class
MindFusion.Diagramming Namespace

RoutePattern Class

Represents a template from which routes are created.

Syntax

```
JavaScript
// class
RoutePattern.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.RoutePattern

See Also

RoutePattern Members
MindFusion.Diagramming Namespace

RoutePattern Members

Represents a template from which routes are created.

The following tables list the members exposed by the RoutePattern type.

Public Properties

	Name	Description
	steps	A list of RouteStep objects that generate a route.

See Also

RoutePattern Class
MindFusion.Diagramming Namespace

RoutePattern Properties

RoutePattern.steps Property

A list of RouteStep objects that generate a route.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

JavaScript

```
get steps() {}
set steps(value) {}
```

Property Value

Array

An array of RouteStep objects.

See Also

RoutePattern Members
RoutePattern Class
MindFusion.Diagramming Namespace

Router Class

Implements a simple link routing algorithm that pulls link segments out of nodes they would overlap.

Syntax

JavaScript

```
// class
Router.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Router
MindFusion.Diagramming.CompositeRouter
MindFusion.Diagramming.GridRouter
MindFusion.Diagramming.PatternRouter

See Also

Router Members
MindFusion.Diagramming Namespace
autoRoute Property
routeAllLinks Method
obstacle Property

Router Members

Implements a simple link routing algorithm that pulls link segments out of nodes they would overlap.

The following tables list the members exposed by the Router type.

Public Constructors

	Name	Description
	Router	Initializes a new instance of the Router class.

Public Properties

	Name	Description
	avoidOverlaps	Gets or sets a value indicating whether to avoid overlapping links.

Public Methods

	Name	Description
	route	Called when a single link must be routed.
	routeInternal	Calculates actual route and returns it as an array of points.
	routeLinks	Called when more than one link must be routed.

See Also

Router Class
MindFusion.Diagramming Namespace

Router Constructor

Initializes a new instance of the Router class.

Namespace: MindFusion.Diagramming
File: Router.js

Syntax

```
JavaScript
function Router ([diagram])
```

Parameters

diagram
Optional.
Diagram. The diagram this router is associated with.

See Also

Router Class
MindFusion.Diagramming Namespace

Router Methods

Router.route Method

Called when a single link must be routed.

Namespace: MindFusion.Diagramming
File: Router.js

Syntax

```
JavaScript
function route (link)
```

Parameters

link
DiagramLink. The link to route.

See Also

Router Members
Router Class
MindFusion.Diagramming Namespace

Router.routeInternal Method

Calculates actual route and returns it as an array of points.

Namespace: MindFusion.Diagramming
File: Router.js

Syntax

```
JavaScript
function routeInternal (link)
```

Parameters

link
DiagramLink. The link to route.

Return Value

Array.<Point>. An array of points representing the link's new shape, or null if routing fails.

See Also

Router Members
Router Class
MindFusion.Diagramming Namespace

Router.routeLinks Method

Called when more than one link must be routed.

Namespace: MindFusion.Diagramming
File: Router.js

Syntax

JavaScript

```
function routeLinks (links)
```

Parameters

links
Array.<DiagramLink>. A list of links to route.

See Also

Router Members
Router Class
MindFusion.Diagramming Namespace

Router Properties

Router.avoidOverlaps Property

Gets or sets a value indicating whether to avoid overlapping links.

Namespace: MindFusion.Diagramming
File: Router.js

Syntax

JavaScript

```
get avoidOverlaps() {}
set avoidOverlaps(value) {}
```

Property Value

boolean

true to avoid link-link overlaps, otherwise false.

See Also

Router Members
Router Class
MindFusion.Diagramming Namespace

RouteStep Class

Represents segments or turns in routing patterns.

Syntax

JavaScript
`// class` `RouteStep.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.RouteStep

See Also

RouteStep Members
MindFusion.Diagramming Namespace

RouteStep Members

Represents segments or turns in routing patterns.

The following tables list the members exposed by the RouteStep type.

Public Constructors

	Name	Description
	RouteStep	Initializes a new instance of the **RouteStep** class.

See Also

RouteStep Class
MindFusion.Diagramming Namespace

RouteStep Constructor

Initializes a new instance of the RouteStep class.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

JavaScript

```
function RouteStep (kind, [amount])
```

Parameters

kind
RouteStepKind. A member of the **RouteStepKind** enumeration.

amount
Optional.
Number. The distance to add when step kind is Advance*.

See Also

RouteStep Class
MindFusion.Diagramming Namespace

RouteStepKind Enumeration

Represents parts of a route template.

Namespace: MindFusion.Diagramming
File: PatternRouter.js

Syntax

JavaScript

```
// enum
RouteStepKind = {}
```

Members

	Member name	Description
	AdvanceAbsolute	Advance by the specified absolute distance.Corresponds to "ADA" command in pattern strings.
	AdvanceDest	Advance by the specified relative remaining distance to destination node.Corresponds to "ADD" command in pattern strings.
	AdvancePreferred	Advance by PreferredDistance.Corresponds to "ADP" command in pattern strings.
	AdvanceRelative	Advance by the specified relative distance from origin to destination node.Equivalent to "ADR" command in pattern strings.
	AdvanceRemaining	Advance by the specified relative remaining distance to link's end point.Corresponds to "ADM" command in pattern strings.
	AlignToDestination	Advance to align with nearest side of destination node.Corresponds to "ALD" command in pattern strings.

	AlignToEnd	Advance to align with link's end point.Corresponds to "ALE" command in pattern strings.
	AlignToFarBorder	Advance to align with furthest border of both nodes.Corresponds to "AFB" command in pattern strings.
	AlignToOrigin	Advance to align with nearest side of origin node.Corresponds to "ALO" command in pattern strings.
	AlignToStart	Advance to align with link's start point.Corresponds to "ALS" command in pattern strings.
	TurnLeft	Turn 90 degrees to the left.Corresponds to "TRL" command in pattern strings.
	TurnRight	Turn 90 degrees to the right.Corresponds to "TRR" command in pattern strings.

See Also

MindFusion.Diagramming Namespace

Ruler Class

The **Ruler** control provides horizontal and vertical scales that help users measure and align diagram items.

Syntax

JavaScript

```
// class
Ruler.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.Control
  MindFusion.Diagramming.Ruler

See Also

Ruler Members
MindFusion.Diagramming Namespace

Ruler Members

The Ruler control provides horizontal and vertical scales that help users measure and align diagram items.

The following tables list the members exposed by the **Ruler** type.

Public Constructors

	Name	Description
	Ruler	Initializes a new instance of the Ruler class.

Public Properties

	Name	Description
	backColor	Gets or sets the background color of the control.
	diagramView	Gets or sets the DiagramView shown by this Ruler.
	element	Returns a reference to the control's DOM element. (Inherited from Control.)
	enabled	Gets or sets whether mouse events are enabled. (Inherited from Control.)
	enableGuides	Gets or sets a value indicating whether the alignment guides are enabled.
	foreColor	Gets or sets the foreground color of the control.
	guideColor	Gets or sets the color of the alignment guide lines.
	horizontalScaleVisible	Gets or sets a value indicating whether the horizontal ruler scale is visible.
	licenseLocation	Gets or sets the location of the license key file. (Inherited from Control.)
	negatedX	Gets or sets whether the horizontal scale should show negated values, simulating that the coordinate system's X axis grows to the left.
	negatedY	Gets or sets whether the vertical scale should show negated values, simulating that the coordinate system's Y axis grows to the top.
	pointerColor	Gets or sets the color of the mouse pointer's projections.
	projectionColor	Gets or sets the color used to paint the projection of the active node.
	projectRotatedBounds	Gets or sets whether projections of nodes on ruler's scales reflect their rotated boundaries.
	textColor	Gets or sets the color used to paint text in the ruler.
	unit	Gets the measure units displayed in the ruler scales.
	verticalScaleVisible	Gets or sets a value indicating whether the vertical ruler scale is visible.

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the instance. (Inherited from Disposable.)
S	addHandlers	Adds an event listener to a DOM element. (Inherited from Control.)
S	clearHandlers	Removes all event listeners from a DOM element. (Inherited from Control.)

	create	Creates and initializes a new Ruler from the specified element. This method is static and can be called without creating an instance of the class.
	find	Returns the specified Ruler object. This member is static and can be invoked without creating an instance of the class.
	fromJson	Constructs a JavaScript object from a JSON string. (Inherited from Control.)
	raiseEvent	Raises an event. (Inherited from Disposable.)
	registerForSubmit	Registers an onsubmit handler for the Ruler's parent html form to flush postback data.
	removeEventListener	Removes a single event listener attached to the instance. (Inherited from Disposable.)
	toJson	Converts a JavaScript object to a JSON string. (Inherited from Control.)

Public Events

	Name	Description
	controlLoaded	Raised when the control is loaded. (Inherited from Control.)
	controlUnloaded	Raised when the control is unloaded. (Inherited from Control.)
	controlUnloading	Raised when the control is unloading. (Inherited from Control.)

See Also

Ruler Class
MindFusion.Diagramming Namespace

Ruler Constructor

Initializes a new instance of the Ruler class.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
function Ruler (element)
```

Parameters

element
HTMLDivElement. The Canvas DOM Element this Ruler is associated with.

See Also

Ruler Class
MindFusion.Diagramming Namespace

Ruler Methods

Ruler.create Method

Creates and initializes a new Ruler from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

```
JavaScript
function create (element)
```

Parameters

element
HTMLDivElement. The DOM element that the ruler should be attached to.

Return Value

A Ruler object that represents the newly created ruler.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.find Method

Returns the specified Ruler object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

```
JavaScript
function find (id)
```

Parameters

id
String. The id of the Ruler's element.

Return Value

A Ruler object with the specified id, if found, or **null** otherwise.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.registerForSubmit Method

Registers an onsubmit handler for the Ruler's parent html form to flush postback data.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
function registerForSubmit (id)
```

Parameters

id
String. The id of the hidden field to flush the data to.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler Properties

Ruler.backColor Property

Gets or sets the background color of the control.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get backColor() {}
set backColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Ruler Members
Ruler Class

Ruler.diagramView Property

Gets or sets the DiagramView shown by this Ruler.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

```
JavaScript
get diagramView() {}
set diagramView(value) {}
```

Property Value

DiagramView

The DiagramView attached to this ruler.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.enableGuides Property

Gets or sets a value indicating whether the alignment guides are enabled.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

```
JavaScript
get enableGuides() {}
set enableGuides(value) {}
```

Property Value

Boolean

true to allow aligning diagram items using ruler guides, or **false** otherwise. The default is **true**.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.foreColor Property

Gets or sets the foreground color of the control.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

```
JavaScript
get foreColor() {}
set foreColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.guideColor Property

Gets or sets the color of the alignment guide lines.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

```
JavaScript
get guideColor() {}
set guideColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.horizontalScaleVisible Property

Gets or sets a value indicating whether the horizontal ruler scale is visible.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get horizontalScaleVisible() {}
set horizontalScaleVisible(value) {}
```

Property Value

Boolean

true to display the horizontal scale, or **false** otherwise. The default is **true**.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.negatedX Property

Gets or sets whether the horizontal scale should show negated values, simulating that the coordinate system's X axis grows to the left.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get negatedX() {}
set negatedX(value) {}
```

Property Value

Boolean

true if the horizontal scale should show negated values, or **false** otherwise. The default value is **false**.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.negatedY Property

Gets or sets whether the vertical scale should show negated values, simulating that the coordinate system's Y axis grows to the top.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get negatedY() {}
set negatedY(value) {}
```

Property Value

Boolean

true if the vertical scale should show negated values, or **false** otherwise. The default value is **false**.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.pointerColor Property

Gets or sets the color of the mouse pointer's projections.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get pointerColor() {}
set pointerColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.projectionColor Property

Gets or sets the color used to paint the projection of the active node.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get projectionColor() {}
set projectionColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.projectRotatedBounds Property

Gets or sets whether projections of nodes on ruler's scales reflect their rotated boundaries.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get projectRotatedBounds() {}
set projectRotatedBounds(value) {}
```

Property Value

Boolean

true if node's projection should match the boundaries of rotated node shape, or **false** to ignore rotation and project only the coordinates of node's Bounds. The default value is **true**.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.textColor Property

Gets or sets the color used to paint text in the ruler.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.unit Property

Gets the measure units displayed in the ruler scales.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get unit() {}
set unit(value) {}
```

Property Value

GraphicsUnit

A member of the GraphicsUnit enumeration.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler.verticalScaleVisible Property

Gets or sets a value indicating whether the vertical ruler scale is visible.

Namespace: MindFusion.Diagramming
File: Ruler.js

Syntax

JavaScript

```
get verticalScaleVisible() {}
set verticalScaleVisible(value) {}
```

Property Value

Boolean

true to display the vertical scale, or **false** otherwise. The default is **true**.

See Also

Ruler Members
Ruler Class
MindFusion.Diagramming Namespace

Ruler Events

SaveToStringFormat Enumeration

Specifies in what format to save the diagram when generating a string using the saveToString method.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
// enum
SaveToStringFormat = {}
```

Members

	Member name	Description
	Json	Indicates JSON format.
	Xml	Indicates XML format.

See Also

MindFusion.Diagramming Namespace

ScaleMode Enumeration

Defines Overview scaling modes.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
// enum
ScaleMode = {}
```

Members

	Member name	Description
	CombinedScales	Combine diagram's ZoomFactor with overview's ScaleFactor.
	FitAll	Scale the Overview dynamically to fit the entire diagram.
	FixedScale	Use fixed scale specified via ScaleFactor, ignoring diagram's zoom level.

Remarks

Members of this enumeration can be assigned to the scaleMode property of **Overview**.

See Also

MindFusion.Diagramming Namespace

ScrollTableCommand Class

Handles undo / redo of table scrolling.

Syntax

```
// class
ScrollTableCommand.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Command
  MindFusion.Diagramming.ScrollTableCommand

See Also

ScrollTableCommand Members
MindFusion.Diagramming Namespace

ScrollTableCommand Members

Handles undo / redo of table scrolling.

The following tables list the members exposed by the ScrollTableCommand type.

Public Constructors

	Name	Description
	ScrollTableCommand	Initializes a new instance of the **ScrollTableCommand** class.

Public Methods

	Name	Description
	execute	Command.execute override.
	redo	Command.redo override.
	undo	Command.undo override.

See Also

ScrollTableCommand Class
MindFusion.Diagramming Namespace

ScrollTableCommand Constructor

Initializes a new instance of the ScrollTableCommand class.

Namespace: MindFusion.Diagramming
File: ScrollTableCommand.js

Syntax

```
JavaScript
function ScrollTableCommand (diagram, table)
```

Parameters

diagram
The Diagram modified by this command.

table
The TableNode instance that will be modified.

See Also

ScrollTableCommand Class
MindFusion.Diagramming Namespace

ScrollTableCommand Methods

ScrollTableCommand.execute Method

Command.execute override.

Namespace: MindFusion.Diagramming
File: ScrollTableCommand.js

Syntax

```
JavaScript
function execute ()
```

See Also

ScrollTableCommand Members
ScrollTableCommand Class
MindFusion.Diagramming Namespace

ScrollTableCommand.redo Method

Command.redo override.

Namespace: MindFusion.Diagramming
File: ScrollTableCommand.js

Syntax

```
JavaScript
function redo ()
```

See Also

ScrollTableCommand Members
ScrollTableCommand Class
MindFusion.Diagramming Namespace

ScrollTableCommand.undo Method

Command.undo override.

Namespace: MindFusion.Diagramming
File: ScrollTableCommand.js

Syntax

JavaScript

```
function undo ()
```

See Also

ScrollTableCommand Members
ScrollTableCommand Class
MindFusion.Diagramming Namespace

Selection Class

Represents a set of items selected in the Diagram document.

Syntax

JavaScript

```
// class
Selection.prototype = {}
```

Remarks

Several diagram items could be selected simultaneously. If more than one item is selected, the compound set of selected items is handled by an instance of the **Selection** class, returned by diagram's selection property. Items in the selection can be of any element type. The items property returns an array of all items, while more specific types are returned by links and nodes properties.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.Selection

See Also

Selection Members
MindFusion.Diagramming Namespace

Selection Members

Represents a set of items selected in the Diagram document.

The following tables list the members exposed by the Selection type.

Public Constructors

	Name	Description
	Selection	Initializes a new instance of the **Selection** class.

Public Properties

	Name	Description
	allowMultipleSelection	Gets or sets a value indicating whether more than one items can be selected simultaneously.
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)

	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	items	Returns an array of all items in this selection.
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	links	Returns an array of all links in this selection.
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	nodes	Returns an array of all nodes in this selection.
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	parent	Gets the **Diagram** to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)

	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description

	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addItem	Adds an item to the **Selection**.
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override.
	animate	Animates the item. (Inherited from DiagramItem.)
	clear	Removes all items from the **selection**.
	clone	Creates a copy of this item. (Inherited from DiagramItem.)
	containsPoint	Determines whether this item contains the specified point. (Inherited from DiagramItem.)
	endDrag	DiagramItem.endDrag override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getBounds	Gets a rectangle that defines the position of the selection frame.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	Loads the item content from an XML element. (Inherited from DiagramItem.)
	onRemove	DiagramItem.onRemove override.
	removeCanvasElements	Derived classes must implement this method by removing their graphics primitives from the canvas.elements collection. (Inherited from DiagramItem.)
	removeItem	Removes an **item** from the **Diagram**.
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	DiagramItem.saveLocationState override.
	saveState	Saves the properties of this diagram item. (Inherited from DiagramItem.)
	saveToXml	Saves the item content into an XML element. (Inherited from DiagramItem.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)

	startDrag	**DiagramItem.startDrag** override.
	toJson	Serializes this item into a JSON object. (Inherited from DiagramItem.)
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	DiagramItem.updateDrag override.

See Also

Selection Class
MindFusion.Diagramming Namespace

Selection Fields

Selection Constructor

Initializes a new instance of the Selection class.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function Selection (parent)
```

Parameters

parent
The diagram where the items are selected.

See Also

Selection Class
MindFusion.Diagramming Namespace

Selection Methods

Selection.addItem Method

Adds an item to the Selection.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function addItem (item)
```

Parameters

item
The item to add.

Remarks

This method adds a new item to the selection. It also deactivates the activeItem, if any, and makes the newly added item the active one.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.allowDrag Method

DiagramItem.allowDrag override.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function allowDrag (ist)
```

Parameters

ist
InteractionState. An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.clear Method

Removes all items from the selection.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function clear ()
```

Remarks

Call clear to deselect all items. The items are not deleted. The method also sets the diagram's activeItem to Null.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.endDrag Method

DiagramItem.endDrag override.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function endDrag (ist)
```

Parameters

ist
InteractionState. An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.getBounds Method

Gets a rectangle that defines the position of the selection frame.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function getBounds ()
```

Return Value

Rect. The frame's bounding rectangle.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.onRemove Method

DiagramItem.onRemove override.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function onRemove ()
```

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.removeItem Method

Removes an item from the Diagram.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function removeItem (item)
```

Parameters

item
The item to add.

Remarks

Call this method to deselect an item. Note that the item is not deleted, but just removed from the selection.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.saveLocationState Method

DiagramItem.saveLocationState override.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function saveLocationState ()
```

Return Value

Object. An object containing selection state.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.startDrag Method

DiagramItem.startDrag override.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function startDrag ([ist])
```

Parameters

ist
Optional. An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.updateDrag Method

DiagramItem.updateDrag override.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
function updateDrag (ist)
```

Parameters

ist
An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection Properties

Selection.allowMultipleSelection Property

Gets or sets a value indicating whether more than one items can be selected simultaneously.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
get allowMultipleSelection() {}
set allowMultipleSelection(value) {}
```

Property Value

Boolean

true to let the user select multiple items, or **false** otherwise. The default value is **true**.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.items Property

Returns an array of all items in this selection.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
get items() {}
```

Property Value

Array

Array of DiagramItem objects.

Remarks

Do not modify this array directly, but call Selection methods such as addItem and removeItem, which also update item's state variables and redraw the diagram.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.links Property

Returns an array of all links in this selection.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
get links() {}
```

Property Value

DiagramLink

Array of DiagramLink objects.

Remarks

Do not modify this array directly, but call Selection methods such as addItem and removeItem, which also update item's state variables and redraw the diagram.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

Selection.nodes Property

Returns an array of all nodes in this selection.

Namespace: MindFusion.Diagramming
File: Selection.js

Syntax

JavaScript

```
get nodes() {}
```

Property Value

DiagramNode

Array of DiagramNode objects.

Remarks

Do not modify this array directly, but call Selection methods such as addItem and removeItem, which also update item's state variables and redraw the diagram.

See Also

Selection Members
Selection Class
MindFusion.Diagramming Namespace

SelectionChangedEventArgs Class

Contains the arguments passed to handlers of the selectionChanged event.

Syntax

JavaScript

```
// class
SelectionChangedEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.EventArgs
  MindFusion.Diagramming.SelectionChangedEventArgs

See Also

SelectionChangedEventArgs Members
MindFusion.Diagramming Namespace
Events Class

SelectionChangedEventArgs Members

Contains the arguments passed to handlers of the selectionChanged event.

The following tables list the members exposed by the SelectionChangedEventArgs type.

Public Constructors

	Name	Description
	SelectionChangedEventArgs	Initializes a new instance of the SelectionChangedEventArgs class.

Public Properties

	Name	Description
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	newItems	Gets the selected items after the change.
	oldItems	Gets the selected items before the change.

See Also

SelectionChangedEventArgs Class
MindFusion.Diagramming Namespace

SelectionChangedEventArgs Constructor

Initializes a new instance of the SelectionChangedEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function SelectionChangedEventArgs (args)
```

Parameters

args

See Also

SelectionChangedEventArgs Class
MindFusion.Diagramming Namespace

SelectionChangedEventArgs Properties

SelectionChangedEventArgs.newItems Property

Gets the selected items after the change.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get newItems() {}
set newItems(value) {}
```

Property Value

Array

An array of items.

See Also

SelectionChangedEventArgs Members
SelectionChangedEventArgs Class
MindFusion.Diagramming Namespace

SelectionChangedEventArgs.oldItems Property

Gets the selected items before the change.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get oldItems() {}
set oldItems(value) {}
```

Property Value

Array

An array of items.

See Also

SelectionChangedEventArgs Members
SelectionChangedEventArgs Class
MindFusion.Diagramming Namespace

SelectionEventArgs Class

Contains the arguments passed to handlers of selection-related events.

Syntax

```
JavaScript
// class
SelectionEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
  MindFusion.Diagramming.SelectionEventArgs

See Also

SelectionEventArgs Members
MindFusion.Diagramming Namespace
Events Class

SelectionEventArgs Members

Contains the arguments passed to handlers of selection-related events.

The following tables list the members exposed by the SelectionEventArgs type.

Public Constructors

	Name	Description
	SelectionEventArgs	Initializes a new instance of the SelectionEventArgs class.

Public Properties

	Name	Description
	adjustmentHandle	Gets the adjustment handle being moved by the user.
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	mousePosition	Gets the current mouse pointer position.

Public Methods

	Name	Description
	cancelDrag	Immediately cancels the current selection modification.

See Also

SelectionEventArgs Class
MindFusion.Diagramming Namespace

SelectionEventArgs Constructor

Initializes a new instance of the SelectionEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript
`function SelectionEventArgs (args)`

Parameters

args

See Also

SelectionEventArgs Class
MindFusion.Diagramming Namespace

SelectionEventArgs Methods

SelectionEventArgs.cancelDrag Method

Immediately cancels the current selection modification.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function cancelDrag ()
```

See Also

SelectionEventArgs Members
SelectionEventArgs Class
MindFusion.Diagramming Namespace

SelectionEventArgs Properties

SelectionEventArgs.adjustmentHandle Property

Gets the adjustment handle being moved by the user.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get adjustmentHandle() {}
```

Property Value

AdjustmentHandles

A member of the AdjustmentHandles enumeration.

See Also

SelectionEventArgs Members
SelectionEventArgs Class
MindFusion.Diagramming Namespace

SelectionEventArgs.mousePosition Property

Gets the current mouse pointer position.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get mousePosition() {}
```

Property Value

Point

A Point instance containing mouse pointer coordinates.

See Also

SelectionEventArgs Members
SelectionEventArgs Class
MindFusion.Diagramming Namespace

SelectOnlyBehavior Class

Allows only selection of existing items. Modifying them or drawing new ones is disabled.

Syntax

JavaScript

```
// class
SelectOnlyBehavior.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.BehaviorBase
  MindFusion.Diagramming.SelectOnlyBehavior

See Also

SelectOnlyBehavior Members
MindFusion.Diagramming Namespace

SelectOnlyBehavior Members

Allows only selection of existing items. Modifying them or drawing new ones is disabled.

The following tables list the members exposed by the SelectOnlyBehavior type.

Public Constructors

	Name	Description
	SelectOnlyBehavior	Overloaded.

Public Properties

	Name	Description
	diagram	Gets the target diagram of this behavior object. (Inherited from BehaviorBase.)

Public Methods

	Name	Description
	createController	BehaviorBase.createController override.

See Also

SelectOnlyBehavior Class
MindFusion.Diagramming Namespace

SelectOnlyBehavior Constructor

Overload List

Name	Description
SelectOnlyBehavior	Initializes a new instance of the SelectOnlyBehavior class.
SelectOnlyBehavior	Initializes a new instance of the SelectOnlyBehavior class.

See Also

SelectOnlyBehavior Class
MindFusion.Diagramming Namespace

SelectOnlyBehavior Constructor

Initializes a new instance of the SelectOnlyBehavior class.

Namespace: MindFusion.Diagramming
File: SelectOnlyBehavior.js

Syntax

JavaScript
`function SelectOnlyBehavior (parent)`

Parameters

parent
The Diagram this behavior will be associated with.

See Also

SelectOnlyBehavior Constructor Overload List
SelectOnlyBehavior Class
MindFusion.Diagramming Namespace

SelectOnlyBehavior Constructor

Initializes a new instance of the SelectOnlyBehavior class.

Namespace: MindFusion.Diagramming
File: SelectOnlyBehavior.js

Syntax

```
JavaScript
function SelectOnlyBehavior (parent)
```

Parameters

parent
DiagramView. The DiagramView this behavior will be associated with.

See Also

SelectOnlyBehavior Constructor Overload List
SelectOnlyBehavior Class
MindFusion.Diagramming Namespace

SelectOnlyBehavior Methods

SelectOnlyBehavior.createController Method

BehaviorBase.createController override.

Namespace: MindFusion.Diagramming
File: SelectOnlyBehavior.js

Syntax

```
JavaScript
function createController (state)
```

Parameters

state
Object. An object representing current context and input event.

Return Value

SinglePointerController. An instance of SinglePointerController -derived class.

See Also

SelectOnlyBehavior Members
SelectOnlyBehavior Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs Class

Contains the arguments passed to the serializeTag and deserializeTag event handlers.

Syntax

JavaScript
`// class` `SerializeTagEventArgs.prototype = {}`

Inheritance Hierarchy

MindFusion.Controls.EventArgs
MindFusion.Diagramming.SerializeTagEventArgs

See Also

SerializeTagEventArgs Members
MindFusion.Diagramming Namespace

SerializeTagEventArgs Members

Contains the arguments passed to the serializeTag and deserializeTag event handlers.

The following tables list the members exposed by the SerializeTagEventArgs type.

Public Constructors

	Name	Description
	SerializeTagEventArgs	Initializes a new instance of the SerializeTagEventArgs class.

Public Properties

	Name	Description
	context	Gets an XmlPersistContext object providing helper read and write methods for various value types.
	element	Gets the XML element where the tag value should be serialized to or deserialized from.
	jsonObject	Gets or sets the object representing the tag.
	object	Gets the object whose property is being serialized or deserialized.
	propertyName	Gets the name of the property being serialized.
	tag	Gets or sets the tag value being serialized or deserialized.

See Also

SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs Constructor

Initializes a new instance of the SerializeTagEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
function SerializeTagEventArgs ()
```

See Also

SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs Properties

SerializeTagEventArgs.context Property

Gets an XmlPersistContext object providing helper read and write methods for various value types.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get context() {}
```

Property Value

XmlPersistContext

An instance of the XmlPersistContext class.

See Also

SerializeTagEventArgs Members
SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs.element Property

Gets the XML element where the tag value should be serialized to or deserialized from.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get element() {}
```

Property Value

Element

The XML element.

See Also

SerializeTagEventArgs Members
SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs.jsonObject Property

Gets or sets the object representing the tag.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get jsonObject() {}
set jsonObject(value) {}
```

Property Value

Object

The object represeting the tag.

Remarks

Used in JSON serialization.

See Also

SerializeTagEventArgs Members
SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs.object Property

Gets the object whose property is being serialized or deserialized.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get object() {}
```

Property Value

Object

An object whose custom data is being serialized.

See Also

SerializeTagEventArgs Members
SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs.propertyName Property

Gets the name of the property being serialized.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get propertyName() {}
```

Property Value

String

A string containing property name.

See Also

SerializeTagEventArgs Members
SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

SerializeTagEventArgs.tag Property

Gets or sets the tag value being serialized or deserialized.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
get tag() {}
set tag(value) {}
```

Property Value

Object

An object containing custom data.

See Also

SerializeTagEventArgs Members
SerializeTagEventArgs Class
MindFusion.Diagramming Namespace

ShadowsStyle Enumeration

Enumerates possible shadows-drawing styles.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
ShadowsStyle = {}
```

Members

	Member name	Description
	None	No shadows are drawn.
	OneLevel	All shadows are drawn at the lowest z-level, and appear to be on one level.
	ZOrder	Every shadows is drawn just below its object, possibly covering other objects.

Remarks

Members of this enumeration can be assigned to the shadowsStyle property of Diagram.

See Also

MindFusion.Diagramming Namespace

Shape Class

An instance of this class represents a shape defined through a series of arc, line and Bezier segments. The segments can be a part of the shape outline, in which case they define the part of the shape is filled, used for hit-testing and for aligning links to the node edges. Segments can also be added as decoration elements, in which case they only affect the appearance of a ShapeNode.

Syntax

```
JavaScript
// class
Shape.prototype = {}
```

Remarks

The **Shape** class can also be used as a component in composite nodes. When its **isOutline** attribute is set, the shape will control CompositeNode's geometry, defining hit test and clip area, and link alignment points along node's border. If **isOutline** is disabled, the shape will serve mostly as a decorative element. The code below shows a sample fragment specifying shape component in template:

```
JavaScript
```

```
{
    component: "Shape",
    id: "Cloud",
    autoProperty: true,
    name: "OutlineShape",
    pen: "gray",
    brush: "Transparent",
    isOutline: true
},
```

Example

Custom shapes can be defined using SVG Path syntax. The following code creates custom shape geoometry and assigns it to a ShapeNode:

JavaScript

```
var shape = new MindFusion.Diagramming.Shape({
   id: "DoubleEllipse",
   outline: "M100,50 A50,50,0,1,1,0,50 A50,50,0,1,1,100,50",
   decoration: "M98,50 A48,48,0,1,1,2,50 A48,48,0,1,1,98,50",
   fillMode: "nonzero"});

node.shape = shape;
```

Inheritance Hierarchy

MindFusion.Diagramming.ComponentBase
MindFusion.Diagramming.Shape

See Also

Shape Members
MindFusion.Diagramming Namespace

Shape Members

An instance of this class represents a shape defined through a series of arc, line and Bezier segments. The segments can be a part of the shape outline, in which case they define the part of the shape is filled, used for hit-testing and for aligning links to the node edges. Segments can also be added as decoration elements, in which case they only affect the appearance of a ShapeNode.

The following tables list the members exposed by the Shape type.

Public Constructors

	Name	Description
	Shape	Initializes a new instance of the Shape class.

Public Properties

	Name	Description
	anchorPattern	Gets or sets the default anchor pattern for nodes that use this shape.
	brush	Gets or sets the brush used to paint the interior of this shape.

	pen	Gets or sets the pen used to stroke the frame of this shape.
	strokeThickness	Gets or sets the thickness of the shape's frame.

Public Methods

	Name	Description
	component	Creates a shape component.
	fromId	Returns a reference to the **Shape** with the specified string identifier.
	getId	Gets the id of this **Shape**.
	isElliptic	Gets a value indicating whether this shape is elliptic.
	loadFromXml	Loads the shape content from an XML element.

See Also

Shape Class
MindFusion.Diagramming Namespace

Shape Constructor

Initializes a new instance of the Shape class.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

JavaScript

```
function Shape ([params])
```

Parameters

params.outline
Optional.
String. A string defining the shape's outline.

params.decoration
Optional.
String. A string defining the shape's decoration.

params.shapeDecorations
Optional.
Array. An array containing shapeDecoration objects defining additional shape decoration.

params.id
Optional.
String. The string identifier of the shape.

params.fillMode
Optional.
String. A string specifying the fill mode;

params.outlinePen

Optional.
String. A string specifying the stroke of the shape's outline.

params.outlineBrush
Optional.
String. A string specifying the fill of the shape's outline.

params.decorationPen
Optional.
String. A string specifying the stroke of the shape's decoration.

params.decorationBrush
Optional.
String. A string specifying the fill of the shape's decoration.

See Also

Shape Class
MindFusion.Diagramming Namespace

Shape Methods

Shape.component Method

Creates a shape component.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

JavaScript

```
function component (id, [isOutline, [node]])
```

Parameters

id
String. The identifier of the shape.

isOutline
Optional.
Boolean. A value indicating whether this shape defines the outline of its host node.

node
Optional.
CompositeNode. The host node of this component.

Return Value

Shape. A reference to a Shape component with the specified identifier.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.fromId Method

Returns a reference to the Shape with the specified string identifier.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

JavaScript

```
function fromId (id)
```

Parameters

id
The identifier of the Shape of interest.

Return Value

A reference to the **Shape** with the specified identifier or **null** if there is no such shape.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.getId Method

Gets the id of this Shape.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

JavaScript

```
function getId ()
```

Return Value

The string identifier.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.isElliptic Method

Gets a value indicating whether this shape is elliptic.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

```
JavaScript
function isElliptic ()
```

Return Value

Boolean. True if the shape is elliptic, or false otherwise.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.loadFromXml Method

Loads the shape content from an XML element.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that contains the the shape's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape Properties

Shape.anchorPattern Property

Gets or sets the default anchor pattern for nodes that use this shape.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

JavaScript

```
get anchorPattern() {}
set anchorPattern(value) {}
```

Property Value

AnchorPattern

An AnchorPattern instance. The default value is **null**.

Remarks

If the anchorPattern of a ShapeNode is set, it overrides the one associated with the Shape definition.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.brush Property

Gets or sets the brush used to paint the interior of this shape.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

JavaScript

```
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

Used to paint Shape components of CompositeNode.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.pen Property

Gets or sets the pen used to stroke the frame of this shape.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

```
JavaScript
get pen() {}
set pen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

Remarks

Used to paint Shape components of CompositeNode.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

Shape.strokeThickness Property

Gets or sets the thickness of the shape's frame.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

```
JavaScript
get strokeThickness() {}
set strokeThickness(value) {}
```

Property Value

Number

A number specifying stroke thickness.

Remarks

Used to paint Shape components of CompositeNode.

See Also

Shape Members
Shape Class
MindFusion.Diagramming Namespace

ShapeDecoration Class

An instance of this class represents a child decoration figure of shape's geometry.

Syntax

```
JavaScript
// class
ShapeDecoration.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ShapeDecoration

See Also

ShapeDecoration Members
MindFusion.Diagramming Namespace

ShapeDecoration Members

An instance of this class represents a child decoration figure of shape's geometry.

The following tables list the members exposed by the ShapeDecoration type.

Public Constructors

	Name	Description
	ShapeDecoration	Initializes a new instance of the **ShapeDecoration** class.

See Also

ShapeDecoration Class
MindFusion.Diagramming Namespace

ShapeDecoration Constructor

Initializes a new instance of the ShapeDecoration class.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

```
JavaScript
function ShapeDecoration ([elements, [fillColor, [strokeColor, [strokeThickness,
[strokeDashStyle]]]]])
```

Parameters

elements
Optional.
Array. An array of ShapeElement objects.

fillColor
Optional.
String. Optional fill color.

strokeColor
Optional.
String. Optional stroke color.

strokeThickness
Optional.
Number. Optional stroke thickness.

strokeDashStyle
Optional.
DashStyle. Optional dash style.

See Also

ShapeDecoration Class
MindFusion.Diagramming Namespace

ShapeElement Class

An instance of this class represents an element of shape's geometry, such as line segment, arc or Bezier curve.

Syntax

JavaScript

```
// class
ShapeElement.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ShapeElement

See Also

ShapeElement Members
MindFusion.Diagramming Namespace

ShapeElement Members

An instance of this class represents an element of shape's geometry, such as line segment, arc or Bezier curve.

The following tables list the members exposed by the ShapeElement type.

Public Constructors

	Name	Description
	ShapeElement	Initializes a new instance of the **ShapeElement** class.

See Also

ShapeElement Class
MindFusion.Diagramming Namespace

ShapeElement Constructor

Initializes a new instance of the ShapeElement class.

Namespace: MindFusion.Diagramming
File: Shape.js

Syntax

```
JavaScript
function ShapeElement ([func, [args, [strokeColor, [strokeThickness, [strokeDashStyle]]]]])
```

Parameters

func
Optional.
String. One of the moveTo, lineTo, arc, bezierCurveTo, roundRect functions.

args
Optional.
Array. Coordinates to pass to the drawing function.

strokeColor
Optional.
String. Optional color to apply to the element.

strokeThickness
Optional.
Number. Optional thickness to apply to the element.

strokeDashStyle
Optional.
DashStyle. Optional dash style to apply to the element.

See Also

ShapeElement Class
MindFusion.Diagramming Namespace

ShapeLibrary Class

The **ShapeLibrary** class represents a collection of shape definitions stored together.

Syntax

```
JavaScript
// class
ShapeLibrary.prototype = {}
```

Remarks

Shape libraries can be created using the ShapeDesigner tool that comes with MindFusion.Diagramming. Use the loadFromXml method to load a shape library from a file. The loaded shapes can be accessed through the Shapes property.

Inheritance Hierarchy

MindFusion.Diagramming.ShapeLibrary

See Also

ShapeLibrary Members
MindFusion.Diagramming Namespace

ShapeLibrary Members

The ShapeLibrary class represents a collection of shape definitions stored together.

The following tables list the members exposed by the **ShapeLibrary** type.

Public Constructors

	Name	Description
	ShapeLibrary	Initializes a new instance of the ShapeLibrary class.

Public Fields

	Name	Description
	shapes	A list containing all the shapes.

Public Methods

	Name	Description
	getShapes	Gets the list with all shapes contained in the library.
	loadFromXml	Overloaded.

See Also

ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeLibrary Fields

ShapeLibrary.shapes Field

A list containing all the shapes.

Namespace: MindFusion.Diagramming
File: ShapeLibrary.js

Syntax

JavaScript
`var shapes`

See Also

ShapeLibrary Members
ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeLibrary Constructor

Initializes a new instance of the ShapeLibrary class.

Namespace: MindFusion.Diagramming
File: ShapeLibrary.js

Syntax

```
JavaScript
function ShapeLibrary (shapes)
```

Parameters

shapes
Array. A list of MindFusion.Diagramming.Shape objects to initialize the library with.

See Also

ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeLibrary Methods

ShapeLibrary.getShapes Method

Gets the list with all shapes contained in the library.

Namespace: MindFusion.Diagramming
File: ShapeLibrary.js

Syntax

```
JavaScript
function getShapes ()
```

Return Value

Array. A list with all the MindFusion.Diagramming.Shape objects contained in the library.

See Also

ShapeLibrary Members
ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeLibrary.loadFromXml Method

Overload List

Name	Description
ShapeLibrary.loadFromXml	Loads the library from the specified XML document.
ShapeLibrary.loadFromXml	Loads the library from the specified XML document.

See Also

ShapeLibrary Members
ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeLibrary.loadFromXml Method

Loads the library from the specified XML document.

Namespace: MindFusion.Diagramming
File: ShapeLibrary.js

Syntax

JavaScript

```
function loadFromXml (fileUrl, [onLoad, [onError]])
```

Parameters

fileUrl
String. The URL of an XML file where the data should be read from.

onLoad
Optional.
function. A callback that should be invoked if the file is loaded successfully.

onError
Optional.
function. A callback that should be invoked if the file could not be downloaded.

See Also

loadFromXml Method Overload List
ShapeLibrary Members
ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeLibrary.loadFromXml Method

Loads the library from the specified XML document.

Namespace: MindFusion.Diagramming
File: ShapeLibrary.js

Syntax

JavaScript

```
function loadFromXml (fileUrl, [onLoad, [onError]])
```

Parameters

fileUrl
The URL of an XML file where the data should be read from.

onLoad
Optional.
A callback that should be invoked if the file is loaded successfully.

onError
Optional.
A callback that should be invoked if the file could not be downloaded.

See Also

loadFromXml Method Overload List
ShapeLibrary Members
ShapeLibrary Class
MindFusion.Diagramming Namespace

ShapeNode Class

ShapeNode instances are diagram nodes that represent geometric shapes.

Syntax

JavaScript

```
// class
ShapeNode.prototype = {}
```

Remarks

ShapeNode instance might represent nodes in graphs, tree, process diagrams, and so on. Node's geometry is specified via its shape property. The default shape of new nodes is specified by diagram's defaultShape property. Shapes can be rotated, either interactively using the rotation handle above the node, or programmatically using the rotationAngle property. Shape nodes can display images and text.

Assign an image to be painted inside a node by setting image, imageLocation or imageContent properties. Image position and alignment are set via the imagePadding and imageAlign properties.

Text inside a shape node is represented by the text property. Its appearance can be customized using textColor and font properties. To get even richer text formatting, set enableStyledText to **true** and embed HTML-like formatting tags in the raw text. To make a node big enough to display its text without clipping, call resizeToFitText.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.ShapeNode
      MindFusion.Diagramming.SvgNode

See Also

ShapeNode Members

ShapeNode instances are diagram nodes that represent geometric shapes.

The following tables list the members exposed by the **ShapeNode** type.

Public Constructors

	Name	Description
	ShapeNode	Initializes a new instance of the **ShapeNode** class with default values supplied form the specified Diagram.

Public Properties

	Name	Description
	allowFlip	Gets or sets whether the node's shape flips when users drag an edge over its opposite edge.
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	DiagramNode.anchorPattern override.
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	clipText	Gets or sets a value indicating whether to clip the text assigned to this node to fit into the shape's geometry.
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item.

	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled.
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)

	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	flipX	Gets or sets a value indicating whether the node's shape is fliped horizontally.
	flipY	Gets or sets a value indicating whether the node's shape is fliped vertically.
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	image	Gets or sets the image displayed by this node.
	imageAlign	Gets or sets the alignment of node's image.
	imageContent	Gets or sets a Base64-encoded image to display inside this node.
	imageLocation	Gets or sets URL of the image to display inside this node.
	imagePadding	Gets or sets the padding distance between node borders and image.
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)

	parent	Gets the **Diagram** to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	polygonalTextLayout	Gets or sets a value indicating whether text is wrapped to fit the node's shape.
	rotateImage	Gets or sets a value indicating whether image gets rotated along with the node.
	rotateText	Gets or sets a value indicating whether text gets rotated along with the node.
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the geometric shape displayed by this node.
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)

	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	transparent	Gets or sets a value indicating whether the node's geometry is visible.
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	Checks if the **ShapeNode** contains the specified point.

	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramNode.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets an array of points that define the outline of this node shape.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramNode.LoadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	resizeToFitText	Makes the shape node big enough to display its text without clipping.
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreLocationState	**DiagramItem.restoreLocationState** override.

	restoreState	DiagramNode.restoreState override.
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	DiagramItem.saveLocationState override.
	saveState	DiagramNode.saveState override.
	saveToXml	Saves the node's content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	DiagramItem.updateDrag override.

See Also

ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode Constructor

Initializes a new instance of the ShapeNode class with default values supplied form the specified Diagram.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
function ShapeNode ([parent])
```

Parameters

parent
Optional. The Diagram from which to obtain default values for the node properties.

Remarks

If parent is not specified, this constructor initializes the ShapeNode properties with the values specified in the respective default-value properties of the Diagram object.

If parent is specified, the constructor copies the ShapeNode property values from the specified prototype node.

See Also

ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode Methods

ShapeNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.containsPoint Method

Checks if the ShapeNode contains the specified point.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function containsPoint (point)
```

Parameters

point
The Point to check.

Return Value

true if the **Point** is within the bounds of this node; otherwise, false.

Remarks

This method is used during hit-testing to see if the mouse is inside the item.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.getOutline Method

Gets an array of points that define the outline of this node shape.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function getOutline ()
```

Return Value

Array. An array with the points.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.loadFromXml Method

DiagramNode.LoadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context

XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.resizeToFitText Method

Makes the shape node big enough to display its text without clipping.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
function resizeToFitText (fit)
```

Parameters

fit
One of the FitSize enumeration values.

Return Value

true if the node is resized successfully; otherwise, **false**.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace
text Property
font Property

ShapeNode.restoreLocationState Method

DiagramItem.restoreLocationState override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
function restoreLocationState (ist)
```

Parameters

ist
ShapeNodeState. A ShapeNodeState instance.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.restoreState Method

DiagramNode.restoreState override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.saveLocationState Method

DiagramItem.saveLocationState override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
function saveLocationState ()
```

Return Value

ShapeNodeState. A ShapeNodeState instance.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.saveState Method

DiagramNode.saveState override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function saveState ()
```

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.updateDrag Method

DiagramItem.updateDrag override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
function updateDrag (ist)
```

Parameters

ist
InteractionState. An InteractionState object that encapsulates the current state of user interaction with the control.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode Properties

ShapeNode.allowFlip Property

Gets or sets whether the node's shape flips when users drag an edge over its opposite edge.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
get allowFlip() {}
set allowFlip(value) {}
```

Property Value

Boolean

true to flip the shape, or **false** otherwise.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.anchorPattern Property

DiagramNode.anchorPattern override.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get anchorPattern() {}
set anchorPattern(value) {}
```

Property Value

AnchorPattern

An AnchorPattern object that defines a set of anchor points.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.clipText Property

Gets or sets a value indicating whether to clip the text assigned to this node to fit into the shape's geometry.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get clipText() {}
set clipText(value) {}
```

Property Value

Boolean

true to clip the node's text, or **false** otherwise. The default value is **false**.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace
text Property
rotateText Property

ShapeNode.cssElementName Property

Gets the name of the CSS element associated with this item.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
get cssElementName() {}
set cssElementName(value) {}
```

Property Value

String

A string containing the CSS element name.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.enableStyledText Property

Gets or sets a value indicating whether styled text rendering is enabled.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
get enableStyledText() {}
set enableStyledText(value) {}
```

Property Value

Boolean

true to parse the node's text in order to find formatting tags, or **false** otherwise.

Remarks

Styled text allows applying various attributes to the characters it contains. Attributes are specified using HTML-like formatting tags embedded in the raw text. At this time the component supports the following formatting tags:

- <b> specifies bold text
- <i> specifies italic text
- <u> specifies underlined text
- <color=value> specifies text color
- <sub>specifies subscript
- <sup>specifies superscript
-
 indicates line break
- <a=url> renders specified text as a hyperlink and raises hyperlinkClicked event when link has been clicked
- <size=value>specifies font size

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace
text Property
polygonalTextLayout Property
font Property
textColor Property

ShapeNode.flipX Property

Gets or sets a value indicating whether the node's shape is fliped horizontally.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get flipX() {}
set flipX(value) {}
```

Property Value

Boolean

true if the shape is flipped horizontally, or **false** otherwise.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.flipY Property

Gets or sets a value indicating whether the node's shape is fliped vertically.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get flipY() {}
set flipY(value) {}
```

Property Value

Boolean

true if the shape is flipped vertically, or **false** otherwise.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.image Property

Gets or sets the image displayed by this node.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
get image() {}
set image(value) {}
```

Property Value

HTMLImageElement

An image object to render as node's content.

Remarks

This property lets you display an image inside the node. Set the imageAlign property to specify how the image is aligned inside shape's area. By default, the image is displayed in the center of the bounding rectangle of the node.

If a node should display multiple images, you could add and arrange them using NodeLabel objects.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.imageAlign Property

Gets or sets the alignment of node's image.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

```
JavaScript
get imageAlign() {}
set imageAlign(value) {}
```

Property Value

ImageAlign

A member of the ImageAlign enumeration.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.imageContent Property

Gets or sets a Base64-encoded image to display inside this node.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get imageContent() {}
set imageContent(value) {}
```

Property Value

String

A Base64-encoded representation of an image.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.imageLocation Property

Gets or sets URL of the image to display inside this node.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get imageLocation() {}
set imageLocation(value) {}
```

Property Value

String

A URL string that identifies the image to be loaded.

Remarks

This property lets you display an image inside the node. Set the imageAlign property to specify how the image is aligned inside shape's area. By default, the image is displayed in the center of the bounding rectangle of the node.

If a node should display multiple images, you could add and arrange them using NodeLabel objects.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.imagePadding Property

Gets or sets the padding distance between node borders and image.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get imagePadding() {}
set imagePadding(value) {}
```

Property Value

Thickness

A Thickness instance specifying image padding.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.polygonalTextLayout Property

Gets or sets a value indicating whether text is wrapped to fit the node's shape.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get polygonalTextLayout() {}
set polygonalTextLayout(value) {}
```

Property Value

Boolean

true to wrap the text according to node's geometry, or **false** otherwise.

Remarks

The formatting algorithm is designed to lay out text inside polygonal outlines. A polygonal approximation is created for outlines that contain curved segments, so text can also be laid out relatively to Bézier or spline segments.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace
text Property
enableStyledText Property

ShapeNode.rotateImage Property

Gets or sets a value indicating whether image gets rotated along with the node.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get rotateImage() {}
set rotateImage(value) {}
```

Property Value

Boolean

true to rotate the image, or **false** otherwise.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

ShapeNode.rotateText Property

Gets or sets a value indicating whether text gets rotated along with the node.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get rotateText() {}
set rotateText(value) {}
```

Property Value

Boolean

true to rotate the text, or **false** to keep it horizontal.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace

text Property
clipText Property

ShapeNode.shape Property

Gets or sets the geometric shape displayed by this node.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get shape() {}
set shape(value) {}
```

Property Value

Shape | *String*

The Shape of this node.

See Also

ShapeNode Members
ShapeNode Class
MindFusion.Diagramming Namespace
imageLocation Property
allowFlip Property
rotationAngle Property
cornerRadius Property

ShapeNode.transparent Property

Gets or sets a value indicating whether the node's geometry is visible.

Namespace: MindFusion.Diagramming
File: ShapeNode.js

Syntax

JavaScript

```
get transparent() {}
set transparent(value) {}
```

Property Value

Boolean

true if node's geometry is hidden, or **false** otherwise.

Remarks

If a node is transparent, only its image and text are drawn. Shape, filling and frames are not painted.

See Also

Shapes Class

Provides alternative methods of access to the predefined shapes.

Syntax

JavaScript

```
// class
Shapes.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Shapes

See Also

Shapes Members
MindFusion.Diagramming Namespace

Shapes Members

Provides alternative methods of access to the predefined shapes.

The following tables list the members exposed by the Shapes type.

Public Methods

	Name	Description
	Actor	Gets the Actor predefined shape.
	Alternative	Gets the Alternative predefined shape.
	Arrow	Gets the Arrow predefined shape.
	Arrow1	Gets the Arrow1 predefined shape.
	Arrow2	Gets the Arrow2 predefined shape.
	Arrow3	Gets the Arrow3 predefined shape.
	Arrow4	Gets the Arrow4 predefined shape.
	Arrow5	Gets the Arrow5 predefined shape.
	Arrow6	Gets the Arrow6 predefined shape.
	Arrow7	Gets the Arrow7 predefined shape.
	Arrow8	Gets the Arrow8 predefined shape.
	Arrow9	Gets the Arrow9 predefined shape.

	BackSlash	Gets the BackSlash predefined shape.
	BeginLoop	Gets the BeginLoop predefined shape.
	BowArrow	Gets the BowArrow predefined shape.
	BpmnComplex	Gets the BpmnComplex predefined shape.
	BpmnDataBasedXor	Gets the BpmnDataBasedXor predefined shape.
	BpmnEndCancel	Gets the BpmnEndCancel predefined shape.
	BpmnEndCompensation	Gets the BpmnEndCompensation predefined shape.
	BpmnEndError	Gets the BpmnEndError predefined shape.
	BpmnEndLink	Gets the BpmnEndLink predefined shape.
	BpmnEndMessage	Gets the BpmnEndMessage predefined shape.
	BpmnEndMultiple	Gets the BpmnEndMultiple predefined shape.
	BpmnEndTerminate	Gets the BpmnEndTerminate predefined shape.
	BpmnEventBasedXor	Gets the BpmnEventBasedXor predefined shape.
	BpmnInclusive	Gets the BpmnInclusive predefined shape.
	BpmnIntermediateCancel	Gets the BpmnIntermediateCancel predefined shape.
	BpmnIntermediateCompensation	Gets the BpmnIntermediateCompensation predefined shape.
	BpmnIntermediateError	Gets the BpmnIntermediateError predefined shape.
	BpmnIntermediateLink	Gets the BpmnIntermediateLink predefined shape.
	BpmnIntermediateMessage	Gets the BpmnIntermediateMessage predefined shape.
	BpmnIntermediateMultiple	Gets the BpmnIntermediateMultiple predefined shape.
	BpmnIntermediateRule	Gets the BpmnIntermediateRule predefined shape.
	BpmnIntermediateTimer	Gets the BpmnIntermediateTimer predefined shape.
	BpmnParallelFork	Gets the BpmnParallelFork predefined shape.
	BpmnStartLink	Gets the BpmnStartLink predefined shape.
	BpmnStartMessage	Gets the BpmnStartMessage predefined shape.
	BpmnStartMultiple	Gets the BpmnStartMultiple predefined shape.
	BpmnStartRule	Gets the BpmnStartRule predefined shape.
	BpmnStartTimer	Gets the BpmnStartTimer predefined shape.
	Circle	Gets the Circle predefined shape.
	Cloud	Gets the Cloud predefined shape.
	Collate	Gets the Collate predefined shape.
	ConeDown	Gets the ConeDown predefined shape.
	ConeUp	Gets the ConeUp predefined shape.

	ConnectedIssues	Gets the ConnectedIssues predefined shape.
	CreateRequest	Gets the CreateRequest predefined shape.
	Cross	Gets the Cross predefined shape.
	Cube	Gets the Cube predefined shape.
	Cylinder	Gets the Cylinder predefined shape.
	Database	Gets the Database predefined shape.
	DataTransmition	Gets the DataTransmition predefined shape.
	DDelay	Gets the DDelay predefined shape.
	Decagon	Gets the Decagon predefined shape.
	Decision	Gets the Decision predefined shape.
	Decision2	Gets the Decision2 predefined shape.
	DefaultFlow	Gets the DefaultFlow predefined shape.
	Delay	Gets the Delay predefined shape.
	DInDelay	Gets the DInDelay predefined shape.
	DirectAccessStorage	Gets the DirectAccessStorage predefined shape.
	DiskStorage	Gets the DiskStorage predefined shape.
	Display	Gets the Display predefined shape.
	DividedEvent	Gets the DividedEvent predefined shape.
	DividedProcess	Gets the DividedProcess predefined shape.
	Document	Gets the Document predefined shape.
	Donut	Gets the Donut predefined shape.
	DoubleArrow	Gets the DoubleArrow predefined shape.
	DOutDelay	Gets the DOutDelay predefined shape.
	Ellipse	Gets the Ellipse predefined shape.
	EndLoop	Gets the EndLoop predefined shape.
	ExternalOrganization	Gets the ExternalOrganization predefined shape.
	ExternalProcess	Gets the ExternalProcess predefined shape.
	File	Gets the File predefined shape.
	FramedRectangle	Gets the FramedRectangle predefined shape.
	Gate	Gets the Gate predefined shape.
	Heart	Gets the Heart predefined shape.
	Heptagon	Gets the Heptagon predefined shape.
	Hourglass	Gets the Hourglass predefined shape.

	Input	Gets the Input predefined shape.
	InternalStorage	Gets the InternalStorage predefined shape.
	Interrupt	Gets the Interrupt predefined shape.
	Interrupt2	Gets the Interrupt2 predefined shape.
	IsoProcess	Gets the IsoProcess predefined shape.
	Junction	Gets the Junction predefined shape.
	Lightning	Gets the Lightning predefined shape.
	LinedDocument	Gets the LinedDocument predefined shape.
	ManualOperation	Gets the ManualOperation predefined shape.
	Merge	Gets the Merge predefined shape.
	MessageFromUser	Gets the MessageFromUser predefined shape.
	MessageToUser	Gets the MessageToUser predefined shape.
	Microform	Gets the Microform predefined shape.
	MicroformProcessing	Gets the MicroformProcessing predefined shape.
	MicroformRecording	Gets the MicroformRecording predefined shape.
	MultiDocument	Gets the MultiDocument predefined shape.
	MultiProc	Gets the MultiProc predefined shape.
	Octagon	Gets the Octagon predefined shape.
	OfflineStorage	Gets the OfflineStorage predefined shape.
	OffpageConnection	Gets the OffpageConnection predefined shape.
	OffpageReference	Gets the OffpageReference predefined shape.
	Or	Gets the Or predefined shape.
	Pentagon	Gets the Pentagon predefined shape.
	PentagonArrow	Gets the PentagonArrow predefined shape.
	Plaque	Gets the Plaque predefined shape.
	PointerArrow	Gets the PointerArrow predefined shape.
	PrimitiveFromCall	Gets the PrimitiveFromCall predefined shape.
	PrimitiveToCall	Gets the PrimitiveToCall predefined shape.
	Procedure	Gets the Procedure predefined shape.
	Process	Gets the Process predefined shape.
	Prysm	Gets the Prysm predefined shape.
	PunchedCard	Gets the PunchedCard predefined shape.
	PyramidDown	Gets the PyramidDown predefined shape.

	PyramidUp	Gets the PyramidUp predefined shape.
	Quill	Gets the Quill predefined shape.
	Rectangle	Gets the Rectangle predefined shape.
	Reversed	Gets the Reversed predefined shape.
	RevTriangle	Gets the RevTriangle predefined shape.
	RevWithCirc	Gets the RevWithCirc predefined shape.
	RevWithLine	Gets the RevWithLine predefined shape.
	Rhombus	Gets the Rhombus predefined shape.
	RightTriangle	Gets the RightTriangle predefined shape.
	RoundRect	Gets the RoundRect predefined shape.
	RSave	Gets the RSave predefined shape.
	Save	Gets the Save predefined shape.
	Slash	Gets the Slash predefined shape.
	Sort	Gets the Sort predefined shape.
	Sphere	Gets the Sphere predefined shape.
	Star	Gets the Star predefined shape.
	Star16Pointed	Gets the Star16Pointed predefined shape.
	Star24Pointed	Gets the Star24Pointed predefined shape.
	Star32Pointed	Gets the Star32Pointed predefined shape.
	Star4Pointed	Gets the Star4Pointed predefined shape.
	Star5Pointed	Gets the Star5Pointed predefined shape.
	Star6Pointed	Gets the Star6Pointed predefined shape.
	Star7Pointed	Gets the Star7Pointed predefined shape.
	Start	Gets the Start predefined shape.
	Tape	Gets the Tape predefined shape.
	Terminator	Gets the Terminator predefined shape.
	Tetragon	Gets the Tetragon predefined shape.
	TransmittalTape	Gets the TransmittalTape predefined shape.
	Trapezoid	Gets the Trapezoid predefined shape.
	Triangle	Gets the Triangle predefined shape.

See Also

Shapes Class
MindFusion.Diagramming Namespace

Shapes Methods

Shapes.Actor Method

Gets the Actor predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Actor ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Alternative Method

Gets the Alternative predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Alternative ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow Method

Gets the Arrow predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Arrow ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow1 Method

Gets the Arrow1 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Arrow1 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow2 Method

Gets the Arrow2 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Arrow2 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow3 Method

Gets the Arrow3 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Arrow3 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow4 Method

Gets the Arrow4 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Arrow4 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow5 Method

Gets the Arrow5 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Arrow5 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow6 Method

Gets the Arrow6 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Arrow6 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow7 Method

Gets the Arrow7 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Arrow7 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow8 Method

Gets the Arrow8 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Arrow8 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Arrow9 Method

Gets the Arrow9 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Arrow9 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BackSlash Method

Gets the BackSlash predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BackSlash ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BeginLoop Method

Gets the BeginLoop predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BeginLoop ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BowArrow Method

Gets the BowArrow predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BowArrow ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnComplex Method

Gets the BpmnComplex predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnComplex ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnDataBasedXor Method

Gets the BpmnDataBasedXor predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnDataBasedXor ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEndCancel Method

Gets the BpmnEndCancel predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnEndCancel ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEndCompensation Method

Gets the BpmnEndCompensation predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnEndCompensation ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEndError Method

Gets the BpmnEndError predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
```

```
function BpmnEndError ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEndLink Method

Gets the BpmnEndLink predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnEndLink ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEndMessage Method

Gets the BpmnEndMessage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnEndMessage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class

Shapes.BpmnEndMultiple Method

Gets the BpmnEndMultiple predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnEndMultiple ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEndTerminate Method

Gets the BpmnEndTerminate predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnEndTerminate ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnEventBasedXor Method

Gets the BpmnEventBasedXor predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnEventBasedXor ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnInclusive Method

Gets the BpmnInclusive predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnInclusive ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateCancel Method

Gets the BpmnIntermediateCancel predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnIntermediateCancel ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateCompensation Method

Gets the BpmnIntermediateCompensation predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnIntermediateCompensation ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateError Method

Gets the BpmnIntermediateError predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnIntermediateError ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateLink Method

Gets the BpmnIntermediateLink predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnIntermediateLink ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateMessage Method

Gets the BpmnIntermediateMessage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnIntermediateMessage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateMultiple Method

Gets the BpmnIntermediateMultiple predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnIntermediateMultiple ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateRule Method

Gets the BpmnIntermediateRule predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnIntermediateRule ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnIntermediateTimer Method

Gets the BpmnIntermediateTimer predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnIntermediateTimer ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnParallelFork Method

Gets the BpmnParallelFork predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnParallelFork ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnStartLink Method

Gets the BpmnStartLink predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnStartLink ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnStartMessage Method

Gets the BpmnStartMessage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function BpmnStartMessage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnStartMultiple Method

Gets the BpmnStartMultiple predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnStartMultiple ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnStartRule Method

Gets the BpmnStartRule predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnStartRule ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.BpmnStartTimer Method

Gets the BpmnStartTimer predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function BpmnStartTimer ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Circle Method

Gets the Circle predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Circle ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Cloud Method

Gets the Cloud predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Cloud ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Collate Method

Gets the Collate predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Collate ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.ConeDown Method

Gets the ConeDown predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function ConeDown ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class

Shapes.ConeUp Method

Gets the ConeUp predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function ConeUp ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.ConnectedIssues Method

Gets the ConnectedIssues predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function ConnectedIssues ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.CreateRequest Method

Gets the CreateRequest predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function CreateRequest ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Cross Method

Gets the Cross predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Cross ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Cube Method

Gets the Cube predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Cube ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Cylinder Method

Gets the Cylinder predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Cylinder ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Database Method

Gets the Database predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Database ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DataTransmition Method

Gets the DataTransmition predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DataTransmition ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DDelay Method

Gets the DDelay predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DDelay ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Decagon Method

Gets the Decagon predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Decagon ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Decision Method

Gets the Decision predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Decision ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Decision2 Method

Gets the Decision2 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Decision2 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DefaultFlow Method

Gets the DefaultFlow predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DefaultFlow ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Delay Method

Gets the Delay predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Delay ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DInDelay Method

Gets the DInDelay predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DInDelay ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DirectAccessStorage Method

Gets the DirectAccessStorage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function DirectAccessStorage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DiskStorage Method

Gets the DiskStorage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function DiskStorage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Display Method

Gets the Display predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Display ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DividedEvent Method

Gets the DividedEvent predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DividedEvent ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DividedProcess Method

Gets the DividedProcess predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
```

```
function DividedProcess ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Document Method

Gets the Document predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Document ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Donut Method

Gets the Donut predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Donut ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class

Shapes.DoubleArrow Method

Gets the DoubleArrow predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DoubleArrow ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.DOutDelay Method

Gets the DOutDelay predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function DOutDelay ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Ellipse Method

Gets the Ellipse predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Ellipse ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.EndLoop Method

Gets the EndLoop predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function EndLoop ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.ExternalOrganization Method

Gets the ExternalOrganization predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function ExternalOrganization ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.ExternalProcess Method

Gets the ExternalProcess predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function ExternalProcess ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.File Method

Gets the File predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function File ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.FramedRectangle Method

Gets the FramedRectangle predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function FramedRectangle ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Gate Method

Gets the Gate predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Gate ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Heart Method

Gets the Heart predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Heart ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Heptagon Method

Gets the Heptagon predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Heptagon ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Hourglass Method

Gets the Hourglass predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Hourglass ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Input Method

Gets the Input predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Input ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.InternalStorage Method

Gets the InternalStorage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function InternalStorage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Interrupt Method

Gets the Interrupt predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Interrupt ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Interrupt2 Method

Gets the Interrupt2 predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Interrupt2 ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.IsoProcess Method

Gets the IsoProcess predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function IsoProcess ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Junction Method

Gets the Junction predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Junction ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Lightning Method

Gets the Lightning predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Lightning ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.LinedDocument Method

Gets the LinedDocument predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
```

```
function LinedDocument ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.ManualOperation Method

Gets the ManualOperation predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function ManualOperation ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Merge Method

Gets the Merge predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Merge ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class

Shapes.MessageFromUser Method

Gets the MessageFromUser predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function MessageFromUser ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.MessageToUser Method

Gets the MessageToUser predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function MessageToUser ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Microform Method

Gets the Microform predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Microform ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.MicroformProcessing Method

Gets the MicroformProcessing predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function MicroformProcessing ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.MicroformRecording Method

Gets the MicroformRecording predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function MicroformRecording ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.MultiDocument Method

Gets the MultiDocument predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function MultiDocument ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.MultiProc Method

Gets the MultiProc predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function MultiProc ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Octagon Method

Gets the Octagon predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Octagon ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.OfflineStorage Method

Gets the OfflineStorage predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function OfflineStorage ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.OffpageConnection Method

Gets the OffpageConnection predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function OffpageConnection ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.OffpageReference Method

Gets the OffpageReference predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function OffpageReference ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Or Method

Gets the Or predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Or ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Pentagon Method

Gets the Pentagon predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Pentagon ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.PentagonArrow Method

Gets the PentagonArrow predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function PentagonArrow ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Plaque Method

Gets the Plaque predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Plaque ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.PointerArrow Method

Gets the PointerArrow predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function PointerArrow ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.PrimitiveFromCall Method

Gets the PrimitiveFromCall predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function PrimitiveFromCall ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.PrimitiveToCall Method

Gets the PrimitiveToCall predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function PrimitiveToCall ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Procedure Method

Gets the Procedure predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Procedure ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Process Method

Gets the Process predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Process ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Prysm Method

Gets the Prysm predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Prysm ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.PunchedCard Method

Gets the PunchedCard predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function PunchedCard ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class

Shapes.PyramidDown Method

Gets the PyramidDown predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function PyramidDown ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.PyramidUp Method

Gets the PyramidUp predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function PyramidUp ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Quill Method

Gets the Quill predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Quill ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Rectangle Method

Gets the Rectangle predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Rectangle ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Reversed Method

Gets the Reversed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Reversed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.RevTriangle Method

Gets the RevTriangle predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function RevTriangle ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.RevWithCirc Method

Gets the RevWithCirc predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function RevWithCirc ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.RevWithLine Method

Gets the RevWithLine predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function RevWithLine ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Rhombus Method

Gets the Rhombus predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Rhombus ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.RightTriangle Method

Gets the RightTriangle predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function RightTriangle ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.RoundRect Method

Gets the RoundRect predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function RoundRect ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.RSave Method

Gets the RSave predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function RSave ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Save Method

Gets the Save predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Save ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Slash Method

Gets the Slash predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Slash ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Sort Method

Gets the Sort predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Sort ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Sphere Method

Gets the Sphere predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript
`function Sphere ()`

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star Method

Gets the Star predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript
`function Star ()`

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star16Pointed Method

Gets the Star16Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star16Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star24Pointed Method

Gets the Star24Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star24Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star32Pointed Method

Gets the Star32Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star32Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star4Pointed Method

Gets the Star4Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star4Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star5Pointed Method

Gets the Star5Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star5Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class

Shapes.Star6Pointed Method

Gets the Star6Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star6Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Star7Pointed Method

Gets the Star7Pointed predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Star7Pointed ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Start Method

Gets the Start predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Start ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Tape Method

Gets the Tape predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Tape ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Terminator Method

Gets the Terminator predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

JavaScript

```
function Terminator ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Tetragon Method

Gets the Tetragon predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Tetragon ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.TransmittalTape Method

Gets the TransmittalTape predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function TransmittalTape ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Trapezoid Method

Gets the Trapezoid predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Trapezoid ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

Shapes.Triangle Method

Gets the Triangle predefined shape.

Namespace: MindFusion.Diagramming
File: Shapes.js

Syntax

```
JavaScript
function Triangle ()
```

Return Value

An instance of the Shape class representing the predefined shape.

See Also

Shapes Members
Shapes Class
MindFusion.Diagramming Namespace

ShowAnchors Enumeration

Specifies when anchor points should be displayed.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
ShowAnchors = {}
```

Members

	Member name	Description
	Always	Anchor point marks are always painted.
	Auto	Anchor point marks are shown for the node under the mouse cursor.
	Never	Anchor point marks are never painted.
	Selected	Anchor point marks are painted for the selected node.

Remarks

Members of this enumeration can be assigned to the showAnchors property of Diagram.

See Also

MindFusion.Diagramming Namespace

SimplePanel Class

A layout container that arranges its child components on top of each other.

Syntax

JavaScript

```
// class
SimplePanel.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Container
  MindFusion.Diagramming.Panel
    MindFusion.Diagramming.SimplePanel

      MindFusion.Diagramming.Border

See Also

SimplePanel Members
MindFusion.Diagramming Namespace
Composite Nodes

SimplePanel Members

A layout container that arranges its child components on top of each other.

The following tables list the members exposed by the SimplePanel type.

Public Constructors

	Name	Description
	SimplePanel	Initializes a new instance of the SimplePanel class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	getComponents	Gets the child components of this panel. (Inherited from Panel.)

See Also

SimplePanel Class
MindFusion.Diagramming Namespace

SimplePanel Constructor

Initializes a new instance of the SimplePanel class.

Namespace: MindFusion.Diagramming
File: SimplePanel.js

Syntax

JavaScript
`function SimplePanel ()`

See Also

SimplePanel Class
MindFusion.Diagramming Namespace

SimpleShape Enumeration

Defines the possible values for shape property of tables and containers.

Namespace: MindFusion.Diagramming
File: Enum.js

Syntax

```
JavaScript
// enum
SimpleShape = {}
```

Members

	Member name	Description
	Rectangle	Sets node's shape to rectangle.
	RoundedRectangle	Sets node's shape to rounded rectangle.

See Also

MindFusion.Diagramming Namespace

SinglePointerController Class

Processes mouse or single-touch events targeting a diagram element.

Syntax

```
JavaScript
// class
SinglePointerController.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SinglePointerController
MindFusion.Diagramming.CreateLinkController
MindFusion.Diagramming.CreateNodeController
MindFusion.Diagramming.CreateSelectionController
MindFusion.Diagramming.ModifyLinkController
MindFusion.Diagramming.ModifyNodeController
MindFusion.Diagramming.ModifySelectionController
MindFusion.Diagramming.PanController

See Also

SinglePointerController Members
MindFusion.Diagramming Namespace

SinglePointerController Members

Processes mouse or single-touch events targeting a diagram element.

The following tables list the members exposed by the SinglePointerController type.

Public Constructors

	Name	Description
	SinglePointerController	Initializes a new instance of the SinglePointerController class.

Public Methods

	Name	Description
	cancel	Called when the user stops drawing or modifying a diagram element, rejecting the operation.
	commit	Called when the user stops drawing or modifying a diagram element, accepting the operation.
	drawInteraction	Called to draw current state of user interaction.
	move	Called continuously while the user draws with the mouse or touch.
	start	Called when the user starts drawing or modifying a diagram element.
	validate	Called to validate the operation at current pointer position.

See Also

SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController Constructor

Initializes a new instance of the SinglePointerController class.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

JavaScript
`function SinglePointerController (modifiedItem, adjustmentHandle)`

Parameters

modifiedItem
DiagramItem. The item that will be modified by this controller.

adjustmentHandle
Object. An object representing the adjustment handle used to modify the item.

See Also

SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController Methods

SinglePointerController.cancel Method

Called when the user stops drawing or modifying a diagram element, rejecting the operation.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

JavaScript

```
function cancel (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

SinglePointerController Members
SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController.commit Method

Called when the user stops drawing or modifying a diagram element, accepting the operation.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

JavaScript

```
function commit (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

SinglePointerController Members
SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController.drawInteraction Method

Called to draw current state of user interaction.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

```
JavaScript
function drawInteraction (context)
```

Parameters

context
DrawingContext2D. The drawing context for diagram's underlying canvas.

See Also

SinglePointerController Members
SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController.move Method

Called continuously while the user draws with the mouse or touch.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

```
JavaScript
function move (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

SinglePointerController Members
SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController.start Method

Called when the user starts drawing or modifying a diagram element.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

```
JavaScript
function start (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

See Also

SinglePointerController Members
SinglePointerController Class
MindFusion.Diagramming Namespace

SinglePointerController.validate Method

Called to validate the operation at current pointer position.

Namespace: MindFusion.Diagramming
File: SinglePointerController.js

Syntax

JavaScript

```
function validate (position)
```

Parameters

position
Point. A point specified in diagram coordinates.

Return Value

Boolean. true to accept the operation, or false otherwise.

See Also

SinglePointerController Members
SinglePointerController Class
MindFusion.Diagramming Namespace

StackPanel Class

A layout container that arranges its child components stacked horizontally or vertically.

Syntax

JavaScript

```
// class
StackPanel.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Container
  MindFusion.Diagramming.Panel
    MindFusion.Diagramming.StackPanel

See Also

StackPanel Members

A layout container that arranges its child components stacked horizontally or vertically.

The following tables list the members exposed by the StackPanel type.

Public Constructors

	Name	Description
	StackPanel	Initializes a new instance of the **StackPanel** class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	orientation	Gets or sets the stack orientation.
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	getComponents	Gets the child components of this panel. (Inherited from Panel.)
	getOrientation	Gets the stack orientation.
	setOrientation	Sets the stack orientation.

See Also

StackPanel Class
MindFusion.Diagramming Namespace

StackPanel Constructor

Initializes a new instance of the StackPanel class.

Namespace: MindFusion.Diagramming
File: StackPanel.js

Syntax

```
JavaScript
function StackPanel ()
```

See Also

StackPanel Class
MindFusion.Diagramming Namespace

StackPanel Methods

StackPanel.getOrientation Method

Gets the stack orientation.

Namespace: MindFusion.Diagramming
File: StackPanel.js

Syntax

```
JavaScript
function getOrientation ()
```

Return Value

A member of the Orientation enumeration.

See Also

StackPanel Members
StackPanel Class
MindFusion.Diagramming Namespace

StackPanel.setOrientation Method

Sets the stack orientation.

Namespace: MindFusion.Diagramming
File: StackPanel.js

Syntax

```
JavaScript
function setOrientation (value)
```

Parameters

value

A member of the Orientation enumeration.

See Also

StackPanel Members
StackPanel Class
MindFusion.Diagramming Namespace

StackPanel Properties

StackPanel.orientation Property

Gets or sets the stack orientation.

Namespace: MindFusion.Diagramming
File: StackPanel.js

Syntax

JavaScript

```
get orientation() {}
set orientation(value) {}
```

Property Value

Orientation

A member of the Orientation enumeration.

See Also

StackPanel Members
StackPanel Class
MindFusion.Diagramming Namespace

Style Class

Represents a set of appearance properties whose values can be inherited from parent objects if not set locally for an item.

Syntax

JavaScript

```
// class
Style.prototype = {}
```

Remarks

An instance of this class can be used to set items' appearance properties as a single unit via their style property. In order for style attributes to be used, the respective local appearance properties of a DiagramItem objects must have undefined or **null** values. If an item's **style** does not define a value for some attribute, it is looked up in the styling hierarchy, first in the diagram's style, next in the current theme, and finally in the set of default styles defined for each item class.

Inheritance Hierarchy

MindFusion.Diagramming.Style

See Also

Style Members
MindFusion.Diagramming Namespace

Style Members

Represents a set of appearance properties whose values can be inherited from parent objects if not set locally for an item.

The following tables list the members exposed by the Style type.

Public Constructors

	Name	Description
	Style	Initializes a new instance of the Style class.

Public Properties

	Name	Description
	backBrush	Gets or sets the brush used to paint diagram background.
	brush	Gets or sets the brush used to fill the interior of diagram items.
	captionBackBrush	Gets or sets the brush used to fill nodes' caption bars.
	fontName	Gets or sets the font name used to draw text of diagram items.
	fontSize	Gets or sets the font size used to draw text of diagram items.
	fontStyle	Gets or sets the font style used to draw text of diagram items.
	nodeEffects	Gets or sets a list of effects applied to diagram nodes.
	selectedBrush	Gets or sets the brush used to paint the background of selected items.
	selectedStroke	Gets or sets the color used to stroke outlines of selected items.
	shadowColor	Gets or sets the color used to draw shadows of diagram items.
	stroke	Gets or sets the color used to stroke outlines of diagram items.
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking outlines of diagram items.
	strokeThickness	Gets or sets the line width applied when stroking outlines of diagram items.

	textColor	Gets or sets the color used to draw text of diagram items.

See Also

Style Class
MindFusion.Diagramming Namespace

Style Constructor

Initializes a new instance of the Style class.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

JavaScript

```
function Style ([prototype])
```

Parameters

prototype
Optional.

See Also

Style Class
MindFusion.Diagramming Namespace

Style Properties

Style.backBrush Property

Gets or sets the brush used to paint diagram background.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

JavaScript

```
get backBrush() {}
set backBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Style Members
Style Class

Style.brush Property

Gets or sets the brush used to fill the interior of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.captionBackBrush Property

Gets or sets the brush used to fill nodes' caption bars.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get captionBackBrush() {}
set captionBackBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

Remarks

This value is applied to TableNode and ContainerNode objects.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.fontName Property

Gets or sets the font name used to draw text of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get fontName() {}
set fontName(value) {}
```

Property Value

String

A string containing font name.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.fontSize Property

Gets or sets the font size used to draw text of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get fontSize() {}
set fontSize(value) {}
```

Property Value

Number

A number specifying font size.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.fontStyle Property

Gets or sets the font style used to draw text of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get fontStyle() {}
set fontStyle(value) {}
```

Property Value

FontStyle

A combination of FontStyle enumeration members.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.nodeEffects Property

Gets or sets a list of effects applied to diagram nodes.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get nodeEffects() {}
set nodeEffects(value) {}
```

Property Value

Array

An array of NodeEffect objects.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.selectedBrush Property

Gets or sets the brush used to paint the background of selected items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get selectedBrush() {}
set selectedBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.selectedStroke Property

Gets or sets the color used to stroke outlines of selected items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

JavaScript

```
get selectedStroke() {}
set selectedStroke(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.shadowColor Property

Gets or sets the color used to draw shadows of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

JavaScript

```
get shadowColor() {}
set shadowColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.stroke Property

Gets or sets the color used to stroke outlines of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get stroke() {}
set stroke(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.strokeDashStyle Property

Gets or sets the line dash pattern applied when stroking outlines of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get strokeDashStyle() {}
set strokeDashStyle(value) {}
```

Property Value

DashStyle

A member of the DashStyle enumeration.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.strokeThickness Property

Gets or sets the line width applied when stroking outlines of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get strokeThickness() {}
set strokeThickness(value) {}
```

Property Value

Number

A number specifying stroke thickness.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

Style.textColor Property

Gets or sets the color used to draw text of diagram items.

Namespace: MindFusion.Diagramming
File: Style.js

Syntax

```
JavaScript
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming Namespace

SvgContent Class

SvgContent represents the markup code of an SVG drawing.

Syntax

JavaScript
`// class` `SvgContent.prototype = {}`

Remarks

SvgContent objects are assigned to the content property of SvgNode.

Inheritance Hierarchy

MindFusion.Diagramming.SvgContent

See Also

SvgContent Members
MindFusion.Diagramming Namespace

SvgContent Members

SvgContent represents the markup code of an SVG drawing.

The following tables list the members exposed by the **SvgContent** type.

Public Methods

	Name	Description
	parse	Loads SVG code from the specified file.
	parseSvg	Loads SVG code from the specified string.

See Also

SvgContent Class
MindFusion.Diagramming Namespace

SvgContent Methods

SvgContent.parse Method

Loads SVG code from the specified file.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript
`function parse (value)`

Parameters

value

String. A string specifying the SVG file location.

Remarks

The MIME type of .svg files on server should be set to image/svg+xml.

See Also

SvgContent Members
SvgContent Class
MindFusion.Diagramming Namespace

SvgContent.parseSvg Method

Loads SVG code from the specified string.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript

```
function parseSvg (value)
```

Parameters

value
String. A string containing the SVG markup code.

See Also

SvgContent Members
SvgContent Class
MindFusion.Diagramming Namespace

SvgExporter Class

Creates SVG (Scalable Vector Graphics) drawings from the content of Diagram objects.

Syntax

JavaScript

```
// class
SvgExporter.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.SvgExporter

See Also

SvgExporter Members

Creates SVG (Scalable Vector Graphics) drawings from the content of Diagram objects.

The following tables list the members exposed by the SvgExporter type.

Public Constructors

	Name	Description
	SvgExporter	Initializes a new instance of the **SvgExporter** class.

Public Methods

	Name	Description
	exportElement	Overloaded.
	exportString	Creates an SVG document representing the specified diagram.

See Also

SvgExporter Class
MindFusion.Diagramming Namespace

SvgExporter Constructor

Initializes a new instance of the SvgExporter class.

Namespace: MindFusion.Diagramming
File: SvgExporter.js

Syntax

JavaScript
`function SvgExporter ()`

See Also

SvgExporter Class
MindFusion.Diagramming Namespace

SvgExporter Methods

SvgExporter.exportElement Method

Overload List

Name	Description
SvgExporter.exportElement	Creates an SVG element representing the specified diagram.
SvgExporter.exportElement	Creates an SVG element representing the specified diagram.

See Also

SvgExporter Members
SvgExporter Class
MindFusion.Diagramming Namespace

SvgExporter.exportElement Method

Creates an SVG element representing the specified diagram.

Namespace: MindFusion.Diagramming
File: SvgExporter.js

Syntax

JavaScript

```
function exportElement (diagram)
```

Parameters

diagram
A reference to the diagram to be exported.

See Also

exportElement Method Overload List
SvgExporter Members
SvgExporter Class
MindFusion.Diagramming Namespace

SvgExporter.exportElement Method

Creates an SVG element representing the specified diagram.

Namespace: MindFusion.Diagramming
File: SvgExporter.js

Syntax

JavaScript

```
function exportElement (diagram, [parentDoc])
```

Parameters

diagram
Diagram. A reference to the diagram to be exported.

parentDoc
Optional.

Document. The document to export to.

Return Value

Object. A DOM SVG element.

See Also

exportElement Method Overload List
SvgExporter Members
SvgExporter Class
MindFusion.Diagramming Namespace

SvgExporter.exportString Method

Creates an SVG document representing the specified diagram.

Namespace: MindFusion.Diagramming
File: SvgExporter.js

Syntax

JavaScript

```
function exportString (diagram)
```

Parameters

diagram
Diagram. A reference to the diagram to be exported.

Return Value

Object. A string containing the SVG document's content.

See Also

SvgExporter Members
SvgExporter Class
MindFusion.Diagramming Namespace

SvgNode Class

SvgNode instances are diagram nodes that can render SVG drawings.

Syntax

JavaScript

```
// class
SvgNode.prototype = {}
```

Remarks

An **SvgNode** instance represents nodes that can display SVG drawings in addition to the rendering provided by the base ShapeNode class. The SVG drawing of a node is specified through its content

property. The SVG graphics are loaded and associated with the node through an SvgContent object, which should be initialized and then assigned to **content**. Note that the MIME type of .svg files on server should be set to image/svg+xml.

Example

The following code illustrates how to load SVG graphics from a server file and associate it with a new **SvgNode**:

JavaScript

```
var node = diagram.factory.createSvgNode(bounds);
node.transparent = true; // hide ShapeNode geometry

var svg = new SvgContent();
svg.parse("image.svg");
node.content = svg;
```

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.ShapeNode
      MindFusion.Diagramming.SvgNode

See Also

SvgNode Members
MindFusion.Diagramming Namespace

SvgNode Members

SvgNode instances are diagram nodes that can render SVG drawings.

The following tables list the members exposed by the **SvgNode** type.

Public Constructors

	Name	Description
	SvgNode	Initializes a new instance of the SvgNode class with default values supplied from the specified Diagram.

Public Properties

	Name	Description
	allowFlip	Gets or sets whether the node's shape flips when users drag an edge over its opposite edge. (Inherited from ShapeNode.)
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	DiagramNode.anchorPattern override. (Inherited from ShapeNode.)

	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	clipText	Gets or sets a value indicating whether to clip the text assigned to this node to fit into the shape's geometry. (Inherited from ShapeNode.)
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	content	Gets or sets an SvgContent instance representing the SVG drawing rendered in this node.
	contentAlign	Gets or sets the alignment of the **SvgContent** displayed in this node.
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from ShapeNode.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)

	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled. (Inherited from ShapeNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	flipX	Gets or sets a value indicating whether the node's shape is fliped horizontally. (Inherited from ShapeNode.)
	flipY	Gets or sets a value indicating whether the node's shape is fliped vertically. (Inherited from ShapeNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)

	image	Gets or sets the image displayed by this node. (Inherited from ShapeNode.)
	imageAlign	Gets or sets the alignment of node's image. (Inherited from ShapeNode.)
	imageContent	Gets or sets a Base64-encoded image to display inside this node. (Inherited from ShapeNode.)
	imageLocation	Gets or sets URL of the image to display inside this node. (Inherited from ShapeNode.)
	imagePadding	Gets or sets the padding distance between node borders and image. (Inherited from ShapeNode.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	polygonalTextLayout	Gets or sets a value indicating whether text is wrapped to fit the node's shape. (Inherited from ShapeNode.)
	rotateImage	Gets or sets a value indicating whether image gets rotated along with the node. (Inherited from ShapeNode.)
	rotateText	Gets or sets a value indicating whether text gets rotated along with the node. (Inherited from ShapeNode.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)

	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the geometric shape displayed by this node. (Inherited from ShapeNode.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	transparent	Gets or sets a value indicating whether the node's geometry is visible. (Inherited from ShapeNode.)

	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	ShapeNode.clone override.
	containsPoint	Checks if the **ShapeNode** contains the specified point. (Inherited from ShapeNode.)
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	fromJson	ShapeNode.fromJson override.
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)

	getOutline	Gets an array of points that define the outline of this node shape. (Inherited from ShapeNode.)
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	init	Returns a Builder object used to configure this **SvgNode** instance.
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	loadFromXml	DiagramNode.LoadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	resizeToFitText	Makes the shape node big enough to display its text without clipping. (Inherited from ShapeNode.)
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreLocationState	**DiagramItem.restoreLocationState** override. (Inherited from ShapeNode.)
	restoreState	DiagramNode.restoreState override. (Inherited from ShapeNode.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	DiagramItem.saveLocationState override. (Inherited from ShapeNode.)
	saveState	DiagramNode.saveState override. (Inherited from ShapeNode.)
	saveToXml	Saves the node's content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)

	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	ShapeNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	DiagramItem.updateDrag override. (Inherited from ShapeNode.)
S	With	Returns a Builder object used to configure and create new **SvgNode** instances.

See Also

SvgNode Class
MindFusion.Diagramming Namespace

SvgNode Constructor

Initializes a new instance of the SvgNode class with default values supplied from the specified Diagram.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript

```
function SvgNode ([parent])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the node properties.

See Also

SvgNode Class
MindFusion.Diagramming Namespace

SvgNode Methods

SvgNode.clone Method

ShapeNode.clone override.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript

```
function clone ()
```

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode.fromJson Method

ShapeNode.fromJson override.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode.init Method

Returns a Builder object used to configure this SvgNode instance.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript

```
function init ()
```

Return Value

A Builder instance.

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode.loadFromXml Method

DiagramNode.LoadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode.toJson Method

ShapeNode.toJson override.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode.With Method

Returns a Builder object used to configure and create new SvgNode instances.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

```
JavaScript
function With (diagram)
```

Parameters

diagram

Return Value

A Builder instance.

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

SvgNode Properties

SvgNode.content Property

Gets or sets an SvgContent instance representing the SVG drawing rendered in this node.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

```
JavaScript
get content() {}
set content(value) {}
```

Property Value

SvgContent

An instance of the SvgContent class.

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace
contentAlign Property

SvgNode.contentAlign Property

Gets or sets the alignment of the SvgContent displayed in this node.

Namespace: MindFusion.Diagramming
File: SvgNode.js

Syntax

JavaScript

```
get contentAlign() {}
set contentAlign(value) {}
```

Property Value

ImageAlign

A member of the ImageAlign enumeration.

See Also

SvgNode Members
SvgNode Class
MindFusion.Diagramming Namespace

TableColumn Class

Represents a column of a TableNode.

Syntax

JavaScript

```
// class
TableColumn.prototype = {}
```

Remarks

Call the getColumn method of TableNode to get the **TableColumn** instance at specified position. In order to add or remove columns, set table's columnCount property, or call methods such as addColumn, insertColumn, deleteColumn, redimTable.

Inheritance Hierarchy

MindFusion.Diagramming.TableColumn

See Also

TableColumn Members
MindFusion.Diagramming Namespace
TableRow Class

TableColumn Members

Represents a column of a TableNode.

The following tables list the members exposed by the TableColumn type.

Public Constructors

	Name	Description
	TableColumn	Initializes a new instance of the TableColumn class.

Public Properties

	Name	Description
	columnStyle	Gets or sets the column style.
	width	Gets or sets the width of this column.

See Also

TableColumn Class
MindFusion.Diagramming Namespace

TableColumn Constructor

Initializes a new instance of the TableColumn class.

Namespace: MindFusion.Diagramming
File: TableColumn.js

Syntax

JavaScript

```
function TableColumn ()
```

See Also

TableColumn Class
MindFusion.Diagramming Namespace

TableColumn Properties

TableColumn.columnStyle Property

Gets or sets the column style.

Namespace: MindFusion.Diagramming
File: TableColumn.js

Syntax

```
JavaScript
get columnStyle() {}
set columnStyle(value) {}
```

Property Value

ColumnStyle

A member of the ColumnStyle enumeration.

See Also

TableColumn Members
TableColumn Class
MindFusion.Diagramming Namespace

TableColumn.width Property

Gets or sets the width of this column.

Namespace: MindFusion.Diagramming
File: TableColumn.js

Syntax

```
JavaScript
get width() {}
set width(value) {}
```

Property Value

Number

A number specifying the width.

See Also

TableColumn Members
TableColumn Class
MindFusion.Diagramming Namespace

TableConnectionPoint Class

A class that represents a connection between an edge and a node in a graph.

Syntax

JavaScript

```
// class
TableConnectionPoint.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ConnectionPoint
  MindFusion.Diagramming.TableConnectionPoint

See Also

TableConnectionPoint Members
MindFusion.Diagramming Namespace

TableConnectionPoint Members

A class that represents a connection between an edge and a node in a graph.

The following tables list the members exposed by the TableConnectionPoint type.

Public Properties

	Name	Description
	endPoint	Calculates the coordinates of this connection point expressed in document coordinates.

Public Methods

	Name	Description
	addLinkToNode	Adds the link associated with this connection point to the associated node.
	getInitialPoint	Returns a point (usually the center of the bounding rectangle) contained within the associated node.
	getIntersection	Calculates the intersection point between the specified line segment and the contour of the node associated with this connection point.
	getNearestAnchorPoint	Returns the nearest anchor Point pertinent to this connection. (Inherited from ConnectionPoint.)
	getNearestBorderPoint	Returns the nearest point lying on the node border.
	getNodeRect	Returns the bounding rectangle of the node associated with this connection point. (Inherited from ConnectionPoint.)
	linkChanges	Checks whether the link would change if attached to the specified node at the specified point.
	nodesIntersect	Checks whether the node associated with the current connection point and the node associated with the specified connection point intersect. (Inherited from ConnectionPoint.)

	removeLinkFromNode	Removes the link associated with this connection point from the associated node.
	saveEndRelative	Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node. (Inherited from ConnectionPoint.)
	saveEndRelative	Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node.

See Also

TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint Methods

TableConnectionPoint.addLinkToNode Method

Adds the link associated with this connection point to the associated node.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function addLinkToNode ()
```

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint.getInitialPoint Method

Returns a point (usually the center of the bounding rectangle) contained within the associated node.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function getInitialPoint ()
```

Return Value

Point. A Point contained within the node.

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint.getIntersection Method

Calculates the intersection point between the specified line segment and the contour of the node associated with this connection point.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function getIntersection (point1, point2)
```

Parameters

point1
Point. The first point of the segment.

point2
Point. The second point of the segment.

Return Value

Point. A Point object representing the intersection point.

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint.getNearestBorderPoint Method

Returns the nearest point lying on the node border.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function getNearestBorderPoint (point)
```

Parameters

point
Point. A Point instance specifying location in the diagram.

Return Value

AnchorPointDetails. An AnchorPointDetails specifying the anchor point properties.

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint.linkChanges Method

Checks whether the link would change if attached to the specified node at the specified point.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function linkChanges (item, point)
```

Parameters

item
DiagramItem. A DiagramItem under the mouse cursor.

point
Point. A point where the link will be connected.

Return Value

Boolean. true if the link would connect to a different entity if dropped at the specified location; otherwise, false.

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint.removeLinkFromNode Method

Removes the link associated with this connection point from the associated node.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function removeLinkFromNode ()
```

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint.saveEndRelative Method

Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
function saveEndRelative ()
```

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableConnectionPoint Properties

TableConnectionPoint.endPoint Property

Calculates the coordinates of this connection point expressed in document coordinates.

Namespace: MindFusion.Diagramming
File: TableConnectionPoint.js

Syntax

JavaScript

```
get endPoint() {}
set endPoint(value) {}
```

Property Value

Point

A Point that represents the coordinate.

See Also

TableConnectionPoint Members
TableConnectionPoint Class
MindFusion.Diagramming Namespace

TableNode Class

TableNode instances are diagram nodes that can be used to display tabular or list data.

Syntax

JavaScript

```
// class
TableNode.prototype = {}
```

Remarks

Tables consist of Cell instances distributed across rows and columns. The number of rows and columns can be set by calling the redimTable method. Cells can be made to span several rows and columns by setting their rowSpan and columnSpan properties to values larger than one.

Rows can be assigned to distinct sections of the table. Each section can be collapsed or expanded, hiding or showing all rows in the section except the header one. To define a section, set a header row. A section consists of all rows after a header row and spans to the next header. Each header row displays a [±] button that allows expanding or collapsing the section interactively. The button appears on the left of the first cell of a header row. Clicking that button raises the tableSectionCollapsed and tableSectionExpanded events. Sections can be collapsed or expanded programmatically by setting the expanded property of their header rows.

The structure of a table can be dynamically changed by adding and deleting rows and columns. This can be done with the addColumn and addRow methods as well as with the insertColumn, insertRow and **redimTable** methods. deleteRow and deleteColumn methods delete existing rows and columns.

Table cells can display text as specified via their text, textColor and textAlignment properties. Cells can also contain images, as set via the image and imageAlign properties.

Table nodes can show a caption in their title bar, whose text is set via the text property inherited from DiagramItem. The height of the caption bar is set via the captionHeight property.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.TableNode

See Also

TableNode Members
MindFusion.Diagramming Namespace

TableNode Members

TableNode instances are diagram nodes that can be used to display tabular or list data.

The following tables list the members exposed by the **TableNode** type.

Public Constructors

	Name	Description
	TableNode	Initializes a new instance of the TableNode class with default values supplied from the specified Diagram.

Public Fields

	Name	Description
	useScrollBars	Set this flag to show full scrollbars for scrollable tables instead of up/down buttons in captions.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	allowResizeColumns	Gets or sets a value indicating whether users are allowed to resize table columns.
	allowResizeRows	Gets or sets a value indicating whether users are allowed to resize table rows.
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	captionBackBrush	Gets or sets an object specifying how to fill the caption bar.
	captionFont	Gets or sets the font used to render the node's caption text.
	captionHeight	Gets or sets the height of the node's caption area.
	cellFrameStyle	Gets or sets the style of cells' frame lines.
	cellStyle	Gets or sets the style used to draw table's cells.
	cellTextColor	Gets or sets the color used to fill cells text.
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	columnCount	Gets or sets the number of columns in the table.
	connectionStyle	Gets or sets whether drawing a link from/to this node should connectan individual table row, or the whole table as an integral entity.
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)

	currentScrollRow	Gets or sets the table's current scroll position.
	dividerStroke	Gets or sets the color used to stroke the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	dividerStrokeDashArray	Gets or sets an array of values specifying the dash pattern of the node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from CaptionedNode.)
	dividerStrokeDashStyle	Gets or sets the line dash pattern applied when stroking node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeThickness	Gets or sets the width of the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveDividerStrokeThickness	Gets the effective line thickness that should be applied when stroking this node's divider line. (Inherited from CaptionedNode.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)

	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled.
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)

	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	rowAnchorPattern	Gets or sets the default anchor pattern for the rows of a table.
	rowCount	Gets or sets the number of rows in the table.
	scrollable	Gets or sets whether the user is allowed to scroll the table rows.
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the shape of the node's outline.
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)

	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addColumn	Adds a new column to the table.
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	addRow	Adds a new row to the table.
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)

	cellFromPoint	Returns the cell that contains the specified point.
	clone	DiagramNode.clone override.
	containsPoint	Checks if the **TableNode** contains the specified point.
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point.
	deleteColumn	Deletes the specified column.
	deleteRow	Deletes the specified row.
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramNode.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getCell	Returns a reference to the cell located at the specified column and row of this table.
	getColumn	Returns a reference to the column at the specified position within the table's list of columns.
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets an array of points that define the outline of this node shape.
	getParentRow	Returns the number of the first header row before thespecified row. If the specified row is a header rowitself, returns 'row'. If there is no header rowbefore this one, the return value is -1.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
	getRow	Returns a reference to the row at the specified position within the table's list of rows.
	getRowRect	Returns the bounding rectangle of the specified row.
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	insertColumn	Inserts a new column at the specified position.
	insertRow	Inserts a new row at the specified position.

	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
	isRowCollapsed	Checks whether the specified row is collapsed.Collapsed are all rows whose header row is collapsed.
	loadFromXml	DiagramNode.loadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	redimTable	Changes the number of columns and rows in this table.
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	resizeToFitText	Resizes the columns and rows so that cells are large enough to fit their text.
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	restoreState	DiagramNode.restoreState override.
	rowFromPoint	Returns the index of the table's row that contains the specified point.
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramNode.saveState override.
	saveToXml	Saves the node's content into an XML element.
	scrollDown	Scrolls the table's rows down.
	scrollUp	Scrolls the table's rows up.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

See Also

TableNode Class
MindFusion.Diagramming Namespace

TableNode Fields

TableNode.useScrollBars Field

Set this flag to show full scrollbars for scrollable tables instead of up/down buttons in captions.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
var useScrollBars
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode Constructor

Initializes a new instance of the TableNode class with default values supplied from the specified Diagram.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function TableNode ([parent])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the node properties.

See Also

TableNode Class
MindFusion.Diagramming Namespace

TableNode Methods

TableNode.addColumn Method

Adds a new column to the table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function addColumn ()
```

Return Value

Object. The newly created column object.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
deleteColumn Method
addRow Method
redimTable Method

TableNode.addRow Method

Adds a new row to the table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function addRow ()
```

Return Value

Object. The newly created row object.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
deleteRow Method
addColumn Method
redimTable Method

TableNode.cellFromPoint Method

Returns the cell that contains the specified point.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function cellFromPoint (point)
```

Parameters

point
A Point instance.

Return Value

A JavaScript object containing following fields:
{
cell: Cell reference,
cellRect: Rect,
column: Number,
row: Number
};

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.containsPoint Method

Checks if the TableNode contains the specified point.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
```

```
function containsPoint (point)
```

Parameters

point
The Point to check.

Return Value

true if the point is within the bounds of this node, or **false** otherwise.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.createConnectionPoint Method

Creates a connection point between this node and the specified link at the specified point.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function createConnectionPoint (link, point, incoming)
```

Parameters

link
DiagramLink. A DiagramLink that should be connected to this node.

point
Point. The point at which the link should be connected.

incoming
Boolean. true if link is an incoming link, otherwise, false.

Return Value

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.deleteColumn Method

Deletes the specified column.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function deleteColumn (col)
```

Parameters

col
Number. An integer value specifying which column to delete.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.deleteRow Method

Deletes the specified row.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function deleteRow (row)
```

Parameters

row
Number. An integer value specifying which row to delete.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function fromJson ()
```

See Also

TableNode Members
TableNode Class

TableNode.getCell Method

Returns a reference to the cell located at the specified column and row of this table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function getCell (col, row)
```

Parameters

col
Number. An integer value specifying the index of a table's column.

row
Number. An integer value specifying the index of a table's row.

Return Value

A Cell instance representing the specified cell.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.getColumn Method

Returns a reference to the column at the specified position within the table's list of columns.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function getColumn (col)
```

Parameters

col
Number. An integer value specifying the index of a table's column.

Return Value

A TableColumn instance.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
columnCount Property

TableNode.getOutline Method

Gets an array of points that define the outline of this node shape.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function getOutline ()
```

Return Value

Array. An array with the points.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.getParentRow Method

Returns the number of the first header row before thespecified row. If the specified row is a header rowitself, returns 'row'. If there is no header rowbefore this one, the return value is -1.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function getParentRow ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.getRow Method

Returns a reference to the row at the specified position within the table's list of rows.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function getRow (row)
```

Parameters

row
Number. An integer value specifying the index of a table's row.

Return Value

A TableRow instance.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
rowCount Property

TableNode.getRowRect Method

Returns the bounding rectangle of the specified row.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function getRowRect (row)
```

Parameters

row
Number. An integer value specifying the index of a table's row.

Return Value

Rect. A Rect object representing the row's bounding rectangle.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.insertColumn Method

Inserts a new column at the specified position.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function insertColumn (col)
```

Parameters

col
Number. An integer value specifying the position within the table's list of columns where the new column should be inserted.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
columnCount Property

TableNode.insertRow Method

Inserts a new row at the specified position.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function insertRow (row)
```

Parameters

row
Number. An integer value specifying the position within the table's list of rows where the new row should be inserted.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
rowCount Property

TableNode.isRowCollapsed Method

Checks whether the specified row is collapsed.Collapsed are all rows whose header row is collapsed.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function isRowCollapsed ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.loadFromXml Method

DiagramNode.loadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.redimTable Method

Changes the number of columns and rows in this table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function redimTable (columns, rows)
```

Parameters

columns
Number. An integer value specifying the new number of columns.

rows
Number. An integer value specifying the new number of rows.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.resizeToFitText Method

Resizes the columns and rows so that cells are large enough to fit their text.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function resizeToFitText (ignoreCaption, keepCellWidth)
```

Parameters

ignoreCaption
true to ignore caption text and consider only cells for table size, or **false** otherwise.

keepCellWidth
true to preserve column widths and resize only row heights, or **false** otherwise.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.restoreState Method

DiagramNode.restoreState override.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function restoreState ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.rowFromPoint Method

Returns the index of the table's row that contains the specified point.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function rowFromPoint (point)
```

Parameters

point
Point. A Point instance.

Return Value

Number. An integer value specifying the row's index.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.saveState Method

DiagramNode.saveState override.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function saveState ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement

Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.scrollDown Method

Scrolls the table's rows down.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function scrollDown ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
currentScrollRow Property

TableNode.scrollUp Method

Scrolls the table's rows up.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
function scrollUp ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
currentScrollRow Property

TableNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode Properties

TableNode.allowResizeColumns Property

Gets or sets a value indicating whether users are allowed to resize table columns.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
get allowResizeColumns() {}
set allowResizeColumns(value) {}
```

Property Value

Boolean

true if users are allowed to resize table columns, or **false** otherwise.

Remarks

A column can be resized by dragging its right border. If the columnStyle field of a column is set to AutoWidth, it is automatically reset to **FixedWidth** when the column is resized. Rightmost columns always fill all remaining space in their tables and cannot be resized.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
tableColumnResized Event
tableColumnResizing Event

TableNode.allowResizeRows Property

Gets or sets a value indicating whether users are allowed to resize table rows.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
get allowResizeRows() {}
set allowResizeRows(value) {}
```

Property Value

Boolean

true if users are allowed to resize table rows, or **false** otherwise.

Remarks

A row can be resized by dragging its bottom border.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace
tableRowResized Event
tableRowResizing Event

TableNode.captionBackBrush Property

Gets or sets an object specifying how to fill the caption bar.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
get captionBackBrush() {}
set captionBackBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.captionFont Property

Gets or sets the font used to render the node's caption text.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
get captionFont() {}
set captionFont(value) {}
```

Property Value

Font

A Font object specifying the font of the caption bar.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.captionHeight Property

Gets or sets the height of the node's caption area.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

```
JavaScript
get captionHeight() {}
set captionHeight(value) {}
```

Property Value

Number

A number specifying the height of the caption bar. The default value is 6.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.cellFrameStyle Property

Gets or sets the style of cells' frame lines.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get cellFrameStyle() {}
set cellFrameStyle(value) {}
```

Property Value

CellFrameStyle

A member of the CellFrameStyle enumeration.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.cellStyle Property

Gets or sets the style used to draw table's cells.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get cellStyle() {}
set cellStyle(value) {}
```

Property Value

Style

An instance of the Style class.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.cellTextColor Property

Gets or sets the color used to fill cells text.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get cellTextColor() {}
set cellTextColor(value) {}
```

Property Value

String

A string value identifying the text fill color.

Remarks

If this property is not set, cells derive their text color from table's textColor property, which also applies to the table's caption. You can also set textColor property of individual cells.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.columnCount Property

Gets or sets the number of columns in the table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get columnCount() {}
set columnCount(value) {}
```

Property Value

Number

The number of columns in the table.

Remarks

If you increase the number of columns the content of the initialcolumns will remain intact. If you decrease the number, the lastcolumns are removed and the other remain unchanged.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.connectionStyle Property

Gets or sets whether drawing a link from/to this node should connectan individual table row, or the whole table as an integral entity.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get connectionStyle() {}
set connectionStyle(value) {}
```

Property Value

ConnectionStyle

A member of the ConnectionStyle enumeration.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.currentScrollRow Property

Gets or sets the table's current scroll position.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get currentScrollRow() {}
set currentScrollRow(value) {}
```

Property Value

Number

The index of the first row that should be visible in the table.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.enableStyledText Property

Gets or sets a value indicating whether styled text rendering is enabled.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get enableStyledText() {}
set enableStyledText(value) {}
```

Property Value

Boolean

true to parse the node's text in order to find formatting tags, or **false** otherwise.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.rowAnchorPattern Property

Gets or sets the default anchor pattern for the rows of a table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get rowAnchorPattern() {}
set rowAnchorPattern(value) {}
```

Property Value

AnchorPattern

An AnchorPattern object that defines a set of anchor points.

Remarks

You can specify individual row's anchor points by setting the anchorPattern property of TableRow.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.rowCount Property

Gets or sets the number of rows in the table.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get rowCount() {}
set rowCount(value) {}
```

Property Value

Number

The number of rows in the table.

Remarks

If you increase the number of rows the content of the initialrows will remain intact. If you decrease the number, the lastrows are removed and the other remain unchanged.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.scrollable Property

Gets or sets whether the user is allowed to scroll the table rows.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get scrollable() {}
set scrollable(value) {}
```

Property Value

Boolean

true to allow scrolling, or **false** otherwise.

See Also

TableNode Members
TableNode Class
MindFusion.Diagramming Namespace

TableNode.shape Property

Gets or sets the shape of the node's outline.

Namespace: MindFusion.Diagramming
File: TableNode.js

Syntax

JavaScript

```
get shape() {}
set shape(value) {}
```

Property Value

SimpleShape

A member of the SimpleShape enumeration.

See Also

TableRow Class

Represents a row of a TableNode.

Syntax

```
JavaScript
// class
TableRow.prototype = {}
```

Remarks

Call the getRow method of TableNode to get the **TableRow** instance at specified position. In order to add or remove rows, set table's rowCount property, or call methods such as addRow, insertRow, deleteRow, redimTable.

Inheritance Hierarchy

MindFusion.Diagramming.TableRow

See Also

TableRow Members
MindFusion.Diagramming Namespace
TableColumn Class

TableRow Members

Represents a row of a TableNode.

The following tables list the members exposed by the TableRow type.

Public Constructors

	Name	Description
	TableRow	Initializes a new instance of the TableRow class.

Public Properties

	Name	Description
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this row.
	expanded	Gets or sets a value indicating whether the table section starting from this header row is collapsed or expanded.

	header	Gets or sets a value indicating whether this row is a section header.
	height	Gets or sets the height of this row.

See Also

TableRow Class
MindFusion.Diagramming Namespace

TableRow Constructor

Initializes a new instance of the TableRow class.

Namespace: MindFusion.Diagramming
File: TableRow.js

Syntax

JavaScript

```
function TableRow ()
```

See Also

TableRow Class
MindFusion.Diagramming Namespace

TableRow Properties

TableRow.anchorPattern Property

Gets or sets the anchor points to which links attach when connected to this row.

Namespace: MindFusion.Diagramming
File: TableRow.js

Syntax

JavaScript

```
get anchorPattern() {}
set anchorPattern(value) {}
```

Property Value

AnchorPattern

An AnchorPattern object that defines a set of anchor points.

Remarks

If this property is not set, links attach to points defined by table's rowAnchorPattern property.

See Also

TableRow Members
TableRow Class
MindFusion.Diagramming Namespace

TableRow.expanded Property

Gets or sets a value indicating whether the table section starting from this header row is collapsed or expanded.

Namespace: MindFusion.Diagramming
File: TableRow.js

Syntax

```
JavaScript
get expanded() {}
set expanded(value) {}
```

Property Value

Boolean

true if the table section that starts from this row is expanded; or **false** otherwise. The default is **true**.

Remarks

This property can be set only if the row is a header. Links connected to the rows of a collapsed table section appear to be connected to the section header row.

See Also

TableRow Members
TableRow Class
MindFusion.Diagramming Namespace

TableRow.header Property

Gets or sets a value indicating whether this row is a section header.

Namespace: MindFusion.Diagramming
File: TableRow.js

Syntax

```
JavaScript
get header() {}
set header(value) {}
```

Property Value

Boolean

true if this is the header row of a table section, otherwise **false**. The default is **false**.

Remarks

Each header row defines a section of the table that includes all non-header rows after the header. Such sections can be expanded or collapsed, that is, the rows included in them can be shown or hidden. This can be done interactively by pressing the [±] button displayed in header rows, or programmatically by setting the expanded property. The tableSectionCollapsed and tableSectionExpanded events are raised when a user clicks the [±] button.

See Also

TableRow Members
TableRow Class
MindFusion.Diagramming Namespace

TableRow.height Property

Gets or sets the height of this row.

Namespace: MindFusion.Diagramming
File: TableRow.js

Syntax

JavaScript

```
get height() {}
set height(value) {}
```

Property Value

Number

A number specifying the width.

See Also

TableRow Members
TableRow Class
MindFusion.Diagramming Namespace

Theme Class

Represents a set of Style objects whose properties are used to render items of a specific type in the current diagram.

Syntax

JavaScript

```
// class
Theme.prototype = {}
```

Remarks

Theme objects can be assigned to the theme property of Diagram.

Inheritance Hierarchy

MindFusion.Diagramming.Theme

See Also

Theme Members
MindFusion.Diagramming Namespace

Theme Members

Represents a set of Style objects whose properties are used to render items of a specific type in the current diagram.

The following tables list the members exposed by the Theme type.

Public Constructors

	Name	Description
	Theme	Initializes a new instance of the Theme class.

Public Properties

	Name	Description
	styles	A dictionary object that maps item type identifiers to **Style** objects.

Public Methods

	Name	Description
	fromId	Gets a built-in theme by its string identificator.
	getBlack	Gets the built-in Black theme.
	getBlue	Gets the built-in Blue theme.
	getBusiness	Gets the built-in Business theme.
	getEarth	Gets the built-in Earth theme.
	getGray	Gets the built-in Gray theme.
	getGreen	Gets the built-in Green theme.
	getLight	Gets the built-in Light theme.
	getPastel	Gets the built-in Pastel theme.
	getPeach	Gets the built-in Peach theme.
	getStandard	Gets the built-in Standard theme.

See Also

Theme Class
MindFusion.Diagramming Namespace

Theme Constructor

Initializes a new instance of the Theme class.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function Theme ()
```

See Also

Theme Class
MindFusion.Diagramming Namespace

Theme Methods

Theme.fromId Method

Gets a built-in theme by its string identificator.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function fromId (id)
```

Parameters

id
String. The theme identificator.

Return Value

Theme. An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getBlack Method

Gets the built-in Black theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

JavaScript

```
function getBlack ()
```

Return Value

Theme. An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getBlue Method

Gets the built-in Blue theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

JavaScript

```
function getBlue ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getBusiness Method

Gets the built-in Business theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

JavaScript

```
function getBusiness ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getEarth Method

Gets the built-in Earth theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getEarth ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getGray Method

Gets the built-in Gray theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getGray ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getGreen Method

Gets the built-in Green theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getGreen ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getLight Method

Gets the built-in Light theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getLight ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getPastel Method

Gets the built-in Pastel theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getPastel ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getPeach Method

Gets the built-in Peach theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getPeach ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme.getStandard Method

Gets the built-in Standard theme.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
function getStandard ()
```

Return Value

An instance of the Theme class.

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

Theme Properties

Theme.styles Property

A dictionary object that maps item type identifiers to Style objects.

Namespace: MindFusion.Diagramming
File: Theme.js

Syntax

```
JavaScript
get styles() {}
set styles(value) {}
```

Property Value

Map

Maps item types to Style objects.

Example

The following code sets default white background of ShapeNode objects when the theme is applied.

```
JavaScript
var shapeNodeStyle = new Style();
shapeNodeStyle.brush = "white";
theme.styles.set("std:ShapeNode", shapeNodeStyle);
```

See Also

Theme Members
Theme Class
MindFusion.Diagramming Namespace

TreeItemEventArgs Class

Contains the arguments passed to handlers of tree view items -related events.

Syntax

```
JavaScript
// class
TreeItemEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Controls.CancelEventArgs
  MindFusion.Diagramming.TreeItemEventArgs

See Also

TreeItemEventArgs Members
MindFusion.Diagramming Namespace
Events Class

TreeItemEventArgs Members

Contains the arguments passed to handlers of tree view items -related events.

The following tables list the members exposed by the TreeItemEventArgs type.

Public Constructors

	Name	Description
	TreeItemEventArgs	Initializes a new instance of the TreeItemEventArgs class.

Public Properties

	Name	Description
	cancel	Gets or sets a value indicating whether to allow the current operation. (Inherited from CancelEventArgs.)
	context	Gets the canvas rendering context.
	handled	Gets or sets a value indicating whether the event has been handled. (Inherited from EventArgs.)
	item	Gets the item related to the event.
	mouseButton	Gets which mouse button has been pressed.
	newText	Gets the text that has just been entered by the user.
	node	Gets the node related to the event.
	oldText	Gets the text that had been displayed in the node before the user edited it.

See Also

TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs Constructor

Initializes a new instance of the TreeItemEventArgs class.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

JavaScript

```
function TreeItemEventArgs ()
```

See Also

TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs Properties

TreeItemEventArgs.context Property

Gets the canvas rendering context.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get context() {}
set context(value) {}
```

Property Value

CanvasRenderingContext2D

A rendering context used to draw on the Canvas element.

See Also

TreeItemEventArgs Members
TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs.item Property

Gets the item related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get item() {}
set item(value) {}
```

Property Value

TreeViewItem

A TreeViewItem instance.

See Also

TreeItemEventArgs Members
TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs.mouseButton Property

Gets which mouse button has been pressed.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get mouseButton() {}
set mouseButton(value) {}
```

Property Value

Number

An integer mouse button id.

See Also

TreeItemEventArgs Members
TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs.newText Property

Gets the text that has just been entered by the user.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get newText() {}
set newText(value) {}
```

Property Value

String

A string containing newly entered text.

See Also

TreeItemEventArgs Members
TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs.node Property

Gets the node related to the event.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get node() {}
set node(value) {}
```

Property Value

TreeViewNode

A TreeViewNode instance.

See Also

TreeItemEventArgs Members
TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeItemEventArgs.oldText Property

Gets the text that had been displayed in the node before the user edited it.

Namespace: MindFusion.Diagramming
File: EventArgs.js

Syntax

```
JavaScript
get oldText() {}
set oldText(value) {}
```

Property Value

String

A string containing item's original text.

See Also

TreeItemEventArgs Members
TreeItemEventArgs Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint Class

A class that represents a connection between an edge and a node in a graph.

Syntax

```
JavaScript
// class
TreeViewConnectionPoint.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.ConnectionPoint
MindFusion.Diagramming.TreeViewConnectionPoint

See Also

TreeViewConnectionPoint Members
MindFusion.Diagramming Namespace

TreeViewConnectionPoint Members

A class that represents a connection between an edge and a node in a graph.

The following tables list the members exposed by the TreeViewConnectionPoint type.

Public Properties

	Name	Description
	endPoint	Calculates the coordinates of this connection point expressed in document coordinates.

Public Methods

	Name	Description
	addLinkToNode	Adds the link associated with this connection point to the associated node.
	getInitialPoint	Returns a point (usually the center of the bounding rectangle) contained within the associated node.
	getIntersection	Calculates the intersection point between the specified line segment and the contour of the node associated with this connection point.
	getNearestAnchorPoint	Returns the nearest anchor Point pertinent to this connection. (Inherited from ConnectionPoint.)
	getNearestBorderPoint	Returns the nearest point lying on the node border.
	getNodeRect	Returns the bounding rectangle of the node associated with this connection point. (Inherited from ConnectionPoint.)
	linkChanges	Checks whether the link would change if attached to the specified node at the specified point.
	nodesIntersect	Checks whether the node associated with the current connection point and the node associated with the specified connection point intersect. (Inherited from ConnectionPoint.)
	removeLinkFromNode	Removes the link associated with this connection point from the associated node.
	saveEndRelative	Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node. (Inherited from ConnectionPoint.)
	saveEndRelative	Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node.

See Also

TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint Methods

TreeViewConnectionPoint.addLinkToNode Method

Adds the link associated with this connection point to the associated node.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

JavaScript

```
function addLinkToNode ()
```

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint.getInitialPoint Method

Returns a point (usually the center of the bounding rectangle) contained within the associated node.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

JavaScript

```
function getInitialPoint ()
```

Return Value

Point. A Point contained within the node.

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint.getIntersection Method

Calculates the intersection point between the specified line segment and the contour of the node associated with this connection point.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

```
JavaScript
function getIntersection (point1, point2)
```

Parameters

point1
Point. The first point of the segment.

point2
Point. The second point of the segment.

Return Value

Point. A Point object representing the intersection point.

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint.getNearestBorderPoint Method

Returns the nearest point lying on the node border.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

```
JavaScript
function getNearestBorderPoint (point)
```

Parameters

point
Point. A Point instance specifying location in the diagram.

Return Value

AnchorPointDetails. An AnchorPointDetails specifying the anchor point properties.

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint.linkChanges Method

Checks whether the link would change if attached to the specified node at the specified point.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

```
JavaScript
function linkChanges (item, point)
```

Parameters

item
DiagramItem. A DiagramItem under the mouse cursor.

point
Point. A point where the link will be connected.

Return Value

Boolean. true if the link would connect to a different entity if dropped at the specified location; otherwise, false.

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint.removeLinkFromNode Method

Removes the link associated with this connection point from the associated node.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

```
JavaScript
function removeLinkFromNode ()
```

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint.saveEndRelative Method

Updates the coordinates of the connection point represented by this link, relative to the bounding rectangle of the associated node.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

```
JavaScript
function saveEndRelative ()
```

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewConnectionPoint Properties

TreeViewConnectionPoint.endPoint Property

Calculates the coordinates of this connection point expressed in document coordinates.

Namespace: MindFusion.Diagramming
File: TreeViewConnectionPoint.js

Syntax

```
JavaScript
get endPoint() {}
set endPoint(value) {}
```

Property Value

Point

A Point that represents the coordinate.

See Also

TreeViewConnectionPoint Members
TreeViewConnectionPoint Class
MindFusion.Diagramming Namespace

TreeViewItem Class

Represents a constituent item of the hierarchy displayed inside a TreeViewNode.

Syntax

```
JavaScript
// class
TreeViewItem.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.TreeViewItem

See Also

TreeViewItem Members
MindFusion.Diagramming Namespace

TreeViewItem Members

Represents a constituent item of the hierarchy displayed inside a TreeViewNode.

The following tables list the members exposed by the TreeViewItem type.

Public Constructors

	Name	Description
	TreeViewItem	Initializes a new instance of the TreeViewItem class.

Public Properties

	Name	Description
	brush	Gets or sets the brush used to paint item's background.
	children	Gets a list of this item's children.
	expanded	Gets or sets a value indicating whether this item is expanded to show its children.
	font	Gets or sets the font used to render item's text.
	height	Gets or sets the height of this item.
	image	Gets or sets the image displayed by this item.
	imageContent	Gets or sets a Base64-encoded image to display inside this item.
	imageLocation	Gets or sets URL of the image to display inside this item.
	imagePadding	Gets or sets the padding distance between item's borders and image.
	parent	Gets the parent item of this item.
	tag	Gets or sets custom data tag associated with this item.
	text	Gets or sets the text label displayed by this item.
	textColor	Gets or sets the item's text color.
	tooltip	Gets or sets the tooltip text that should be displayed when the mouse hovers over this item.

Public Methods

	Name	Description
	loadFromXml	Loads the item's content from an XML element.

	saveToXml	Saves the item's content into an XML element.

See Also

TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem Constructor

Initializes a new instance of the TreeViewItem class.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
function TreeViewItem ()
```

See Also

TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem Methods

TreeViewItem.loadFromXml Method

Loads the item's content from an XML element.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.saveToXml Method

Saves the item's content into an XML element.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement

Element. An XML DOM element that will contain the item's data.

context

XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem Properties

TreeViewItem.brush Property

Gets or sets the brush used to paint item's background.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.children Property

Gets a list of this item's children.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

```
JavaScript
get children() {}
```

Property Value

Array

An array of TreeViewItem objects.

Remarks

Do not add items to this array directly, but call the addItem method of TreeViewNode.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.expanded Property

Gets or sets a value indicating whether this item is expanded to show its children.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

```
JavaScript
get expanded() {}
set expanded(value) {}
```

Property Value

Boolean

true if the item is expanded, or **false** otherwise.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.font Property

Gets or sets the font used to render item's text.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

```
JavaScript
get font() {}
set font(value) {}
```

Property Value

Font

An instance of the Font class.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.height Property

Gets or sets the height of this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

```
JavaScript
get height() {}
set height(value) {}
```

Property Value

Number

A number specifying item's height.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.image Property

Gets or sets the image displayed by this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get image() {}
set image(value) {}
```

Property Value

HTMLImageElement

An image object to render as item's content.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.imageContent Property

Gets or sets a Base64-encoded image to display inside this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get imageContent() {}
set imageContent(value) {}
```

Property Value

String

A Base64-encoded representation of an image.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.imageLocation Property

Gets or sets URL of the image to display inside this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get imageLocation() {}
set imageLocation(value) {}
```

Property Value

String

A URL string that identifies the image to be loaded.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.imagePadding Property

Gets or sets the padding distance between item's borders and image.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get imagePadding() {}
set imagePadding(value) {}
```

Property Value

Thickness

A Thickness instance specifying image padding.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.parent Property

Gets the parent item of this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get parent() {}
set parent(value) {}
```

Property Value

TreeViewItem

The parent item.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.tag Property

Gets or sets custom data tag associated with this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get tag() {}
set tag(value) {}
```

Property Value

Object

An Object containing custom data.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.text Property

Gets or sets the text label displayed by this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

A string containing the item's text. The default value is empty string "".

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.textColor Property

Gets or sets the item's text color.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

```
JavaScript
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string value identifying the text color.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewItem.tooltip Property

Gets or sets the tooltip text that should be displayed when the mouse hovers over this item.

Namespace: MindFusion.Diagramming
File: TreeViewItem.js

Syntax

```
JavaScript
get tooltip() {}
set tooltip(value) {}
```

Property Value

String

A string containing the tooltip text.

See Also

TreeViewItem Members
TreeViewItem Class
MindFusion.Diagramming Namespace

TreeViewNode Class

TreeViewNode instances are diagram nodes that can draw hierarchical data.

Syntax

JavaScript

```
// class
TreeViewNode.prototype = {}
```

Remarks

The root items displayed in the node can be accessed through the rootItems property. Items can be added and removed individually by using the addItem and removeItem methods, or in bulk by calling the fromObject method, which loads the tree view items from an array of objects.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.CaptionedNode
      MindFusion.Diagramming.TreeViewNode

See Also

TreeViewNode Members
MindFusion.Diagramming Namespace

TreeViewNode Members

TreeViewNode instances are diagram nodes that can draw hierarchical data.

The following tables list the members exposed by the **TreeViewNode** type.

Public Constructors

	Name	Description
	TreeViewNode	Initializes a new instance of the TreeViewNode class with default values supplied from the specified Diagram.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)

	captionBackBrush	Gets or sets an object specifying how to fill the caption bar.
	captionFont	Gets or sets the font used to render the node's caption text.
	captionHeight	Gets or sets the height of the node's caption area.
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	connectionStyle	Gets or sets whether drawing a link from/to this node should connectan individual TreeViewItem, or the whole node as an integral entity.
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	dividerStroke	Gets or sets the color used to stroke the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	dividerStrokeDashArray	Gets or sets an array of values specifying the dash pattern of the node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from CaptionedNode.)
	dividerStrokeDashStyle	Gets or sets the line dash pattern applied when stroking node's divider line. (Inherited from CaptionedNode.)
	dividerStrokeThickness	Gets or sets the width of the divider line between node's caption bar and content area. (Inherited from CaptionedNode.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)
	effectiveDividerStrokeThickness	Gets the effective line thickness that should be applied when stroking this node's divider line. (Inherited from CaptionedNode.)
	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)

	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	enableStyledText	Gets or sets a value indicating whether styled text rendering is enabled.
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	flatItems	Gets a list of all items in this node.
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)
	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)

	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rootItems	Gets a list of root items displayed in this node.
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	scrollable	Gets or sets whether the node's items can be scrolled vertically.
	scrollPosition	Gets or sets the node's current scroll position.
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	shape	Gets or sets the shape of the node's outline.

	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	showLines	Gets or sets whether to display lines connecting child items to parent items.
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)
	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description
	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addItem	Adds a new child item to the specified parent item or to the root items list, if a parent is not specified.
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	Checks if the TreeViewNode contains the specified point.
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point.
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramNode.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	fromObject	Loads the tree view items from a data object.
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getItemRect	Returns the bounding rectangle of the specified item.
	getOutline	Gets an array of points that define the outline of this node shape.
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)

getRotatedBounds	Gets the bounding rectangle of this item when it is rotated. (Inherited from DiagramNode.)
hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)
itemFromPoint	Returns the item that contains the specified point.
loadFromXml	DiagramNode.LoadFromXml override. Loads the node's content from an XML element.
onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
removeItem	Removes the specified item from the node.
removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
resizeToFitText	Resizes the tree view so that it is large enough to fit the text of its items.
restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
restoreState	DiagramItem.restoreState override. (Inherited from CaptionedNode.)
saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
saveState	DiagramItem.saveState override. (Inherited from CaptionedNode.)
saveToXml	Saves the node's content into an XML element.
scrollDown	Scrolls the node down.
scrollUp	Scrolls the table's rows up.
setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
toJson	DiagramNode.toJson override.

	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

See Also

TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode Constructor

Initializes a new instance of the TreeViewNode class with default values supplied from the specified Diagram.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function TreeViewNode ([parent])
```

Parameters

parent
Optional.
Diagram. The Diagram from which to obtain default values for the node properties.

See Also

TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode Methods

TreeViewNode.addItem Method

Adds a new child item to the specified parent item or to the root items list, if a parent is not specified.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function addItem (item, [parentItem])
```

Parameters

item
TreeViewItem. The item to add.

parentItem
Optional.
TreeViewItem. The parent item.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function clone ()
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.containsPoint Method

Checks if the TreeViewNode contains the specified point.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function containsPoint (point)
```

Parameters

point
Point. The Point to check.

Return Value

Boolean. true if the point is within the bounds of this node; otherwise, false.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.createConnectionPoint Method

Creates a connection point between this node and the specified link at the specified point.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
function createConnectionPoint (link, point, incoming)
```

Parameters

link
DiagramLink. A DiagramLink that should be connected to this node.

point
Point. The point at which the link should be connected.

incoming
Boolean. true if link is an incoming link, otherwise, false.

Return Value

TreeViewConnectionPoint. The connection point.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.fromObject Method

Loads the tree view items from a data object.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function fromObject (data)
```

Parameters

data
Object. The object containing the tree view data.

Example

JavaScript

```
node.fromObject(
[{
    text: "Order",
    imageLocation: "icon_calculator.png"
},
{
    text: "Date",
    imageLocation: "icon_time.png"
},
{
    text: "Products",
    imageLocation: "icon_delivery.png",
    children: [{ text: "Name" }, { text: "TotalPrice", brush: "lightblue" }]
}]
);
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.getItemRect Method

Returns the bounding rectangle of the specified item.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function getItemRect (item)
```

Parameters

item
TreeViewItem. The item.

Return Value

Rect. A Rect object representing the item's bounding rectangle.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.getOutline Method

Gets an array of points that define the outline of this node shape.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function getOutline ()
```

Return Value

Array. An array with the points.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.itemFromPoint Method

Returns the item that contains the specified point.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function itemFromPoint (point)
```

Parameters

point
Point. A Point instance.

Return Value

TreeViewItem. The item containing the point, or null.

See Also

TreeViewNode Members
TreeViewNode Class

TreeViewNode.loadFromXml Method

DiagramNode.LoadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.removeItem Method

Removes the specified item from the node.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function removeItem (item)
```

Parameters

item
TreeViewItem. The item to remove.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.resizeToFitText Method

Resizes the tree view so that it is large enough to fit the text of its items.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function resizeToFitText ()
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.scrollDown Method

Scrolls the node down.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function scrollDown ()
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.scrollUp Method

Scrolls the table's rows up.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function scrollUp ()
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
function toJson ()
```

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode Properties

TreeViewNode.captionBackBrush Property

Gets or sets an object specifying how to fill the caption bar.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get captionBackBrush() {}
set captionBackBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.captionFont Property

Gets or sets the font used to render the node's caption text.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get captionFont() {}
set captionFont(value) {}
```

Property Value

Font

An object specifying the font of the caption bar.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.captionHeight Property

Gets or sets the height of the node's caption area.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get captionHeight() {}
set captionHeight(value) {}
```

Property Value

Number

A number specifying the height of the caption bar. The default value is 6.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.connectionStyle Property

Gets or sets whether drawing a link from/to this node should connectan individual TreeViewItem, or the whole node as an integral entity.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
get connectionStyle() {}
set connectionStyle(value) {}
```

Property Value

ConnectionStyle

A member of the ConnectionStyle enumeration.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.enableStyledText Property

Gets or sets a value indicating whether styled text rendering is enabled.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
get enableStyledText() {}
set enableStyledText(value) {}
```

Property Value

Boolean

true to parse the node's text in order to find formatting tags, or **false** otherwise.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.flatItems Property

Gets a list of all items in this node.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
get flatItems() {}
set flatItems(value) {}
```

Property Value

Array

An array containing all TreeViewItem objects in the node.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.rootItems Property

Gets a list of root items displayed in this node.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

JavaScript

```
get rootItems() {}
set rootItems(value) {}
```

Property Value

Array

An array of root TreeViewItem objects.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.scrollable Property

Gets or sets whether the node's items can be scrolled vertically.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get scrollable() {}
set scrollable(value) {}
```

Property Value

Boolean

true to allow scrolling, or **false** otherwise.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.scrollPosition Property

Gets or sets the node's current scroll position.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get scrollPosition() {}
set scrollPosition(value) {}
```

Property Value

Number

A number specifying the scroll position.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.shape Property

Gets or sets the shape of the node's outline.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get shape() {}
set shape(value) {}
```

Property Value

SimpleShape

A member of the SimpleShape enumeration.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

TreeViewNode.showLines Property

Gets or sets whether to display lines connecting child items to parent items.

Namespace: MindFusion.Diagramming
File: TreeViewNode.js

Syntax

```
JavaScript
get showLines() {}
set showLines(value) {}
```

Property Value

Boolean

true to display connecting lines, or **false** otherwise.

See Also

TreeViewNode Members
TreeViewNode Class
MindFusion.Diagramming Namespace

UndoEventArgs Class

Contains the arguments passed to the actionUndone and actionRedone event handlers.

Syntax

```
JavaScript
// class
UndoEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.EventArgs
MindFusion.Diagramming.UndoEventArgs

See Also

UndoEventArgs Members
MindFusion.Diagramming Namespace

UndoEventArgs Members

Contains the arguments passed to the actionUndone and actionRedone event handlers.

The following tables list the members exposed by the UndoEventArgs type.

Public Constructors

	Name	Description
	UndoEventArgs	Initializes a new instance of the UndoEventArgs class.

Public Properties

	Name	Description
	command	Gets the command related to the event.

See Also

UndoEventArgs Class
MindFusion.Diagramming Namespace

UndoEventArgs Constructor

Initializes a new instance of the UndoEventArgs class.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

JavaScript

```
function UndoEventArgs (args)
```

Parameters

args

See Also

UndoEventArgs Class
MindFusion.Diagramming Namespace

UndoEventArgs Properties

UndoEventArgs.command Property

Gets the command related to the event.

Namespace: MindFusion.Diagramming
File: Diagram.js

Syntax

```
JavaScript
get command() {}
set command(value) {}
```

Property Value

Command

A Command instance.

See Also

UndoEventArgs Members
UndoEventArgs Class
MindFusion.Diagramming Namespace

VideoNode Class

A **VideoNode** instance renders video stream assigned to the videoLocation property.

Syntax

```
JavaScript
// class
VideoNode.prototype = {}
```

Remarks

VideoNode objects display video streams, along with UI elements to play, pause, seek and change volume. Set node's videoLocation method to specify its video URL. Auto-playing is not supported at this time, and the video can be started only by users clicking the play button.

VideoNode is built around Video component, which can also be used in CompositeNode templates.

Inheritance Hierarchy

MindFusion.Diagramming.DiagramItem
  MindFusion.Diagramming.DiagramNode
    MindFusion.Diagramming.VideoNode

See Also

VideoNode Members

A VideoNode instance renders video stream assigned to the videoLocation property.

The following tables list the members exposed by the **VideoNode** type.

Public Constructors

	Name	Description
	VideoNode	Initializes a new instance of the **VideoNode** class.

Public Properties

	Name	Description
	allowIncomingLinks	Gets or sets whether users are allowed to draw incoming links to this node. (Inherited from DiagramNode.)
	allowOutgoingLinks	Gets or sets whether users are allowed to draw outgoing links from this node. (Inherited from DiagramNode.)
	anchorPattern	Gets or sets the anchor points to which links attach when connected to this node. (Inherited from DiagramNode.)
	attachedNodes	Returns the nodes attached to this node. (Inherited from DiagramNode.)
	bounds	Gets or sets a rectangle that defines the position of this node. (Inherited from DiagramNode.)
	brush	Gets or sets an object specifying how to fill the interior of this item. (Inherited from DiagramItem.)
	center	Gets the center of this node's bounding rectangle. (Inherited from DiagramNode.)
	constraints	Gets or sets constraints applied to the location and size of this node. (Inherited from DiagramNode.)
	container	Gets the parent ContainerNode containing this node. (Inherited from DiagramItem.)
	cornerRadius	Gets or sets the corner radius of nodes whose shape is set to a rounded rectangle. (Inherited from DiagramNode.)
	cssClass	Gets or sets the CSS class associated with this item. (Inherited from DiagramItem.)
	cssElementName	Gets the name of the CSS element associated with this item. (Inherited from DiagramItem.)
	effectiveBrush	Gets the effective brush that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveCaptionBackBrush	Gets the effective brush that should be used to paint node's caption bar. (Inherited from DiagramItem.)

	effectiveEffects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	effectiveFont	Gets the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontName	Gets the name of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontSize	Gets the size of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveFontStyle	Gets the style of the effective font that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveOpacity	Gets the effective opacity that should be used to paint this item. (Inherited from DiagramItem.)
	effectiveShadowColor	Gets the effective color that should be used to render this item's shadow. (Inherited from DiagramItem.)
	effectiveStroke	Gets the effective color that should be used to stroke this item. (Inherited from DiagramItem.)
	effectiveStrokeDashArray	Gets the effective line dash array that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashOffset	Gets the effective line dash offset that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeDashStyle	Gets the effective line dash pattern that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveStrokeThickness	Gets the effective line thickness that should be applied when stroking this item. (Inherited from DiagramItem.)
	effectiveTextColor	Gets the effective text color that should be used to render this item's text. (Inherited from DiagramItem.)
	effectiveTextStroke	Gets the effective text color that should be used to render this item's text outline. (Inherited from DiagramItem.)
	effectiveTextStrokeThickness	Gets the effective width of the text outline of this item. (Inherited from DiagramItem.)
	effects	Gets the effects applied to this node. (Inherited from DiagramNode.)
	enabledHandles	Gets or sets a value indicating which adjustment handles are enabled for this node. (Inherited from DiagramNode.)
	expandable	Gets or sets whether users are allowed to expand or collapse the subtree rooted at this node. (Inherited from DiagramNode.)
	expanded	Gets or sets whether the subtree rooted at this node is expanded. (Inherited from DiagramNode.)
	font	Gets or sets the font used to render item's text. (Inherited from DiagramItem.)
	graphicsContent	Gets an array of visual elements used to render this item on the canvas. (Inherited from DiagramItem.)

	handlesStyle	Gets or sets a value specifying appearance and behavior of node's adjustment handles. (Inherited from DiagramNode.)
	hyperLink	Gets or sets the hyperlink associated with this diagram item. (Inherited from DiagramItem.)
	id	Gets or sets a custom identifier associated with this item. (Inherited from DiagramItem.)
	ignoreLayout	Gets or sets a value indicating whether the position of this item should not be changed by automatic layout methods. (Inherited from DiagramItem.)
	incomingLinks	Gets the incoming links to this node. (Inherited from DiagramNode.)
	layoutTraits	Gets an object containing properties specific to individual layout algorithms. (Inherited from DiagramItem.)
	lineAlignment	Gets or sets the vertical alignment of item's text. (Inherited from DiagramItem.)
	locked	Gets or sets whether users are allowed to interact with this item. (Inherited from DiagramItem.)
	masterNode	Returns the node to which this node is attached. (Inherited from DiagramNode.)
	obstacle	Gets or sets whether this node is considered an obstacle by link routing algorithms. (Inherited from DiagramNode.)
	opacity	Gets or sets opacity of this item. (Inherited from DiagramItem.)
	outgoingLinks	Gets the outgoing links from this node. (Inherited from DiagramNode.)
	parent	Gets the Diagram to which this item belongs. (Inherited from DiagramItem.)
	pen	Gets or sets the color used to stroke item's borders. (Inherited from DiagramItem.)
	rotationAngle	Gets or sets the angle at which this node is rotated. (Inherited from DiagramNode.)
	selected	Gets or sets a value indicating whether a diagram item is selected. (Inherited from DiagramItem.)
	shadowColor	Gets or sets the color used to draw item's shadow. (Inherited from DiagramItem.)
	shadowOffsetX	Gets or sets the horizontal offset of item's shadow. (Inherited from DiagramItem.)
	shadowOffsetY	Gets or sets the vertical offset of item's shadow. (Inherited from DiagramItem.)
	showDeleteButton	Gets or sets whether this node displays a delete button. (Inherited from DiagramNode.)
	stroke	Gets or sets the color used to stroke item's frame. (Inherited from DiagramItem.)

	strokeDashArray	Gets or sets an array of values specifying the dash pattern. (Inherited from DiagramItem.)
	strokeDashOffset	Gets or sets the the distance between the line start and the beginning of the dash pattern. (Inherited from DiagramItem.)
	strokeDashStyle	Gets or sets the line dash pattern applied when stroking item's frame. (Inherited from DiagramItem.)
	strokeThickness	Gets or sets the line width applied when stroking item's frame. (Inherited from DiagramItem.)
	style	Gets or sets the style associated with this item. (Inherited from DiagramItem.)
	tag	Gets or sets custom data tag associated with this item. (Inherited from DiagramItem.)
	text	Gets or sets the text label displayed by this item. (Inherited from DiagramItem.)
	textAlignment	Gets or sets the horizontal alignment of item's text. (Inherited from DiagramItem.)
	textColor	Gets or sets the color used to fill item's text. (Inherited from DiagramItem.)
	textComponent	Gets the Text object instance used to render item's text. (Inherited from DiagramItem.)
	textPadding	Gets or sets the padding distance between item's boundaries and text label. (Inherited from DiagramItem.)
	textStroke	Gets or sets the color used to stroke item's text. (Inherited from DiagramItem.)
	textStrokeThickness	Gets or sets the outline width of item's text. (Inherited from DiagramItem.)
	tooltip	Gets or sets tooltip text that should be displayed when the mouse hovers over this item. (Inherited from DiagramItem.)
	topLevel	Gets if this is a top-level item in the diagram. (Inherited from DiagramItem.)
	videoLocation	Gets or sets the URL of the video stream displayed in this VideoNode.
	visible	Gets or sets a value indicating whether this item is visible. (Inherited from DiagramItem.)
	weight	Gets or sets a weight value used by layout and path-finding algorithms. (Inherited from DiagramItem.)
	zIndex	Gets the z-order position of this item. (Inherited from DiagramItem.)

Public Methods

	Name	Description

	acceptLinks	Determines whether the node can be connected with links of the specified type. (Inherited from DiagramNode.)
	addCanvasElements	Derived classes must implement this method by adding their graphics primitives to the canvas.elements collection. (Inherited from DiagramItem.)
	addLabel	Adds a new label to this item. (Inherited from DiagramItem.)
	allowDrag	DiagramItem.allowDrag override. (Inherited from DiagramNode.)
	animate	Animates the item. (Inherited from DiagramItem.)
	attach	Attaches the specified subordinate node to this node, so that when this node is moved, the subordinate follows it. (Inherited from DiagramNode.)
	attachTo	Attaches this node to the specified master node so that when the master is moved, the attached node follows it. (Inherited from DiagramNode.)
	clone	DiagramNode.clone override.
	containsPoint	**DiagramItem.containsPoint** override. (Inherited from DiagramNode.)
	createConnectionPoint	Creates a connection point between this node and the specified link at the specified point. (Inherited from DiagramNode.)
	detach	Detaches this node from its current master node. (Inherited from DiagramNode.)
	endDrag	**DiagramItem.endDrag** override. (Inherited from DiagramNode.)
	fromJson	DiagramNode.fromJson override.
	fromJson	Deserializes this item from a JSON object. (Inherited from DiagramItem.)
	getAllLinks	Returns all links connected to this node. (Inherited from DiagramNode.)
	getIntersection	Calculates the intersection point between the segment defined by the specified point pair and the node's outline. (Inherited from DiagramNode.)
	getOutline	Gets the bounding rectangle of this node. (Inherited from DiagramNode.)
	getRepaintBounds	Gets the repaint region for this item, taking into account factors such as pen width, selection handles and shadow. (Inherited from DiagramItem.)
	getRotatedBounds	Gets the bounding **rectangle** of this item when it is rotated. (Inherited from DiagramNode.)
	hitTest	Returns the diagram item that contains the specified point. (Inherited from DiagramItem.)
	invalidate	Invalidates the region of the canvas, that matches the item's repaint bounds. (Inherited from DiagramItem.)

	loadFromXml	DiagramNode.loadFromXml override. Loads the node's content from an XML element.
	onRemove	Called when an item is removed from a **Diagram**, either by the user, programmatically or via undo / redo. (Inherited from DiagramItem.)
	raiseClicked	Raises an event which notifies that a **node** has been clicked. (Inherited from DiagramNode.)
	raiseDoubleClicked	Raises an event which notifies that a node has been double-clicked. (Inherited from DiagramNode.)
	removeCanvasElements	DiagramItem.removeCanvasElements override. (Inherited from DiagramNode.)
	removeLabel	Removes the specified label from this item. (Inherited from DiagramItem.)
	restoreLocationState	DiagramItem.restoreLocationState override. (Inherited from DiagramNode.)
	restoreState	DiagramItem.restoreState override. (Inherited from DiagramNode.)
	restoreState	Loads the properties of this diagram item. (Inherited from DiagramItem.)
	saveLocationState	**DiagramItem.saveLocationState** override. (Inherited from DiagramNode.)
	saveState	DiagramItem.saveState override. (Inherited from DiagramNode.)
	saveToXml	Saves the node's content into an XML element.
	setBounds	Sets the rectangle that defines the position of the diagram node. (Inherited from DiagramNode.)
	startDrag	Invoked when the user starts dragging this item. (Inherited from DiagramItem.)
	toJson	DiagramNode.toJson override.
	updateCanvasElements	Derived classes must implement this method to update the **item**'s visual elements. (Inherited from DiagramItem.)
	updateDrag	**DiagramItem.updateDrag** override. (Inherited from DiagramNode.)

See Also

VideoNode Class
MindFusion.Diagramming Namespace

VideoNode Constructor

Initializes a new instance of the VideoNode class.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

```
JavaScript
function VideoNode ([parent])
```

Parameters

parent
Optional.
A Diagram instance whose default node attributes are copied to this node.

See Also

VideoNode Class
MindFusion.Diagramming Namespace

VideoNode Methods

VideoNode.clone Method

DiagramNode.clone override.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

```
JavaScript
function clone ()
```

See Also

VideoNode Members
VideoNode Class
MindFusion.Diagramming Namespace

VideoNode.fromJson Method

DiagramNode.fromJson override.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

VideoNode Members
VideoNode Class

VideoNode.loadFromXml Method

DiagramNode.loadFromXml override. Loads the node's content from an XML element.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element containing the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

VideoNode Members
VideoNode Class
MindFusion.Diagramming Namespace

VideoNode.saveToXml Method

Saves the node's content into an XML element.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

VideoNode Members
VideoNode Class
MindFusion.Diagramming Namespace

VideoNode.toJson Method

DiagramNode.toJson override.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

```
JavaScript
function toJson ()
```

See Also

VideoNode Members
VideoNode Class
MindFusion.Diagramming Namespace

VideoNode Properties

VideoNode.videoLocation Property

Gets or sets the URL of the video stream displayed in this VideoNode.

Namespace: MindFusion.Diagramming
File: VideoNode.js

Syntax

```
JavaScript
get videoLocation() {}
set videoLocation(value) {}
```

Property Value

String

A string containing video stream URL.

See Also

VideoNode Members
VideoNode Class
MindFusion.Diagramming Namespace

WebComponents Class

Implements registration of web components.

Syntax

```
JavaScript
```

```
// class
WebComponents.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.WebComponents

See Also

WebComponents Members
MindFusion.Diagramming Namespace

WebComponents Members

Implements registration of web components.

The following tables list the members exposed by the WebComponents type.

Public Methods

	Name	Description
	register	Registers the controls in the package as web components.

See Also

WebComponents Class
MindFusion.Diagramming Namespace

WebComponents Methods

WebComponents.register Method

Registers the controls in the package as web components.

Namespace: MindFusion.Diagramming
File: WebComponents.js

Syntax

JavaScript
`function register ()`

Remarks

Call this method to register the control classes from Diagramming package as web components. After registration, the browser instantiates corresponding classes for following tags in HTML code:

HTML	JS Class
<mindfusion-diagramview>	DiagramView
<mindfusion-overview>	Overview

<mindfusion-nodelistview>	NodeListView
<mindfusion-ruler>	Ruler
<mindfusion-zoomcontrol>	ZoomControl

You can get a reference to objects created by the browser by calling the find method of corresponding class with **id** attribute's value as argument.

See Also

WebComponents Members
WebComponents Class
MindFusion.Diagramming Namespace
create Method

XmlPersistContext Class

Contains helper methods for serialization of diagram contents to and from XML documents.

Syntax

JavaScript

```
// class
XmlPersistContext.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.XmlPersistContext

See Also

XmlPersistContext Members
MindFusion.Diagramming Namespace

XmlPersistContext Members

Contains helper methods for serialization of diagram contents to and from XML documents.

The following tables list the members exposed by the XmlPersistContext type.

Public Constructors

	Name	Description
	XmlPersistContext	Initializes a new instance of the XmlPersistContext class.

Public Methods

	Name	Description
	addChildElement	Adds a new child node with the specified name and value to the specified parent node.

	readBool	Reads a Boolean value with the specified name.
	readBrush	Reads a Brush object with the specified name.
	readFloat	Reads a floating-point number with the specified name.
	readImage	Reads an image object with the specified name.
	readInt	Reads an integer number with the specified name.
	readString	Reads a string value with the specified name.
	writeBool	Writes a Boolean value with the specified name.
	writeBrush	Writes a reference to the specified brush. The brush is registered within the internal hashtable for subsequent serialization.
	writeFloat	Writes a floating-point number with the specified name.
	writeImage	Writes a reference to the specified image. The image is registered within the internal hashtable for subsequent serialization.
	writeInt	Writes an integer number with the specified name.
	writeString	Writes a string value with the specified name.

See Also

XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext Constructor

Initializes a new instance of the XmlPersistContext class.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function XmlPersistContext (diagram)
```

Parameters

diagram
Diagram. The Diagram that will be serialized or deserialized.

See Also

XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext Methods

XmlPersistContext.addChildElement Method

Adds a new child node with the specified name and value to the specified parent node.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function addChildElement (elementName, parentElement, [innerText])
```

Parameters

elementName
String. A string containing the name of new child element.

parentElement
Element. The parent DOM Element object.

innerText
Optional.
String. Optional value to be added as child text node of the new element.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.readBool Method

Reads a Boolean value with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function readBool (elementName, parentElement, defaultValue)
```

Parameters

elementName
String. A string specifying the name of the Boolean XML element.

parentElement
Element. The parent DOM Element object.

defaultValue
Boolean. The default value to return if specified child XML element does not exist.

Return Value

Boolean. The value loaded from XML.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.readBrush Method

Reads a Brush object with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function readBrush (elementName, parentElement, [defaultValue])
```

Parameters

elementName
String. A string specifying the name of the brush XML element.

parentElement
Element. The parent DOM Element object.

defaultValue
Optional.
Object. The default brush to return if specified child XML element does not exist.

Return Value

Object. The brush object loaded from XML.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.readFloat Method

Reads a floating-point number with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function readFloat (elementName, parentElement, [defaultValue])
```

Parameters

elementName
String. A string specifying the name of the number XML element.

parentElement
Element. The parent DOM Element object.

defaultValue
Optional.
Number. The default number to return if specified child XML element does not exist.

Return Value

Number. The number loaded from XML.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.readImage Method

Reads an image object with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function readImage (elementName, parentElement)
```

Parameters

elementName
String. A string specifying the name of the image XML element.

parentElement
Element. The parent DOM Element object.

Return Value

String. Base64-encoded image data loaded from XML.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.readInt Method

Reads an integer number with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function readInt (elementName, parentElement, [defaultValue])
```

Parameters

elementName
String. A string specifying the name of the number XML element.

parentElement
Element. The parent DOM Element object.

defaultValue
Optional.
Number. The default number to return if specified child XML element does not exist.

Return Value

Number. The number loaded from XML.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.readString Method

Reads a string value with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function readString (elementName, parentElement, [defaultValue])
```

Parameters

elementName
String. The name of the string's XML element.

parentElement
Element. The parent DOM Element object.

defaultValue
Optional.
Object. The default string to return if specified child XML element does not exist.

Return Value

String. The string loaded from XML.

See Also

XmlPersistContext.writeBool Method

Writes a Boolean value with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function writeBool (boolValue, elementName, parentElement)
```

Parameters

boolValue
Boolean. The Boolean value that should be saved in current XML document.

elementName
String. A string specifying the name of the new element.

parentElement
Element. The parent DOM Element object.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.writeBrush Method

Writes a reference to the specified brush. The brush is registered within the internal hashtable for subsequent serialization.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function writeBrush (brush, elementName, parentElement)
```

Parameters

brush
Object. The Brush object that should be saved in current XML document.

elementName
String. A string specifying the name of the new element.

parentElement
Element. The parent DOM Element object.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.writeFloat Method

Writes a floating-point number with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function writeFloat (floatValue, elementName, parentElement)
```

Parameters

floatValue
Number. The number that should be saved in current XML document.

elementName
String. A string specifying the name of the new element.

parentElement
Element. The parent DOM Element object.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.writeImage Method

Writes a reference to the specified image. The image is registered within the internal hashtable for subsequent serialization.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function writeImage (image, elementName, parentElement)
```

Parameters

image
String. Base64-encoded image data that should be saved in XML.

elementName
String. A string specifying the name of the new element.

parentElement
Element. The parent DOM Element object.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.writeInt Method

Writes an integer number with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function writeInt (intValue, elementName, parentElement)
```

Parameters

intValue
Number. The number that should be saved in current XML document.

elementName
String. A string specifying the name of the new element.

parentElement
Element. The parent DOM Element object.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

XmlPersistContext.writeString Method

Writes a string value with the specified name.

Namespace: MindFusion.Diagramming
File: XmlPersistContext.js

Syntax

JavaScript

```
function writeString (stringValue, elementName, parentElement)
```

Parameters

stringValue
String. The string that should be saved in current XML document.

elementName
String. A string specifying the name of the new element.

parentElement
Element. The parent DOM Element object.

Return Value

Element. The newly created DOM Element object.

See Also

XmlPersistContext Members
XmlPersistContext Class
MindFusion.Diagramming Namespace

MindFusion.Diagramming.Animations Namespace

BounceAnimation Class

Animates vertical position of nodes.

Syntax

JavaScript

```
// class
BounceAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Example

JavaScript

```
node.animate(
    new BounceAnimation(
```

```
        {
            fromValue: 0,
            toValue: -5,
            duration: 250,
            repeat: true,
            repeatCount: 4,
            reverse: true,
            animationType: Animations.AnimationType.BackBow
        }));
```

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.BounceAnimation

See Also

BounceAnimation Members
MindFusion.Diagramming.Animations Namespace

BounceAnimation Members

Animates vertical position of nodes.

The following tables list the members exposed by the BounceAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

BounceAnimation Class
MindFusion.Diagramming.Animations Namespace

FillColorAnimation Class

Animates the fill color of nodes or link arrowheads.

Syntax

JavaScript

```
// class
FillColorAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Example

JavaScript

```
node.animate(
    new FillColorAnimation(
        {
            fromValue: new Drawing.Color("white"),
            toValue: new Drawing.Color("lightSkyBlue"),
            duration: 250,
            keepLastValue: true
        }));
```

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.FillColorAnimation

See Also

FillColorAnimation Members
MindFusion.Diagramming.Animations Namespace

FillColorAnimation Members

Animates the fill color of nodes or link arrowheads.

The following tables list the members exposed by the FillColorAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)

	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

FillColorAnimation Class
MindFusion.Diagramming.Animations Namespace

ItemAnimation Class

Base class of diagram item animations.

Syntax

JavaScript

```
// class
ItemAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Inheritance Hierarchy

MindFusion.Animations.Animation
MindFusion.Diagramming.Animations.ItemAnimation
MindFusion.Diagramming.Animations.BounceAnimation
MindFusion.Diagramming.Animations.FillColorAnimation
MindFusion.Diagramming.Animations.OpacityAnimation
MindFusion.Diagramming.Animations.PathAnimation
MindFusion.Diagramming.Animations.PulseAnimation
MindFusion.Diagramming.Animations.ShakeAnimation
MindFusion.Diagramming.Animations.StrokeColorAnimation

See Also

ItemAnimation Members
MindFusion.Diagramming.Animations Namespace

ItemAnimation Members

Base class of diagram item animations.

The following tables list the members exposed by the ItemAnimation type.

Public Constructors

	Name	Description
	ItemAnimation	Initializes a new instance of the ItemAnimation class.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

ItemAnimation Class
MindFusion.Diagramming.Animations Namespace

ItemAnimation Constructor

Initializes a new instance of the ItemAnimation class.

Namespace: MindFusion.Diagramming.Animations
File: ItemAnimation.js

Syntax

```
JavaScript
function ItemAnimation ([options])
```

Parameters

options
Optional.
Object. An object used to specify various property values for the animation.

See Also

ItemAnimation Class
MindFusion.Diagramming.Animations Namespace

OpacityAnimation Class

Animates items' opacity for fade-in and fade-out effects.

Syntax

```
JavaScript
// class
OpacityAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Example

```
JavaScript
node.animate(
    new OpacityAnimation(
        {
            fromValue: 1,
            toValue: 0.5,
            duration: 250,
            repeat: true,
            repeatCount: 4,
            reverse: true,
            animationType: Animations.AnimationType.BackBow
        }));
```

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.OpacityAnimation

See Also

OpacityAnimation Members
MindFusion.Diagramming.Animations Namespace

OpacityAnimation Members

Animates items' opacity for fade-in and fade-out effects.

The following tables list the members exposed by the OpacityAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)

	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

OpacityAnimation Class
MindFusion.Diagramming.Animations Namespace

PathAnimation Class

Animates node positions along specified path.

Syntax

JavaScript

```
// class
PathAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Example

JavaScript

```
node.animate(
    new PathAnimation(
        [
            new Point(30, 30),
            new Point(50, 30),
            new Point(75, 66),
            new Point(125, 88)
        ],
        {
            keepLastValue: true
        }));
```

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.PathAnimation

See Also

PathAnimation Members
MindFusion.Diagramming.Animations Namespace

PathAnimation Members

Animates node positions along specified path.

The following tables list the members exposed by the PathAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

PathAnimation Class
MindFusion.Diagramming.Animations Namespace

PulseAnimation Class

Animates node's local scale factor.

Syntax

JavaScript

```
// class
PulseAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Example

JavaScript

```
node.animate(
    new PulseAnimation(
        {
            fromValue: 1,
            toValue: 1.2,
            duration: 250,
            repeat: true,
            repeatCount: 4,
            reverse: true,
            animationType: Animations.AnimationType.BackBow
        }));
```

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.PulseAnimation

See Also

PulseAnimation Members
MindFusion.Diagramming.Animations Namespace

PulseAnimation Members

Animates node's local scale factor.

The following tables list the members exposed by the PulseAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)

	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

PulseAnimation Class
MindFusion.Diagramming.Animations Namespace

ScrollAnimation Class

Animates the scroll position of a diagram view.

Syntax

JavaScript

```
// class
ScrollAnimation.prototype = {}
```

Remarks

Call view's animate method to run an animation.

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ViewAnimation
    MindFusion.Diagramming.Animations.ScrollAnimation

See Also

ScrollAnimation Members
MindFusion.Diagramming.Animations Namespace

ScrollAnimation Members

Animates the scroll position of a diagram view.

The following tables list the members exposed by the ScrollAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description

	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

ScrollAnimation Class
MindFusion.Diagramming.Animations Namespace

ShakeAnimation Class

Animates horizontal position of nodes.

Syntax

JavaScript

```
// class
ShakeAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.ShakeAnimation

See Also

ShakeAnimation Members
MindFusion.Diagramming.Animations Namespace

ShakeAnimation Members

Animates horizontal position of nodes.

The following tables list the members exposed by the ShakeAnimation type.

Public Properties

	Name	Description

	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

ShakeAnimation Class
MindFusion.Diagramming.Animations Namespace

StrokeColorAnimation Class

Animates the color of node borders or link segments.

Syntax

JavaScript

```
// class
StrokeColorAnimation.prototype = {}
```

Remarks

Call item's animate method to run an animation.

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ItemAnimation
    MindFusion.Diagramming.Animations.StrokeColorAnimation

See Also

StrokeColorAnimation Members
MindFusion.Diagramming.Animations Namespace

StrokeColorAnimation Members

Animates the color of node borders or link segments.

The following tables list the members exposed by the StrokeColorAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

StrokeColorAnimation Class
MindFusion.Diagramming.Animations Namespace

ViewAnimation Class

Base class of diagram view animations.

Syntax

JavaScript

```
// class
ViewAnimation.prototype = {}
```

Remarks

Call view's animate method to run an animation.

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ViewAnimation
    MindFusion.Diagramming.Animations.ScrollAnimation
    MindFusion.Diagramming.Animations.ZoomAnimation

See Also

ViewAnimation Members
MindFusion.Diagramming.Animations Namespace

ViewAnimation Members

Base class of diagram view animations.

The following tables list the members exposed by the ViewAnimation type.

Public Constructors

	Name	Description
	ViewAnimation	Initializes a new instance of the ItemAnimation class.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

ViewAnimation Class
MindFusion.Diagramming.Animations Namespace

ViewAnimation Constructor

Initializes a new instance of the ItemAnimation class.

Namespace: MindFusion.Diagramming.Animations
File: ViewAnimation.js

Syntax

JavaScript

```
function ViewAnimation ([options])
```

Parameters

options
Optional.
Object. An object used to specify various property values for the animation.

See Also

ViewAnimation Class
MindFusion.Diagramming.Animations Namespace

ZoomAnimation Class

Animates the zoom level of a diagram view.

Syntax

JavaScript

```
// class
ZoomAnimation.prototype = {}
```

Remarks

Call view's animate method to run an animation.

Inheritance Hierarchy

MindFusion.Animations.Animation
  MindFusion.Diagramming.Animations.ViewAnimation
    MindFusion.Diagramming.Animations.ZoomAnimation

See Also

ZoomAnimation Members
MindFusion.Diagramming.Animations Namespace

ZoomAnimation Members

Animates the zoom level of a diagram view.

The following tables list the members exposed by the ZoomAnimation type.

Public Properties

	Name	Description
	animationType	Gets or sets the type of the animation. (Inherited from Animation.)
	duration	Gets or sets the duration of the animation function in milliseconds. (Inherited from Animation.)
	easingType	Gets or sets the easing type of the animation. (Inherited from Animation.)
	fromValue	Gets or sets the initial value of the property that will be animated. (Inherited from Animation.)
	isRunning	Indicates whether the animation is currently running. (Inherited from Animation.)
	param	Gets or sets an optional value passed to the animation function. (Inherited from Animation.)
	progress	Gets the progress of the current animation iteration. (Inherited from Animation.)
	repeat	Gets or sets a value indicating whether to restart the animation after it completes. (Inherited from Animation.)
	reverse	Gets or sets a value indicating whether to swap the from and to values after an animation loop completes. (Inherited from Animation.)
	toValue	Gets or sets the target value of the property that will be animated. (Inherited from Animation.)

Public Methods

	Name	Description
	addEventListener	Registers a single event listener on the **Animation**. (Inherited from Animation.)
	dispose	Sys.IDisposable implementation. (Inherited from Animation.)
	raiseEvent	Raises an event. (Inherited from Animation.)
	removeEventListener	Removes a single event listener attached to the **Animation**. (Inherited from Animation.)
	start	Starts or resumes the animation. (Inherited from Animation.)
	stop	Stops the animation. (Inherited from Animation.)

See Also

MindFusion.Diagramming.Controls Namespace

Events Class

Defines all events raised by the controls in the MindFusion.Diagrammming.Controls namespace.

Syntax

JavaScript

```
// class
Events.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Controls.Events

See Also

Events Members
MindFusion.Diagramming.Controls Namespace

Events Members

Defines all events raised by the controls in the MindFusion.Diagrammming.Controls namespace.

The following tables list the members exposed by the Events type.

Public Properties

	Name	Description
	pageAdded	Raised when the a page has been added to a TabbedDiagramView control.
	pageAdding	Raised before a page is added to a TabbedDiagramView control.
	pageChanged	Raised when the selected page in a TabbedDiagramView control has changed.
	pageChanging	Raised before the selected page in a TabbedDiagramView control is changed.
	pageRemoved	Raised when the a page has been removed from a TabbedDiagramView control.
	pageRemoving	Raised before a page is removed from a TabbedDiagramView control.

See Also

Events Properties

Events.pageAdded Property

Raised when the a page has been added to a TabbedDiagramView control.

Namespace: MindFusion.Diagramming.Controls
File: Events.js

Syntax

```JavaScript
get pageAdded() {}
set pageAdded(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming.Controls Namespace

Events.pageAdding Property

Raised before a page is added to a TabbedDiagramView control.

Namespace: MindFusion.Diagramming.Controls
File: Events.js

Syntax

```JavaScript
get pageAdding() {}
set pageAdding(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming.Controls Namespace

Events.pageChanged Property

Raised when the selected page in a TabbedDiagramView control has changed.

Namespace: MindFusion.Diagramming.Controls
File: Events.js

Syntax

JavaScript

```
get pageChanged() {}
set pageChanged(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming.Controls Namespace

Events.pageChanging Property

Raised before the selected page in a TabbedDiagramView control is changed.

Namespace: MindFusion.Diagramming.Controls
File: Events.js

Syntax

JavaScript

```
get pageChanging() {}
set pageChanging(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming.Controls Namespace

Events.pageRemoved Property

Raised when the a page has been removed from a TabbedDiagramView control.

Namespace: MindFusion.Diagramming.Controls
File: Events.js

Syntax

```
JavaScript
get pageRemoved() {}
set pageRemoved(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming.Controls Namespace

Events.pageRemoving Property

Raised before a page is removed from a TabbedDiagramView control.

Namespace: MindFusion.Diagramming.Controls
File: Events.js

Syntax

```
JavaScript
get pageRemoving() {}
set pageRemoving(value) {}
```

Property Value

String

A string containing the event name.

See Also

Events Members
Events Class
MindFusion.Diagramming.Controls Namespace

PageEventArgs Class

Contains the arguments passed to handlers of page-related events in a TabbedDiagramView.

Syntax

```
JavaScript
// class
PageEventArgs.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Controls.CancelEventArgs
MindFusion.Diagramming.Controls.PageEventArgs

See Also

PageEventArgs Members
MindFusion.Diagramming.Controls Namespace

PageEventArgs Members

Contains the arguments passed to handlers of page-related events in a TabbedDiagramView.

The following tables list the members exposed by the PageEventArgs type.

Public Constructors

	Name	Description
	PageEventArgs	Initializes a new instance of the PageEventArgs class.

Public Properties

	Name	Description
	oldPage	Gets the page before the event.
	page	Gets the page for which the event is raised.

See Also

PageEventArgs Class
MindFusion.Diagramming.Controls Namespace

PageEventArgs Constructor

Initializes a new instance of the PageEventArgs class.

Namespace: MindFusion.Diagramming.Controls
File: EventArgs.js

Syntax

JavaScript

```
function PageEventArgs (args)
```

Parameters

args

See Also

PageEventArgs Properties

PageEventArgs.oldPage Property

Gets the page before the event.

Namespace: MindFusion.Diagramming.Controls
File: EventArgs.js

Syntax

```
JavaScript
get oldPage() {}
set oldPage(value) {}
```

Property Value

DiagramPage

A DiagramPage instance.

See Also

PageEventArgs Members
PageEventArgs Class
MindFusion.Diagramming.Controls Namespace

PageEventArgs.page Property

Gets the page for which the event is raised.

Namespace: MindFusion.Diagramming.Controls
File: EventArgs.js

Syntax

```
JavaScript
get page() {}
set page(value) {}
```

Property Value

DiagramPage

A DiagramPage instance.

See Also

PageEventArgs Members
PageEventArgs Class
MindFusion.Diagramming.Controls Namespace

Palette Class

Represents a tool palette control that displays DiagramItem objects grouped into categories. Items from the palette can be added to a diagram by drag-and-drop.

Syntax

JavaScript

```
// class
Palette.prototype = {}
```

Remarks

Content Security Policy

To allow for stricter CSP rules, the library no longer loads CSS scripts automatically. If you use **Palette**, you must explicitly link the *common-ui.css* file from distribution's *themes* folder.

The **Palette** is an Accordion subclass that displays ItemListView components as its child panels. Each **ItemListView** displays the items of a palette category.

Call the addCategory method to define a category and create its respective accordion pane. Call addItem to add an item to specified category.

Drag-and-drop operations create a copy of the dragged item, and set its size to newInstanceSize. The copy is created by calling item's clone method. You can override **clone** to copy property values of custom item subclasses.

Icons in the list are drawn by scaling **newInstanceSize** down to iconSize. E.g. if a list icon is half the size of the diagram item it creates, its graphic elements such as cells, labels, strokes would be rendered at half scale too (as opposed to rendering with same label / stroke sizes but in smaller bounding rectangle). The scaling code assumes measureUnit of the target diagram and the palette have matching values.

Configuring the control

Palette comes as s a part of the diagramming-controls npm package or the MindFusion.Diagramming.Controls namespace for UMD users.

For npm users simply import the diagramming-controls package:

JavaScript

```
import * as Controls from '@mindfusion/diagramming-controls';
```

For UMD users, add references to files from distrbution's /umd folder using <script> tags. Note that the order of the scripts matters.

HTML

```
<!-- for core Diagramming -->
<script src="Scripts/collections.js" type="text/javascript"></script>
<script src="Scripts/drawing.js" type="text/javascript"></script>
<script src="Scripts/controls.js" type="text/javascript"></script>
<script src="Scripts/graphs.js" type="text/javascript"></script>
<script src="Scripts/animations.js" type="text/javascript"></script>
<script src="Scripts/diagramming.js" type="text/javascript"></script>

<!-- for Palette -->
<script src="Scripts/common.js" type="text/javascript"></script>
<script src="Scripts/common-collections.js" type="text/javascript"></script>
<script src="Scripts/common-ui.js" type="text/javascript"></script>
<script src="Scripts/diagramming-controls.js" type="text/javascript"></script>
```

Creating the control instance and adding it to an HTML page can be done by calling the static create method:

JavaScript

```
var palette = MindFusion.Diagramming.Controls.Palette
    .create(document.getElementById("palette"));
```

Default images for the collapse/expand buttons must be located in an ./icons folder. Alternatively, you can style the control with a predefined css theme, in which case the themes included in the distribution must be located in a ./themes folder and referenced in the head of the HTML document, e.g.

HTML / JavaScript

```
<link rel="stylesheet" type="text/css" href="themes/common-ui.css" />
<link rel="stylesheet" type="text/css" href="themes/business.css" />
...
palette.theme = "business";
```

Inheritance Hierarchy

MindFusion.Common.UI.Accordion
MindFusion.Diagramming.Controls.Palette

See Also

Palette Members
MindFusion.Diagramming.Controls Namespace

Palette Members

Represents a tool palette control that displays DiagramItem objects grouped into categories. Items from the palette can be added to a diagram by drag-and-drop.

The following tables list the members exposed by the Palette type.

Public Constructors

	Name	Description
	Palette	Initializes a new instance of the **Palette** class.

Public Properties

	Name	Description
	columns	Gets or sets the number of columns in palette's layout.
	iconSize	Gets or sets the size of items displayed in the palette.
	labelPosition	Gets or sets a value indicating the position of labels relative to item icons.
	measureUnit	Gets or sets the unit of measure used to specify item sizes.
	newInstanceSize	Gets or sets the default size of items created by drag-and-drop from this palette.

	padding	Gets or sets the padding space between item's slot bounds and its contents.
	textColor	Gets or sets the color used to paint item captions.

Public Methods

	Name	Description
	addCategory	Adds a new category to the palette.
	addItem	Adds an item to the palette.
S	create	Creates and initializes a new **Palette** from the specified element. This method is static and can be called without creating an instance of the class.
S	find	Returns the specified **Palette** object. This member is static and can be invoked without creating an instance of the class.
	removeCategory	Removes a category from the palette.
	removeItem	Removes an item from the palette.

See Also

Palette Class
MindFusion.Diagramming.Controls Namespace

Palette Constructor

Initializes a new instance of the Palette class.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function Palette (element)
```

Parameters

element
HTMLDivElement. Th DOM Element this Palette is associated with.

See Also

Palette Class
MindFusion.Diagramming.Controls Namespace

Palette Methods

Palette.addCategory Method

Adds a new category to the palette.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function addCategory (title, [image])
```

Parameters

title
String. The title text of the category.

image
Optional.
String. The URL of an image to be displayed in the category header.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace
addItem Method
removeCategory Method

Palette.addItem Method

Adds an item to the palette.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function addItem (item, category, [caption])
```

Parameters

item
DiagramItem. The item to add.

category
String. The category of the item.

caption
Optional.
String. The caption to display for the item.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

addCategory Method
removeItem Method

Palette.create Method

Creates and initializes a new Palette from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function create (element)
```

Parameters

element
HTMLDivElement. The DOM element that the Palette should be attached to.

Return Value

Palette. The newly created **Palette** object.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

Palette.find Method

Returns the specified Palette object. This member is static and can be invoked without creating an instance of the class.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function find (id)
```

Parameters

id
String. The id of the Palette's element.

Return Value

Palette. A **Palette** object with the specified id, if found, or **null** otherwise.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

Palette.removeCategory Method

Removes a category from the palette.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function removeCategory (title)
```

Parameters

title
String. The title of the category to remove.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace
addCategory Method

Palette.removeItem Method

Removes an item from the palette.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
function removeItem (item)
```

Parameters

item
DiagramItem. The item to remove.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace
addItem Method

Palette Properties

Palette.columns Property

Gets or sets the number of columns in palette's layout.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

```
JavaScript
get columns() {}
set columns(value) {}
```

Property Value

Number

An integer value specifying the number of columns. The default value is 1.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

Palette.iconSize Property

Gets or sets the size of items displayed in the palette.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

```
JavaScript
get iconSize() {}
set iconSize(value) {}
```

Property Value

Size

An instance of the Size class specifying item size.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace
newInstanceSize Property
measureUnit Property

Palette.labelPosition Property

Gets or sets a value indicating the position of labels relative to item icons.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
get labelPosition() {}
set labelPosition(value) {}
```

Property Value

LabelPosition

A member of the LabelPosition enumeration. The default value is **Auto**.

Remarks

The default Auto value places labels to the right of item icons if using one-column layout, and below icons if showing multiple columns.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

Palette.measureUnit Property

Gets or sets the unit of measure used to specify item sizes.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
get measureUnit() {}
set measureUnit(value) {}
```

Property Value

GraphicsUnit

A GraphicsUnit instance.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace
iconSize Property
newInstanceSize Property

Palette.newInstanceSize Property

Gets or sets the default size of items created by drag-and-drop from this palette.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
get newInstanceSize() {}
set newInstanceSize(value) {}
```

Property Value

Size

An instance of the Size class specifying item size.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace
iconSize Property
measureUnit Property

Palette.padding Property

Gets or sets the padding space between item's slot bounds and its contents.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

JavaScript

```
get padding() {}
set padding(value) {}
```

Property Value

Number

A number that specifies the padding space size.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

Palette.textColor Property

Gets or sets the color used to paint item captions.

Namespace: MindFusion.Diagramming.Controls
File: Palette.js

Syntax

```
JavaScript
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Palette Members
Palette Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner Class

Represents an interactive vector-drawing control that can be used to design node shapes.

Syntax

```
JavaScript
// class
ShapeDesigner.prototype = {}
```

Remarks

To allow for stricter Content Security Policy rules, the library no longer loads CSS scripts automatically. If you use **ShapeDesigner**, you must explicitly link the *common-ui.css* file from distribution's *themes* folder.

Inheritance Hierarchy

MindFusion.Diagramming.Controls.ShapeDesigner

See Also

ShapeDesigner Members
MindFusion.Diagramming.Controls Namespace

ShapeDesigner Members

Represents an interactive vector-drawing control that can be used to design node shapes.

The following tables list the members exposed by the ShapeDesigner type.

Public Constructors

	Name	Description
	ShapeDesigner	Initializes a new instance of the ShapeDesigner class.

Public Properties

	Name	Description
	alignToGrid	Gets or sets a value indicating whether the diagram items should be aligned to a grid.
	anchorPointsLabel	Gets or sets a string specifying the label of graphic AnchorPoints category in shape designer toolbox.
	gridSizeX	Gets or sets the horizontal distance between adjacent grid points.
	gridSizeY	Gets or sets the vertical distance between adjacent grid points.
	primitivesLabel	Gets or sets a string specifying the label of graphic primitives category in shape designer toolbox.
	selectedElement	Gets or sets the selected element in the shape designer.
	showGrid	Gets or sets a value indicating whether the alignment grid is visible.
	showShapeProperties	Gets or sets a value indicating whether to display the property panel.
	templatesLabel	Gets or sets a string specifying the label of outline templates category in shape designer toolbox.
	theme	Gets or sets the theme of the control.
	themesPath	Gets or sets the path to the control's css themes.

Public Methods

	Name	Description
	copyToClipboard	Copies the currently selected element to the clipboard.
S	create	Creates and initializes a new ShapeDesigner from the specified element. This method is static and can be called without creating an instance of the class.
	exportShape	Creates a new Shape from the current content of the design canvas.
	importShape	Imports a Shape into the designer's drawing canvas.
	pasteFromClipboard	Pastes a previously copied element from the clipboard.
	redo	Executes again the most recently undone user action.
	undo	Undoes the most recent user action.

Public Events

	Name	Description
	shapeChanged	Raised when the shape has changed.

See Also

ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner Constructor

Initializes a new instance of the ShapeDesigner class.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
function ShapeDesigner (element)
```

Parameters

element
HTMLDivElement. The DOM Element this ShapeDesigner is associated with.

See Also

ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner Methods

ShapeDesigner.copyToClipboard Method

Copies the currently selected element to the clipboard.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
function copyToClipboard ()
```

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.create Method

Creates and initializes a new ShapeDesigner from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
function create (element)
```

Parameters

element
HTMLDivElement. The DOM element that the ShapeDesigner should be attached to.

Return Value

ShapeDesigner. The newly created ShapeDesigner object.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.exportShape Method

Creates a new Shape from the current content of the design canvas.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
function exportShape (shapeId)
```

Parameters

shapeId
String. The shape's string identifier.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.importShape Method

Imports a Shape into the designer's drawing canvas.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
function importShape (shape)
```

Parameters

shape
Shape. A Shape instance.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.pasteFromClipboard Method

Pastes a previously copied element from the clipboard.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
function pasteFromClipboard ()
```

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.redo Method

Executes again the most recently undone user action.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
function redo ()
```

See Also

ShapeDesigner Members
ShapeDesigner Class

MindFusion.Diagramming.Controls Namespace

ShapeDesigner.undo Method

Undoes the most recent user action.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
function undo ()
```

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner Properties

ShapeDesigner.alignToGrid Property

Gets or sets a value indicating whether the diagram items should be aligned to a grid.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
get alignToGrid() {}
set alignToGrid(value) {}
```

Property Value

Boolean

true if the diagram items should be aligned to a grid, or false otherwise.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.anchorPointsLabel Property

Gets or sets a string specifying the label of graphic AnchorPoints category in shape designer toolbox.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
get anchorPointsLabel() {}
set anchorPointsLabel(value) {}
```

Property Value

String

The label of graphic AnchorPoints category.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.gridSizeX Property

Gets or sets the horizontal distance between adjacent grid points.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
get gridSizeX() {}
set gridSizeX(value) {}
```

Property Value

Number

The horizontal distance between adjacent grid points.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.gridSizeY Property

Gets or sets the vertical distance between adjacent grid points.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
get gridSizeY() {}
set gridSizeY(value) {}
```

Property Value

Number

The vertical distance between adjacent grid points.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.primitivesLabel Property

Gets or sets a string specifying the label of graphic primitives category in shape designer toolbox.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
get primitivesLabel() {}
set primitivesLabel(value) {}
```

Property Value

String

The label of graphic primitives category.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.selectedElement Property

Gets or sets the selected element in the shape designer.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
get selectedElement() {}
set selectedElement(value) {}
```

Property Value

InteractiveElement

An InteractiveElement instance.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.showGrid Property

Gets or sets a value indicating whether the alignment grid is visible.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
get showGrid() {}
set showGrid(value) {}
```

Property Value

Boolean

true if the alignment grid is visible, or false otherwise.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.showShapeProperties Property

Gets or sets a value indicating whether to display the property panel.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
get showShapeProperties() {}
set showShapeProperties(value) {}
```

Property Value

Boolean

true to display the property panel, or false otherwise. The default value is true.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.templatesLabel Property

Gets or sets a string specifying the label of outline templates category in shape designer toolbox.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
get templatesLabel() {}
set templatesLabel(value) {}
```

Property Value

String

The label of outline templates category.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.theme Property

Gets or sets the theme of the control.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

JavaScript

```
get theme() {}
set theme(value) {}
```

Property Value

Theme

An instance of the Theme class.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner.themesPath Property

Gets or sets the path to the control's css themes.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

```
JavaScript
get themesPath() {}
set themesPath(value) {}
```

Property Value

String

A string specifying the path.

Remarks

Only applies to web components.

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

ShapeDesigner Events

ShapeDesigner.shapeChanged Event

Raised when the shape has changed.

Namespace: MindFusion.Diagramming.Controls
File: ShapeDesigner.js

Syntax

See Also

ShapeDesigner Members
ShapeDesigner Class
MindFusion.Diagramming.Controls Namespace

TabAlignment Enumeration

Specifies the alignment of a tab strip relative to its container control.

Namespace: MindFusion.Diagramming.Controls
File: Enum.js

Syntax

```
JavaScript
// enum
TabAlignment = {}
```

See Also

MindFusion.Diagramming.Controls Namespace

TabbedDiagramView Class

Represents a view that displays DiagramDocument objects.

Syntax

JavaScript

```
// class
TabbedDiagramView.prototype = {}
```

Remarks

Content Security Policy

To allow for stricter CSP rules, the library no longer loads CSS scripts automatically. If you use **TabbedDiagramView**, you must explicitly link the *common-ui.css* file from distribution's *themes* folder.

Individual diagram pages can be activated through the tabs located at the top or at the bottom of the control. The appearance and behavior of the tab strip can be customized using various properties. The location and visibility of the strip is specified through the tabAlignment property. Its height can be changed via the tabsSize property. showAddButton and showNavigationButtons determine the visibility of the buttons for interactive tab creation and tab navigation respectively. showTabCloseButtons specifies whether the tabs can be closed interactively, by clicking on the small 'x' button displayed in each tab. allowReorder enables interactive tab rearrangement by mouse dragging.

Configuring the control

TabbedDiagramView comes as s a part of the diagramming-controls npm package or the MindFusion.Diagramming.Controls namespace for UMD users.

For npm users simply import the diagramming-controls package:

JavaScript

```
import * as Controls from '@mindfusion/diagramming-controls';
```

For UMD users, add references to files from distrbution's /umd folder using <script> tags. Note that the order of the scripts matters.

HTML

```
<!-- for core Diagramming -->
<script src="Scripts/collections.js" type="text/javascript"></script>
<script src="Scripts/drawing.js" type="text/javascript"></script>
<script src="Scripts/controls.js" type="text/javascript"></script>
<script src="Scripts/graphs.js" type="text/javascript"></script>
<script src="Scripts/animations.js" type="text/javascript"></script>
<script src="Scripts/diagramming.js" type="text/javascript"></script>

<!-- for TabbedDiagramView -->
<script src="Scripts/common.js" type="text/javascript"></script>
<script src="Scripts/common-collections.js" type="text/javascript"></script>
<script src="Scripts/common-ui.js" type="text/javascript"></script>
<script src="Scripts/diagramming-controls.js" type="text/javascript"></script>
```

Creating the control instance and adding it to an HTML page is the same as for the DiagramView control. You can pass a DiagramDocument instance to the create function, or omit that parameter to let the control create a new document with a single DiagramPage.

JavaScript

```
var doc = new MindFusion.Diagramming.DiagramDocument();
var page1 = new MindFusion.Diagramming.DiagramPage("diagram1");
var node = page1.factory.createShapeNode(10, 10, 20, 20);
doc.addPage(page1);
var diagramView = MindFusion.Diagramming.Controls.TabbedDiagramView.create(
    document.getElementById("diagram"), doc);
```

or simply

JavaScript

```
var diagramView = MindFusion.Diagramming.Controls.TabbedDiagramView.create(
    document.getElementById("diagram"));
```

You can specify a title, iconUrl and titleColor for diagram pages by setting the corresponding properties.

Default images for the navigation buttons must be located in an ./icons folder. Alternatively, you can style the control with a predefined css theme, in which case the themes included in the distribution must be located in a ./themes folder and referenced in the head of the HTML document, e.g.

HTML / JavaScript

```
<link rel="stylesheet" type="text/css" href="themes/common-ui.css" />
<link rel="stylesheet" type="text/css" href="themes/business.css" />
...
diagramView.theme = MindFusion.Diagramming.Theme.fromId("business");
```

Inheritance Hierarchy

MindFusion.Diagramming.DiagramView
MindFusion.Diagramming.Controls.TabbedDiagramView

See Also

TabbedDiagramView Members
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView Members

Represents a view that displays DiagramDocument objects.

The following tables list the members exposed by the TabbedDiagramView type.

Public Constructors

	Name	Description
	TabbedDiagramView	Initializes a new instance of the TabbedDiagramView class.

Public Properties

	Name	Description
	allowRenamePages	Gets or sets a value indicating whether DiagramPage titles can be renamed interactively.
	allowReorder	Gets or sets a value indicating whether tabs can be rearranged interactively.

	document	Gets or sets the diagram document displayed by this view.
	selectedIndex	Gets or sets the index of the currently selected diagram page.
	selectedPage	Gets or sets the currently selected tab page.
	showAddButton	Gets or sets a value indicating whether to display the 'Add Tab' button.
	showNavigationButtons	Gets or sets a value indicating whether to show tab navigation buttons.
	showTabCloseButtons	Gets or sets a value indicating whether to display a close button for each tab.
	tabAlignment	Gets or sets the tab alignment.
	tabsSize	Gets or sets the height of the tab strip.
	theme	Gets or sets the theme of the control.
	themesPath	Gets or sets the path to the control's css themes.

Public Methods

	Name	Description
S	create	Creates and initializes a new TabbedDiagramView from the specified element. This method is static and can be called without creating an instance of the class.
	dispose	DiagramView.dispose override.
	fromJson	Loads the diagram from a JSON string.
	initialize	DiagramView.initialize override.
	loadFromXml	Loads the diagram from an XML file.
	saveToXml	Saves the diagram to an XML file.
	setTheme	Sets the current theme of the control and its pages.
	toJson	Saves the diagram to a JSON string.

See Also

TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView Constructor

Initializes a new instance of the TabbedDiagramView class.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
function TabbedDiagramView (element, [document])
```

Parameters

element
HTMLCanvasElement. The Canvas DOM Element this TabbedDiagramView is associated with.

document
Optional.
DiagramDocument. The DiagramDocument instance this control is associated with.

See Also

TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView Methods

TabbedDiagramView.create Method

Creates and initializes a new TabbedDiagramView from the specified element. This method is static and can be called without creating an instance of the class.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
function create (element, [document])
```

Parameters

element
HTMLCanvasElement. The DOM element that the DiagramView should be attached to.

document
Optional.
DiagramDocument. The DiagramDocument instance this control is associated with.

Return Value

TabbedDiagramView. The newly created TabbedDiagramView object.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.dispose Method

DiagramView.dispose override.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
function dispose ()
```

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.fromJson Method

Loads the diagram from a JSON string.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
function fromJson ()
```

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.initialize Method

DiagramView.initialize override.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
function initialize ()
```

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.loadFromXml Method

Loads the diagram from an XML file.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
function loadFromXml (fileUrl, [onLoad, [onError]])
```

Parameters

fileUrl
String. The URL of an XML file where the data should be read from.

onLoad
Optional.
function. A callback that should be invoked if the file is loaded successfully.

onError
Optional.
function. A callback that should be invoked if the file could not be downloaded.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.saveToXml Method

Saves the diagram to an XML file.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
function saveToXml (url)
```

Parameters

url
String. A URL specifying where the diagram's XML should be posted to.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.setTheme Method

Sets the current theme of the control and its pages.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
function setTheme (theme, [setToPages])
```

Parameters

theme
Theme. The theme.

setToPages
Optional.
Boolean. true to set pages' themes, otherwise false.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.toJson Method

Saves the diagram to a JSON string.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
function toJson ()
```

Return Value

String. A string containing the diagram's JSON representation.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView Properties

TabbedDiagramView.allowRenamePages Property

Gets or sets a value indicating whether DiagramPage titles can be renamed interactively.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
get allowRenamePages() {}
set allowRenamePages(value) {}
```

Property Value

Boolean

true if tabs can be renamed interactively, or **false** otherwise. The default value is **false**.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.allowReorder Property

Gets or sets a value indicating whether tabs can be rearranged interactively.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
get allowReorder() {}
set allowReorder(value) {}
```

Property Value

Boolean

true if tabs can be rearranged interactively, or false otherwise. The default value is false.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.document Property

Gets or sets the diagram document displayed by this view.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get document() {}
set document(value) {}
```

Property Value

DiagramDocument

The DiagramDocument instance displayed by this view.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.selectedIndex Property

Gets or sets the index of the currently selected diagram page.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get selectedIndex() {}
set selectedIndex(value) {}
```

Property Value

Number

The index of the selected diagram page.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.selectedPage Property

Gets or sets the currently selected tab page.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get selectedPage() {}
set selectedPage(value) {}
```

Property Value

DiagramPage

The selected diagram page.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.showAddButton Property

Gets or sets a value indicating whether to display the 'Add Tab' button.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get showAddButton() {}
set showAddButton(value) {}
```

Property Value

Boolean

true if the 'Add Tab' button is visible, or false otherwise. The default value is false.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.showNavigationButtons Property

Gets or sets a value indicating whether to show tab navigation buttons.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get showNavigationButtons() {}
set showNavigationButtons(value) {}
```

Property Value

Boolean

true if the tab navigation buttons are visible, or false otherwise. The default value is false.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.showTabCloseButtons Property

Gets or sets a value indicating whether to display a close button for each tab.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get showTabCloseButtons() {}
set showTabCloseButtons(value) {}
```

Property Value

Boolean

true if the tabs close buttons are visible, or false otherwise. The default value is false.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.tabAlignment Property

Gets or sets the tab alignment.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get tabAlignment() {}
set tabAlignment(value) {}
```

Property Value

TabAlignment

A member of the TabAlignment enumeration.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.tabsSize Property

Gets or sets the height of the tab strip.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get tabsSize() {}
set tabsSize(value) {}
```

Property Value

Number

The height.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.theme Property

Gets or sets the theme of the control.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

JavaScript

```
get theme() {}
set theme(value) {}
```

Property Value

Theme

An instance of the Theme class.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

TabbedDiagramView.themesPath Property

Gets or sets the path to the control's css themes.

Namespace: MindFusion.Diagramming.Controls
File: TabbedDiagramView.js

Syntax

```
JavaScript
get themesPath() {}
set themesPath(value) {}
```

Property Value

String

A string specifying the path.

Remarks

Only applies to web components.

See Also

TabbedDiagramView Members
TabbedDiagramView Class
MindFusion.Diagramming.Controls Namespace

WebComponents Class

Implements registration of web components.

Syntax

```
JavaScript
// class
WebComponents.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Controls.WebComponents

See Also

WebComponents Members
MindFusion.Diagramming.Controls Namespace

WebComponents Members

Implements registration of web components.

The following tables list the members exposed by the WebComponents type.

Public Methods

	Name	Description
	register	Registers the controls in the package as web components.

See Also

WebComponents Class
MindFusion.Diagramming.Controls Namespace

WebComponents Methods

WebComponents.register Method

Registers the controls in the package as web components.

Namespace: MindFusion.Diagramming.Controls
File: WebComponents.js

Syntax

JavaScript

```
function register ()
```

Remarks

Call this method to register the TabbedDiagramView class as a web component. After registration, the browser instantiates **TabbedDiagramView** objects for <mindfusion-tabbeddiagramview> HTML tags. You can get reference to objects created by the browser by calling the find method.

See Also

WebComponents Members
WebComponents Class
MindFusion.Diagramming.Controls Namespace
create Method

MindFusion.Diagramming.Lanes Namespace

Cell Class

Represents a single cell in a lane grid.

Syntax

JavaScript

```
// class
Cell.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Lanes.Cell

See Also

Cell Members

Represents a single cell in a lane grid.

The following tables list the members exposed by the Cell type.

Public Constructors

	Name	Description
	Cell	Initializes a new instance of the **Cell** class. For internal use only.

Public Properties

	Name	Description
	style	Gets or sets the style of the cell.
	text	Gets or sets the text of the cell.

Public Methods

	Name	Description
	loadFromXml	Deserializes the cell from the specified XML element
	saveToXml	Serializes the cell matrix to the specified XML element

See Also

Cell Class
MindFusion.Diagramming.Lanes Namespace

Cell Constructor

Initializes a new instance of the Cell class. For internal use only.

Namespace: MindFusion.Diagramming.Lanes
File: Cell.js

Syntax

JavaScript
`function Cell ()`

See Also

Cell Class
MindFusion.Diagramming.Lanes Namespace

Cell Methods

Cell.loadFromXml Method

Deserializes the cell from the specified XML element

Namespace: MindFusion.Diagramming.Lanes
File: Cell.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Cell Members
Cell Class
MindFusion.Diagramming.Lanes Namespace

Cell.saveToXml Method

Serializes the cell matrix to the specified XML element

Namespace: MindFusion.Diagramming.Lanes
File: Cell.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Cell Members
Cell Class
MindFusion.Diagramming.Lanes Namespace

Cell Properties

Cell.style Property

Gets or sets the style of the cell.

Namespace: MindFusion.Diagramming.Lanes
File: Cell.js

Syntax

```
JavaScript
get style() {}
set style(value) {}
```

Property Value

Lanes.Style

A Style object that specifies the style of this cell.

See Also

Cell Members
Cell Class
MindFusion.Diagramming.Lanes Namespace

Cell.text Property

Gets or sets the text of the cell.

Namespace: MindFusion.Diagramming.Lanes
File: Cell.js

Syntax

```
JavaScript
get text() {}
set text(value) {}
```

Property Value

String

A string specifying the text of the cell.

See Also

Cell Members
Cell Class
MindFusion.Diagramming.Lanes Namespace

Grid Class

Represents a lane grid.

Syntax

JavaScript
`// class` `Grid.prototype = {}`

Inheritance Hierarchy

MindFusion.Diagramming.Lanes.Grid

See Also

Grid Members
MindFusion.Diagramming.Lanes Namespace

Grid Members

Represents a lane grid.

The following tables list the members exposed by the Grid type.

Public Constructors

	Name	Description
	Grid	Initializes a new instance of the **Grid** class. For internal use only.

Public Properties

	Name	Description
	allowInplaceEdit	Gets or sets a value indicating whether grid cells' text can be edited interactively.
	allowResizeHeaders	Gets or sets a value indicating whether users can resize the headers of the grid interactively.
	columnCount	Gets or sets the number of columns in the grid.
	columnHeaderBounds	Returns the bounding rectangle of all column headers.
	columnHeaders	Gets the collection of topmost column headers.
	columnHeadersHeights	Gets or sets an array of numbers specifying the heights of individual column levels.
	headersOnTop	Gets or sets a value indicating whether lane headers appear at the top of the z-order above all diagram items.
	hookHeaders	Gets or sets a value indicating whether headers will be anchored to their corresponding side.
	leftMargin	Gets or sets the offset of the lane grid from left side of diagram's bounds.
	minHeaderSize	Gets or sets the minimum size of a header.

	rowCount	Gets or sets the number of rows in the grid.
	rowHeaderBounds	Returns the bounding rectangle of all row headers.
	rowHeaders	Gets the collection of topmost row headers.
	rowHeadersWidths	Gets or sets an array of numbers specifying the widths of individual row levels.
	style	Gets the default style of the grid.
	topLeftAreaStyle	Gets or sets the style of the area above all row headers and to the left of all column headers.
	topLeftAreaText	Gets or sets the text displayed in the top left area.
	topMargin	Gets or sets the offset of the lane grid from top side of diagram's bounds.
	totalColumnHeight	Gets the total height of all column headers.
	totalRowWidth	Gets the total width of all row headers.

Public Methods

	Name	Description
	addColumnHeader	Adds the specified header to the grid as a new column header.
	addRowHeader	Adds the specified header to the grid as a new row header.
	findColumn	Finds the first column whose header caption matches the specified string.
	findRow	Finds the first row whose header caption matches the specified string.
	get	Gets the cell or the cluster of cells corresponding to the specified row and column.
	getCellBounds	Returns the bounding rectangle of the specified cell.
	getCellColumn	Returns the column index of the specified cell.
	getCellFromPoint	Gets the cell located at the specified point.
	getCellRow	Returns the row index of the specified cell.
	getColumn	Returns the column header with the specified index from the bottommost column header level.
	getColumnDepth	Gets the number of levels of the column headers.
	getColumnIndex	Returns the 0-based index of the specified column. Only works for bottommost headers. If the specified column is a group header, returns -1.
	getHeaderBounds	Returns the bounding rectangle of the specified header.
	getHeaderFromPoint	Gets the Header that is located at the specified point.

	getRow	Returns the row header with the specified index from the bottommost row header level.
	getRowDepth	Gets the number of levels of the row headers.
	getRowIndex	Returns the 0-based index of the specified row. Only works for bottommost headers. If the specified row is a group header, returns -1;
	loadFromXml	Deserializes the lane grid from the specified XML element.
	removeColumnHeader	Removes the specified column header and all of its associated cells from the grid.
	removeRowHeader	Removes the specified row header and all of its associated cells from the grid.
	saveToXml	Serializes the lane grid to the specified XML element.

See Also

Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid Constructor

Initializes a new instance of the Grid class. For internal use only.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function Grid ()
```

See Also

Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid Methods

Grid.addColumnHeader Method

Adds the specified header to the grid as a new column header.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function addColumnHeader (header, [parent, [i]])
```

Parameters

header
Lanes.Header. The Header object to add.

parent
Optional.
Lanes.Header. The parent header to add to, or **null** to add the new header as a root column.

i
Optional.
Number. The index at which the header was added in the collection.

Example

The following example adds a new header at the end of the column collection of an existing lane grid.

JavaScript

```
grid.addColumnHeader(new Header());
```

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.addRowHeader Method

Adds the specified header to the grid as a new row header.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function addRowHeader (header, [parent, [i]])
```

Parameters

header
Lanes.Header. The Header object to add.

parent
Optional.
Lanes.Header. The parent header to add to, or **null** to add the new header as a root row.

i
Optional.
Number. The index at which the header was added in the collection.

Example

The following example adds a new header at the end of the row collection of an existing lane grid.

JavaScript

```
d.addRowHeader(new Header());
```

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.findColumn Method

Finds the first column whose header caption matches the specified string.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function findColumn (title)
```

Parameters

title
String. A string containing the header title to look for.

Return Value

Returns the first column Header that has the specified caption.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.findRow Method

Finds the first row whose header caption matches the specified string.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function findRow (title)
```

Parameters

title
String. A string containing the header title to look for.

Return Value

Returns the first row Header that has the specified caption.

See Also

Grid.get Method

Gets the cell or the cluster of cells corresponding to the specified row and column.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
function get (column, row)
```

Parameters

column
Number | Lanes.Header. A Header object or a cell index.

row
Number | Lanes.Header. A **Header** object or a cell index.

Return Value

Lanes.Cell. A Cell object representing the cell or cluster of cells with the specified coordinates.

Remarks

This method accepts either Header objects or numbers as arguments, but not a mix of both. If more than one cell corresponds to specified **Header** object, this method returns an object identifying all corresponding cells. Modifying the style of the returned object will affect all cells in the cluster. Calling the method without arguments, or with **null** arguments, will return an object identifying all cells in the grid.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getCellBounds Method

Returns the bounding rectangle of the specified cell.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
function getCellBounds (cell)
```

Parameters

cell
Lanes.Cell. A Cell object representing a cell or a group of cells in the grid.

Return Value

Rect. A Rect instance representing the rectangle that bounds the specified cell or group of cells.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getCellColumn Method

Returns the column index of the specified cell.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getCellColumn (cell)
```

Parameters

cell
Lanes.Cell. A Cell object representing a single cell in the grid.

Return Value

Number. The index of the column that contains cell; -1 if cell is not found or refers to a group of cells.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getCellFromPoint Method

Gets the cell located at the specified point.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getCellFromPoint (point, cellBounds)
```

Parameters

point

A Point instance specifying a point within the grid.

cellBounds
An object that receives the coordinates of the found cell in its **cellBounds** field.

Return Value

A Cell object identifying the found cell, or **null** if there is no cell located at the specified point.

Example

The following example illustrates how to find the cell at a specific position as well as its bounding rectangle:

JavaScript

```
var boundsResult = {};
var cell = grid.getCellFromPoint(new Point(50, 50), boundsResult);
if (cell)
{
    var cellBounds = boundsResult.cellBounds;
}
```

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getCellRow Method

Returns the row index of the specified cell.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getCellRow (cell)
```

Parameters

cell
Lanes.Cell. A Cell object representing a single cell in the grid.

Return Value

Number. The index of the row that contains cell; -1 if cell is not found or refers to a group of cells.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getColumn Method

Returns the column header with the specified index from the bottommost column header level.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getColumn (index)
```

Parameters

index
Number. A number specifying the column index.

Return Value

A Header object representing the column at the specified index.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getColumnDepth Method

Gets the number of levels of the column headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getColumnDepth ()
```

Return Value

Number. A number representing the column depth.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getColumnIndex Method

Returns the 0-based index of the specified column. Only works for bottommost headers. If the specified column is a group header, returns -1.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
function getColumnIndex (column)
```

Parameters

column
Lanes.Header. A Header instance that represents a grid column.

Return Value

Number. The index of column, or -1 if the specified header represents a row or a group of child columns.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getHeaderBounds Method

Returns the bounding rectangle of the specified header.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
function getHeaderBounds (header, includeSubHeaders)
```

Parameters

header
Lanes.Header. The Header object whose coordinates to return.

includeSubHeaders
Boolean. **true** to include the coordinates of the child headers in the returned rectangle, or **false** otherwise.

Return Value

Rect. A Rect that bounds the specified header, and optionally its child headers.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getHeaderFromPoint Method

Gets the Header that is located at the specified point.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getHeaderFromPoint (point, headerBounds, scrollPosition)
```

Parameters

point
A Point instance specifying a point within the grid.

headerBounds
An object that receives the coordinates of the found header in its **headerBounds** field.

scrollPosition
Point. When the headers are hooked this parameter should provide information about the current scroll position in the view.

Return Value

The found Header object, or **null** if there is no header located at the specified point.

Example

The following example illustrates how to find the header at a specific position as well as its bounding rectangle:

JavaScript

```
var boundsResult = {};
var header = grid.getHeaderFromPoint(new Point(20, 10), boundsResult);
if (header)
{
    var headerBounds = boundsResult.headerBounds;
}
```

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getRow Method

Returns the row header with the specified index from the bottommost row header level.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getRow (index)
```

Parameters

index
Number. A number specifying the row index.

Return Value

A Header object representing the row at the specified index.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getRowDepth Method

Gets the number of levels of the row headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getRowDepth ()
```

Return Value

Number. A number representing the row depth.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.getRowIndex Method

Returns the 0-based index of the specified row. Only works for bottommost headers. If the specified row is a group header, returns -1;

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function getRowIndex (row)
```

Parameters

row
Lanes.Header. A Header instance that represents a grid row.

Return Value

Number. The index of row, or -1 if the specified header represents a column or a group of child rows.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.loadFromXml Method

Deserializes the lane grid from the specified XML element.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the deserialization process and some helper deserialization methods.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.removeColumnHeader Method

Removes the specified column header and all of its associated cells from the grid.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function removeColumnHeader (header)
```

Parameters

header
Lanes.Header. The Header object to remove.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.removeRowHeader Method

Removes the specified row header and all of its associated cells from the grid.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function removeRowHeader (header)
```

Parameters

header
Lanes.Header. The Header object to remove.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.saveToXml Method

Serializes the lane grid to the specified XML element.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid Properties

Grid.allowInplaceEdit Property

Gets or sets a value indicating whether grid cells' text can be edited interactively.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get allowInplaceEdit() {}
set allowInplaceEdit(value) {}
```

Property Value

Boolean

true if grid cells' text can be in-place edited, or **false** otherwise.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.allowResizeHeaders Property

Gets or sets a value indicating whether users can resize the headers of the grid interactively.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get allowResizeHeaders() {}
set allowResizeHeaders(value) {}
```

Property Value

Boolean

true if grid headers can resized interactively, or **false** otherwise.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.columnCount Property

Gets or sets the number of columns in the grid.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get columnCount() {}
set columnCount(value) {}
```

Property Value

Number

The number of columns.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.columnHeaderBounds Property

Returns the bounding rectangle of all column headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get columnHeaderBounds() {}
```

Property Value

Rect

A Rect that bounds all column headers.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.columnHeaders Property

Gets the collection of topmost column headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
```

```
get columnHeaders() {}
```

Property Value

Array

An array containing the topmost column headers.

Remarks

The returned array should not be modified directly. To add and remove column headers to the grid, use the addColumnHeader and removeColumnHeader methods respectively.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.columnHeadersHeights Property

Gets or sets an array of numbers specifying the heights of individual column levels.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get columnHeadersHeights() {}
set columnHeadersHeights(value) {}
```

Property Value

Array

An array of numbers representing the heights of the various header levels.

Remarks

If a value is non-positive or does not exist because the array has less arguments than the number of columns, then the height is calculated as the maximum height of the columns in that level.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.headersOnTop Property

Gets or sets a value indicating whether lane headers appear at the top of the z-order above all diagram items.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get headersOnTop() {}
set headersOnTop(value) {}
```

Property Value

Boolean

true to display headers on top of items, or **false** otherwise.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.hookHeaders Property

Gets or sets a value indicating whether headers will be anchored to their corresponding side.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get hookHeaders() {}
set hookHeaders(value) {}
```

Property Value

Boolean

true to anchor the headers, or **false** otherwise.

Remarks

If enabled, the column header stays anchored to the top side and the row header anchored to the left side of diagram's viewport, regardless of scroll position.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.leftMargin Property

Gets or sets the offset of the lane grid from left side of diagram's bounds.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get leftMargin() {}
set leftMargin(value) {}
```

Property Value

Number

A number representing the left margin.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.minHeaderSize Property

Gets or sets the minimum size of a header.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get minHeaderSize() {}
set minHeaderSize(value) {}
```

Property Value

Number

A number representing the minimum size for headers in the grid.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.rowCount Property

Gets or sets the number of rows in the grid.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get rowCount() {}
set rowCount(value) {}
```

Property Value

Number

The number of rows.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.rowHeaderBounds Property

Returns the bounding rectangle of all row headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get rowHeaderBounds() {}
```

Property Value

Rect

A Rect that bounds all row headers.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.rowHeaders Property

Gets the collection of topmost row headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get rowHeaders() {}
set rowHeaders(value) {}
```

Property Value

Array

An array containing the topmost row headers.

Remarks

The returned array should not be modified directly. To add and remove row headers to the grid, use the addRowHeader and removeRowHeader methods respectively.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.rowHeadersWidths Property

Gets or sets an array of numbers specifying the widths of individual row levels.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get rowHeadersWidths() {}
set rowHeadersWidths(value) {}
```

Property Value

Array

An array of numbers representing the widths of the various header levels.

Remarks

If a value is non-positive or does not exist because the array has less arguments than the number of rows, then the width is calculated as the maximum width of the rows in that level.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.style Property

Gets the default style of the grid.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get style() {}
```

Property Value

Lanes.Style

A Style object representing the style of all grid elements, which do not have an explicitly assigned style.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.topLeftAreaStyle Property

Gets or sets the style of the area above all row headers and to the left of all column headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get topLeftAreaStyle() {}
set topLeftAreaStyle(value) {}
```

Property Value

Lanes.Style

A Style object that specifies the appearance of the top-left area.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.topLeftAreaText Property

Gets or sets the text displayed in the top left area.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get topLeftAreaText() {}
set topLeftAreaText(value) {}
```

Property Value

String

A string containing the text.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.topMargin Property

Gets or sets the offset of the lane grid from top side of diagram's bounds.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get topMargin() {}
set topMargin(value) {}
```

Property Value

Number

A number representing the top margin.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.totalColumnHeight Property

Gets the total height of all column headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

JavaScript

```
get totalColumnHeight() {}
```

Property Value

Number

A number representing the height.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Grid.totalRowWidth Property

Gets the total width of all row headers.

Namespace: MindFusion.Diagramming.Lanes
File: Grid.js

Syntax

```
JavaScript
get totalRowWidth() {}
```

Property Value

Number

A number representing the width.

See Also

Grid Members
Grid Class
MindFusion.Diagramming.Lanes Namespace

Header Class

Represents a header in a lane grid.

Syntax

```
JavaScript
// class
Header.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Lanes.Header

See Also

Header Members
MindFusion.Diagramming.Lanes Namespace

Header Members

Represents a header in a lane grid.

The following tables list the members exposed by the Header type.

Public Constructors

	Name	Description
	Header	Initializes a new instance of the **Header** class.

Public Properties

	Name	Description

	height	Gets or sets the height of this header.
	isColumnHeader	Gets a value indicating whether this header is a column header.
	isRowHeader	Gets a value indicating whether this header is a row header.
	resizeType	Gets or sets how a row or a column is resized relatively to its parent.
	rotateTitle	Gets or sets a value indicating whether to rotate the header caption at 90 degrees.
	style	Gets or sets the style of this header.
	subHeaders	Gets a collection of the subheaders of this header.
	title	Gets or sets the header caption.
	width	Gets or sets the width of this header.

Public Methods

	Name	Description
	loadFromXml	Loads the lane header from the specified XML element.
	saveToXml	Serializes the lane header to the specified XML element

See Also

Header Class
MindFusion.Diagramming.Lanes Namespace

Header Constructor

Initializes a new instance of the Header class.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript
`function Header ()`

See Also

Header Class
MindFusion.Diagramming.Lanes Namespace

Header Methods

Header.loadFromXml Method

Loads the lane header from the specified XML element.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.saveToXml Method

Serializes the lane header to the specified XML element

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header Properties

Header.height Property

Gets or sets the height of this header.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

```
JavaScript
get height() {}
set height(value) {}
```

Property Value

Number

A number specifying the header height.

Remarks

If this is a row header, the property specifies the height of the whole row.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.isColumnHeader Property

Gets a value indicating whether this header is a column header.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

```
JavaScript
get isColumnHeader() {}
```

Property Value

Boolean

true if the header is a column header, or **false** otherwise.

Remarks

Both **isColumnHeader** and isRowHeader return **false** if the header is not attached to a lane grid.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.isRowHeader Property

Gets a value indicating whether this header is a row header.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

```
JavaScript
get isRowHeader() {}
```

Property Value

Boolean

true if the header is a row header, or **false** otherwise.

Remarks

Both isColumnHeader and **isRowHeader** return **false** if the header is not attached to a lane grid.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.resizeType Property

Gets or sets how a row or a column is resized relatively to its parent.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

```
JavaScript
get resizeType() {}
set resizeType(value) {}
```

Property Value

ResizeType

A member of the ResizeType enumeration.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.rotateTitle Property

Gets or sets a value indicating whether to rotate the header caption at 90 degrees.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
get rotateTitle() {}
set rotateTitle(value) {}
```

Property Value

Boolean

true if the title should be rotated, or **false** otherwise.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.style Property

Gets or sets the style of this header.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
get style() {}
set style(value) {}
```

Property Value

Style

A Style object that specifies the appearance of the header.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.subHeaders Property

Gets a collection of the subheaders of this header.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
get subHeaders() {}
```

Property Value

Array

An array containing the subheaders.

Remarks

The returned array should not be modified directly. To add and remove subheaders, use the addColumnHeader, addRowHeader, removeColumnHeader and removeRowHeader methods of the Grid respectively.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.title Property

Gets or sets the header caption.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
get title() {}
set title(value) {}
```

Property Value

String

A string containing the caption text.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

Header.width Property

Gets or sets the width of this header.

Namespace: MindFusion.Diagramming.Lanes
File: Header.js

Syntax

JavaScript

```
get width() {}
set width(value) {}
```

Property Value

Number

A number specifying the header width.

Remarks

If this is a column header, the property specifies the width of the whole column.

See Also

Header Members
Header Class
MindFusion.Diagramming.Lanes Namespace

ResizeType Enumeration

Specifies how to resize child columns and rows in the lanes grid when their parent column or row is resized.

Namespace: MindFusion.Diagramming.Lanes
File: ResizeType.js

Syntax

JavaScript

```
// enum
ResizeType = {}
```

Members

	Member name	Description
	Fixed	The header's size (width or height) is fixed and does not change when its parent header is being resized unless all its siblings have fixed size too.
	Proportional	The header resizes in proportion to its parent header.

See Also

MindFusion.Diagramming.Lanes Namespace

Style Class

Encapsulates the appearance properties of an element in a lane grid.

Syntax

JavaScript

```
// class
Style.prototype = {}
```

Inheritance Hierarchy

MindFusion.Diagramming.Lanes.Style

See Also

Style Members
MindFusion.Diagramming.Lanes Namespace

Style Members

Encapsulates the appearance properties of an element in a lane grid.

The following tables list the members exposed by the Style type.

Public Constructors

	Name	Description
	Style	Initializes a new instance of the **Style** class.

Public Properties

	Name	Description
	backgroundBrush	Gets or sets the background brush of this style.
	bottomBorderPen	Gets or sets the color used to stoke the bottom border of elements with this style.
	bottomBorderThickness	Gets or sets the width of the bottom border of elements with this style.
	fontName	Gets or sets the name of the font used to draw texts with this style.
	fontSize	Gets or sets the size of the font used to draw texts with this style.
	leftBorderPen	Gets or sets the color used to stoke the left border of elements with this style.
	leftBorderThickness	Gets or sets the width of the left border of elements with this style.
	lineAlignment	Gets or sets a value indicating how to vertically align texts drawn with this style.

	rightBorderPen	Gets or sets the color used to stoke the right border of elements with this style.
	rightBorderThickness	Gets or sets the width of the right border of elements with this style.
	textAlignment	Gets or sets a value indicating how to horizontally align texts drawn with this style.
	textColor	Gets or sets the color used to draw texts with this style.
	topBorderPen	Gets or sets the color used to stoke the top border of elements with this style.
	topBorderThickness	Gets or sets the width of the top border of elements with this style.

Public Methods

	Name	Description
	loadFromXml	Loads the style from the specified XML element
	saveToXml	Serializes the lane style to the specified XML element

See Also

Style Class
MindFusion.Diagramming.Lanes Namespace

Style Constructor

Initializes a new instance of the Style class.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript
`function Style ()`

See Also

Style Class
MindFusion.Diagramming.Lanes Namespace

Style Methods

Style.loadFromXml Method

Loads the style from the specified XML element

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
function loadFromXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.saveToXml Method

Serializes the lane style to the specified XML element

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
function saveToXml (xmlElement, context)
```

Parameters

xmlElement
Element. An XML DOM element that will contain the item's data.

context
XmlPersistContext. An object providing contextual information about the serialization process and some helper serialization methods.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style Properties

Style.backgroundBrush Property

Gets or sets the background brush of this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
get backgroundBrush() {}
set backgroundBrush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.bottomBorderPen Property

Gets or sets the color used to stoke the bottom border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
get bottomBorderPen() {}
set bottomBorderPen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.bottomBorderThickness Property

Gets or sets the width of the bottom border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
get bottomBorderThickness() {}
set bottomBorderThickness(value) {}
```

Property Value

Number

A number specifying the border width.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.fontName Property

Gets or sets the name of the font used to draw texts with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get fontName() {}
set fontName(value) {}
```

Property Value

String

A string specifying font name.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.fontSize Property

Gets or sets the size of the font used to draw texts with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get fontSize() {}
set fontSize(value) {}
```

Property Value

Number

A number specifying font size in current measure unit.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.leftBorderPen Property

Gets or sets the color used to stoke the left border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get leftBorderPen() {}
set leftBorderPen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.leftBorderThickness Property

Gets or sets the width of the left border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get leftBorderThickness() {}
set leftBorderThickness(value) {}
```

Property Value

Number

A number specifying the border width.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.lineAlignment Property

Gets or sets a value indicating how to vertically align texts drawn with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get lineAlignment() {}
set lineAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.rightBorderPen Property

Gets or sets the color used to stoke the right border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get rightBorderPen() {}
set rightBorderPen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.rightBorderThickness Property

Gets or sets the width of the right border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
get rightBorderThickness() {}
set rightBorderThickness(value) {}
```

Property Value

Number

A number specifying the border width.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.textAlignment Property

Gets or sets a value indicating how to horizontally align texts drawn with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
get textAlignment() {}
set textAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.textColor Property

Gets or sets the color used to draw texts with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

```
JavaScript
get textColor() {}
set textColor(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.topBorderPen Property

Gets or sets the color used to stoke the top border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get topBorderPen() {}
set topBorderPen(value) {}
```

Property Value

String

A string specifying the color name or hexadecimal value.

See Also

Style Members
Style Class
MindFusion.Diagramming.Lanes Namespace

Style.topBorderThickness Property

Gets or sets the width of the top border of elements with this style.

Namespace: MindFusion.Diagramming.Lanes
File: Style.js

Syntax

JavaScript

```
get topBorderThickness() {}
set topBorderThickness(value) {}
```

Property Value

Number

A number specifying the border width.

See Also

MindFusion.Drawing Namespace

Alignment Enumeration

Specifies the alignment of a graphics element relative to its layout rectangle.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
JavaScript
// enum
Alignment = {}
```

Members

	Member name	Description
	Center	The element is drawn at the center of the layout rectangle.
	Far	The element is drawn at the far edge of the layout rectangle (right or bottom).
	Near	The element is drawn at the near edge of the layout rectangle (left or top).

See Also

MindFusion.Drawing Namespace

Border3D Class

Represents a 3D border.

Syntax

```
JavaScript
// class
Border3D.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Border3D

See Also

Border3D Members
MindFusion.Drawing Namespace

Border3D Members

Represents a 3D border.

The following tables list the members exposed by the Border3D type.

Public Constructors

	Name	Description
	Border3D	Initializes a new instance of the Border3D class.

See Also

Border3D Class
MindFusion.Drawing Namespace

Border3D Constructor

Initializes a new instance of the Border3D class.

Namespace: MindFusion.Drawing
File: Border3D.js

Syntax

JavaScript

```
function Border3D (rect)
```

Parameters

rect
Rect. A Rect instance containing the border coordinates.

See Also

Border3D Class
MindFusion.Drawing Namespace

Component Class

Represents components in CompositeNode visual tree.

Syntax

JavaScript

```
// class
Component.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
  MindFusion.Drawing.Component
    MindFusion.Diagramming.Button
    MindFusion.Drawing.Container

See Also

Component Members
MindFusion.Drawing Namespace

Component Members

Represents components in CompositeNode visual tree.

The following tables list the members exposed by the Component type.

Public Constructors

	Name	Description
	Component	Initializes a new instance of the **Component** class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

See Also

Component Class
MindFusion.Drawing Namespace

Component Constructor

Initializes a new instance of the Component class.

Namespace: MindFusion.Drawing
File: Component.js

Syntax

JavaScript
`function Component ()`

See Also

Component Class
MindFusion.Drawing Namespace

ComponentBase Class

A base class for components in CompositeNode visual tree.

Syntax

JavaScript
`// class` `ComponentBase.prototype = {}`

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
MindFusion.Drawing.Component
MindFusion.Drawing.Ellipse
MindFusion.Drawing.Image
MindFusion.Drawing.Path
MindFusion.Drawing.Rect
MindFusion.Drawing.Text
MindFusion.Drawing.Video

See Also

ComponentBase Members
MindFusion.Drawing Namespace

ComponentBase Members

A base class for components in CompositeNode visual tree.

The following tables list the members exposed by the ComponentBase type.

Public Constructors

	Name	Description
	ComponentBase	Initializes a new instance of the **ComponentBase** class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in.

	gridRow	Gets or sets the index of the grid row this component is contained in.
	horizontalAlignment	Gets or sets the component's horizontal alignment.
	verticalAlignment	Gets or sets the component's vertical alignment.
	visibility	Gets or sets the visibility of this component.

See Also

ComponentBase Class
MindFusion.Drawing Namespace

ComponentBase Constructor

Initializes a new instance of the ComponentBase class.

Namespace: MindFusion.Drawing
File: ComponentBase.js

Syntax

JavaScript

```
function ComponentBase ()
```

See Also

ComponentBase Class
MindFusion.Drawing Namespace

ComponentBase Methods

ComponentBase Properties

ComponentBase.gridColumn Property

Gets or sets the index of the grid column this component is contained in.

Namespace: MindFusion.Drawing
File: ComponentBase.js

Syntax

JavaScript

```
get gridColumn() {}
set gridColumn(value) {}
```

Property Value

Number

The grid column index.

Remarks

This value is used when the component is placed inside a GridPanel.

See Also

ComponentBase Members
ComponentBase Class
MindFusion.Drawing Namespace

ComponentBase.gridRow Property

Gets or sets the index of the grid row this component is contained in.

Namespace: MindFusion.Drawing
File: ComponentBase.js

Syntax

JavaScript

```
get gridRow() {}
set gridRow(value) {}
```

Property Value

Number

The grid row index.

Remarks

This value is used when the component is placed inside a GridPanel.

See Also

ComponentBase Members
ComponentBase Class
MindFusion.Drawing Namespace

ComponentBase.horizontalAlignment Property

Gets or sets the component's horizontal alignment.

Namespace: MindFusion.Drawing
File: ComponentBase.js

Syntax

JavaScript

```
get horizontalAlignment() {}
set horizontalAlignment(value) {}
```

Property Value

LayoutAlignment

A member of the LayoutAlignment enumeration.

See Also

ComponentBase Members
ComponentBase Class
MindFusion.Drawing Namespace

ComponentBase.verticalAlignment Property

Gets or sets the component's vertical alignment.

Namespace: MindFusion.Drawing
File: ComponentBase.js

Syntax

JavaScript

```
get verticalAlignment() {}
set verticalAlignment(value) {}
```

Property Value

LayoutAlignment

A member of the LayoutAlignment enumeration.

See Also

ComponentBase Members
ComponentBase Class
MindFusion.Drawing Namespace

ComponentBase.visibility Property

Gets or sets the visibility of this component.

Namespace: MindFusion.Drawing
File: ComponentBase.js

Syntax

JavaScript

```
get visibility() {}
set visibility(value) {}
```

Property Value

Visibility

A member of the Visibility enumeration.

See Also

ComponentBase Members
ComponentBase Class

MindFusion.Drawing Namespace

Container Class

Represents container components in drawing tree.

Syntax

JavaScript

```
// class
Container.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
  MindFusion.Drawing.Component
    MindFusion.Drawing.Container
      MindFusion.Diagramming.Manipulator
      MindFusion.Diagramming.Panel

See Also

Container Members
MindFusion.Drawing Namespace

Container Members

Represents container components in drawing tree.

The following tables list the members exposed by the Container type.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

See Also

Container Class
MindFusion.Drawing Namespace

DashStyle Enumeration

Specifies the dash pattern of lines.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
// enum
DashStyle = {}
```

Members

	Member name	Description
	Custom	Custom pattern.
	Dash	Draw dashed line.
	DashDot	Draw dash-dot pattern.
	DashDotDot	Draw dash-dot-dot pattern.
	Dot	Draw dotted line.
	Solid	Draw solid line.

See Also

MindFusion.Drawing Namespace

DrawingUtils Class

Contains drawing-related helper functions.

Syntax

```
// class
DrawingUtils.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.DrawingUtils

See Also

DrawingUtils Members
MindFusion.Drawing Namespace

DrawingUtils Members

Contains drawing-related helper functions.

The following tables list the members exposed by the DrawingUtils type.

Public Methods

	Name	Description
	approximateBezier	Approximates a Bezier curve with the given quality and by a series of line segments.
	checkIntersect	Checks if the specified rectangle and circle intersect.
	degrees	Converts the specified value from radians to degrees.
	distToPolyline	Calculates the shortest distance from the specified point to the specified polyline, also returning the index of the segment the point is closest to.
	distToRectPoint	Returns the point that lies on the outline of a rectangle and is nearest to the given point.
	getCenter	Returns the center of the specified rectangle.
	getPointsBounds	Calculates the bounding rect of the specified polyline.
	getSegmentIntersection	Checks whether the segments defined by the specified point pairs intersect and returns the intersection point.
	inflate	Creates and returns an enlarged copy of a Rect instance.
	intersect	Determines whether the specified line segments intersect.
	minDistToRect	Calculates the minimum distance between a given point and a given rectangle.
	pointInCircle	Checks whether the specified circle contains the specified point.
	pointInEllipse	Checks whether the specified ellipse contains the specified point.
	pointInPolygon	Checks whether the specified rectangle contains the specified point.
	radians	Converts the specified value from degrees to radians.
	unionRects	Returns the smallest possible rectangle containing both of the specified rectangles.

See Also

DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils Methods

DrawingUtils.approximateBezier Method

Approximates a Bezier curve with the given quality and by a series of line segments.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function approximateBezier (points, quality, start)
```

Parameters

points
Array. A list of Bezier control points.

quality
Number. Specifies the approximation quality.

start
Number. Specifies the start point index.

Return Value

Array. A list of points that approximate the curve as a series of straight line segments.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.checkIntersect Method

Checks if the specified rectangle and circle intersect.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function checkIntersect (point, rect, rad)
```

Parameters

point
Point. The center of the circle.

rect
Rect. The rectangle.

rad
Number. The radius of the circle.

Return Value

Boolean. true if the rectangle and the circle intersect; otherwise, false.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.degrees Method

Converts the specified value from radians to degrees.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function degrees (radians)
```

Parameters

radians
Number. The radians to convert.

Return Value

Number. The converted angle.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.distToPolyline Method

Calculates the shortest distance from the specified point to the specified polyline, also returning the index of the segment the point is closest to.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function distToPolyline ()
```

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.distToRectPoint Method

Returns the point that lies on the outline of a rectangle and is nearest to the given point.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function distToRectPoint (point, rect)
```

Parameters

point
Point. The point to consider.

rect
Rect. The rectangle to consider.

Return Value

Number. The nearest point.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.getCenter Method

Returns the center of the specified rectangle.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function getCenter (rect)
```

Parameters

rect
Rect. The rectangle.

Return Value

Point. The center.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.getPointsBounds Method

Calculates the bounding rect of the specified polyline.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function getPointsBounds (points)
```

Parameters

points
Array. An array of points.

Return Value

Rect. The bounding rect.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.getSegmentIntersection Method

Checks whether the segments defined by the specified point pairs intersect and returns the intersection point.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function getSegmentIntersection (p1, p2, p3, p4)
```

Parameters

p1
Point. The start of the first segment.

p2
Point. The end of the first segment.

p3
Point. The start of the second segment.

p4
Point. The end of the second segment.

Return Value

Boolean. true if the segments intersect, false if not.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.inflate Method

Creates and returns an enlarged copy of a Rect instance.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function inflate (rect, x, y)
```

Parameters

rect
Rect. The Rect instance to inflate.

x
Number. The amount to inflate horizontally.

y
Number. The amount to inflate vertically.

Return Value

Rect. The inflated Rect.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.intersect Method

Determines whether the specified line segments intersect.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function intersect (p1, p2, p3, p4)
```

Parameters

p1
Point. The start of the first line segment.

p2
Point. The end of the first line segment.

p3
Point. The start of the second line segment.

p4
Point. The end of the second line segment.

Return Value

Boolean. true if the segments intersect, false if not.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.minDistToRect Method

Calculates the minimum distance between a given point and a given rectangle.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function minDistToRect (point, rect)
```

Parameters

point
Point. The point.

rect
Rectabgle. The rectangle.

Return Value

Number. The minimum distance.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.pointInCircle Method

Checks whether the specified circle contains the specified point.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function pointInCircle (point, center, radius)
```

Parameters

point
Point. The point to check.

center
Point. The center of the circle.

radius
Number. The radius of the circle.

Return Value

Boolean. true if the circle contains the specified point, false if not.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.pointInEllipse Method

Checks whether the specified ellipse contains the specified point.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function pointInEllipse (point, rect)
```

Parameters

point
Point. The point to check.

rect
Rect. The bounds of the ellipse.

Return Value

Boolean. true if the ellipse contains the specified point, false if not.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.pointInPolygon Method

Checks whether the specified rectangle contains the specified point.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function pointInPolygon (point, polygon)
```

Parameters

point
Point. The point to check.

polygon
Array. An array with the points that define the polygon.

Return Value

Boolean. true if the polygon contains the specified point, false if not.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.radians Method

Converts the specified value from degrees to radians.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function radians (degrees)
```

Parameters

degrees
Number. The degrees to convert.

Return Value

Number. The converted angle.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

DrawingUtils.unionRects Method

Returns the smallest possible rectangle containing both of the specified rectangles.

Namespace: MindFusion.Drawing
File: DrawingUtils.js

Syntax

JavaScript

```
function unionRects (rect1, rect2)
```

Parameters

rect1
Rect. The first rectangle.

rect2
Rect. The second rectangle.

Return Value

Rect. A Rect instance that represents the union of the specified arguments.

See Also

DrawingUtils Members
DrawingUtils Class
MindFusion.Drawing Namespace

Ellipse Class

Represents an Ellipse.

Syntax

JavaScript

```
// class
Ellipse.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
MindFusion.Drawing.Ellipse

See Also

Ellipse Members
MindFusion.Drawing Namespace

Ellipse Members

Represents an Ellipse.

The following tables list the members exposed by the Ellipse type.

Public Constructors

	Name	Description
	Ellipse	Initializes a new instance of the Ellipse class.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	height	Gets or sets the height of the ellipse.
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)
	width	Gets or sets the width of the ellipse.
	x	Gets or sets the x-coordinate of the upper-left corner of the ellipse.
	y	Gets or sets the y-coordinate of the upper-left corner of the ellipse.

Public Methods

	Name	Description
	clone	Creates a **Ellipse** object identical to the current object.

See Also

Ellipse Class
MindFusion.Drawing Namespace

Ellipse Constructor

Initializes a new instance of the Ellipse class.

Namespace: MindFusion.Drawing
File: Ellipse.js

Syntax

JavaScript

```
function Ellipse (x, y, width, height)
```

Parameters

x
Number. The X-coordinate of the top left corner of the Ellipse.

y
Number. The Y-coordinate of the top left corner of the Ellipse.

width
Number. The width of the Ellipse.

height
Number. The height of the Ellipse.

See Also

Ellipse Class
MindFusion.Drawing Namespace

Ellipse Methods

Ellipse.clone Method

Creates a Ellipse object identical to the current object.

Namespace: MindFusion.Drawing
File: Ellipse.js

Syntax

JavaScript

```
function clone ()
```

Return Value

The newly created Ellipse object.

See Also

Ellipse Members
Ellipse Class
MindFusion.Drawing Namespace

Ellipse Properties

Ellipse.height Property

Gets or sets the height of the ellipse.

Namespace: MindFusion.Drawing
File: Ellipse.js

Syntax

JavaScript

```
get height() {}
set height(value) {}
```

Property Value

Number

The height of the ellipse.

See Also

Ellipse Members
Ellipse Class
MindFusion.Drawing Namespace

Ellipse.width Property

Gets or sets the width of the ellipse.

Namespace: MindFusion.Drawing
File: Ellipse.js

Syntax

JavaScript

```
get width() {}
set width(value) {}
```

Property Value

Number

The width of the ellipse.

See Also

Ellipse Members
Ellipse Class
MindFusion.Drawing Namespace

Ellipse.x Property

Gets or sets the x-coordinate of the upper-left corner of the ellipse.

Namespace: MindFusion.Drawing
File: Ellipse.js

Syntax

JavaScript

```
get x() {}
set x(value) {}
```

Property Value

Number

The x-coordinate of the upper-left corner of the ellipse.

See Also

Ellipse Members
Ellipse Class
MindFusion.Drawing Namespace

Ellipse.y Property

Gets or sets the y-coordinate of the upper-left corner of the ellipse.

Namespace: MindFusion.Drawing
File: Ellipse.js

Syntax

JavaScript

```
get y() {}
set y(value) {}
```

Property Value

Number

The x-coordinate of the upper-left corner of the ellipse.

See Also

Ellipse Members
Ellipse Class
MindFusion.Drawing Namespace

Font Class

Represents a font.

Syntax

JavaScript

```
// class
Font.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Font

See Also

Font Members
MindFusion.Drawing Namespace

Font Members

Represents a font.

The following tables list the members exposed by the Font type.

Public Constructors

	Name	Description
	Font	Initializes a new instance of the Font class with the specified attributes.

Public Properties

	Name	Description
	bold	Gets whether the font is bold.
	fontStyle	Gets or sets the style of this font.
	italic	Gets whether the font is italic.
	name	Gets or sets the name of this font.
	size	Gets or sets the size of this font.
	underline	Gets whether the font is underlined.

Public Methods

	Name	Description
	apply	Applies the font properties to a CanvasRenderingContext2D drawing object.
S	fromObject	Returns a Font from a JSON object or string describing this font.
	toObject	Returns a JSON object describing this font.

See Also

Font Class
MindFusion.Drawing Namespace

Font Constructor

Initializes a new instance of the Font class with the specified attributes.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

JavaScript
`function Font (name, [size, [bold, [italic, [underline]]]])`

Parameters

name
String. The font name.

size
Optional.
Number. The font size.

bold
Optional.
Boolean | FontStyle. true if this font is bold, or false otherwise.

italic
Optional.
Boolean. true if this font is italic, or false otherwise.

underline
Optional.
Boolean. true if this font is underlined, or false otherwise.

See Also

Font Class
MindFusion.Drawing Namespace

Font Methods

Font.apply Method

Applies the font properties to a CanvasRenderingContext2D drawing object.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

JavaScript

```
function apply (context)
```

Parameters

context
CanvasRenderingContext2D. The CanvasRenderingContext2D drawing object.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font.fromObject Method

Returns a Font from a JSON object or string describing this font.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

```
JavaScript
function fromObject ()
```

Return Value

Font. The font object.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font.toObject Method

Returns a JSON object describing this font.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

```
JavaScript
function toObject ()
```

Return Value

Object. The object describing this font.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font Properties

Font.bold Property

Gets whether the font is bold.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

```
JavaScript
get bold() {}
```

Property Value

Boolean

true if Bold flag is set in fontStyle, or **false** otherwise.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font.fontStyle Property

Gets or sets the style of this font.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

JavaScript

```
get fontStyle() {}
set fontStyle(value) {}
```

Property Value

FontStyle

A combination of FontStyle enumeration flags.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font.italic Property

Gets whether the font is italic.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

JavaScript

```
get italic() {}
```

Property Value

Boolean

true if Italic flag is set in fontStyle, or **false** otherwise.

See Also

Font Members
Font Class

Font.name Property

Gets or sets the name of this font.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

```
JavaScript
get name() {}
set name(value) {}
```

Property Value

String

A string containing font name.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font.size Property

Gets or sets the size of this font.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

```
JavaScript
get size() {}
set size(value) {}
```

Property Value

Number

The font size.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

Font.underline Property

Gets whether the font is underlined.

Namespace: MindFusion.Drawing
File: Font.js

Syntax

```
JavaScript
get underline() {}
```

Property Value

Boolean

true if Underline flag is set in fontStyle, or **false** otherwise.

See Also

Font Members
Font Class
MindFusion.Drawing Namespace

FontStyle Enumeration

Specifies font style attributes.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
JavaScript
// enum
FontStyle = {}
```

Members

	Member name	Description
	Bold	Bold text.
	Italic	Italic text.
	Regular	Normal text.
	Strikeout	Striked text.
	Underline	Underlined text.

See Also

MindFusion.Drawing Namespace

GraphicsUnit Class

Specifies the unit of measure for length and size properties.

Syntax

```javascript
// class
GraphicsUnit.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.GraphicsUnit

See Also

GraphicsUnit Members
MindFusion.Drawing Namespace

GraphicsUnit Members

Specifies the unit of measure for length and size properties.

The following tables list the members exposed by the GraphicsUnit type.

Public Properties

	Name	Description
	Centimeter	Specifies the centimeter as the unit of measure.
	Display	Specifies the display unit as the unit of measure.
	Document	Specifies the document unit as the unit of measure.
	Inch	Specifies the inch as the unit of measure.
	Millimeter	Specifies the millimeter as the unit of measure.
	Percent	Specifies the world coordinate system unit as the unit of measure.
	Pixel	Specifies the pixel unit as the unit of measure.
	Point	Specifies a printer's point as the unit of measure.
	World	Specifies the world coordinate system unit as the unit of measure.
	WpfPoint	Specifies the WpfPoint unit as the unit of measure.

See Also

GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit Properties

GraphicsUnit.Centimeter Property

Specifies the centimeter as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

JavaScript

```
get Centimeter() {}
set Centimeter(value) {}
```

Property Value

Number

Defines 2.54 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Display Property

Specifies the display unit as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

JavaScript

```
get Display() {}
set Display(value) {}
```

Property Value

Number

Defines 100 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Document Property

Specifies the document unit as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

```
JavaScript
get Document() {}
set Document(value) {}
```

Property Value

Number

Defines 300 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Inch Property

Specifies the inch as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

```
JavaScript
get Inch() {}
set Inch(value) {}
```

Property Value

Number

Defines 1 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Millimeter Property

Specifies the millimeter as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

```
JavaScript
get Millimeter() {}
set Millimeter(value) {}
```

Property Value

Number

Defines 25.4 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Percent Property

Specifies the world coordinate system unit as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

JavaScript

```
get Percent() {}
set Percent(value) {}
```

Property Value

Number

For internal use.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Pixel Property

Specifies the pixel unit as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

JavaScript

```
get Pixel() {}
set Pixel(value) {}
```

Property Value

Number

Defines 96 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.Point Property

Specifies a printer's point as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

JavaScript

```
get Point() {}
set Point(value) {}
```

Property Value

Number

Defines 72 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.World Property

Specifies the world coordinate system unit as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

JavaScript

```
get World() {}
set World(value) {}
```

Property Value

Number

For internal use.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

GraphicsUnit.WpfPoint Property

Specifies the WpfPoint unit as the unit of measure.

Namespace: MindFusion.Drawing
File: GraphicsUnit.js

Syntax

```
JavaScript
get WpfPoint() {}
set WpfPoint(value) {}
```

Property Value

Number

Defines 96 DPI resolution.

See Also

GraphicsUnit Members
GraphicsUnit Class
MindFusion.Drawing Namespace

Image Class

Represents an image.

Syntax

```
JavaScript
// class
Image.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
MindFusion.Drawing.Image

See Also

Image Members
MindFusion.Drawing Namespace

Image Members

Represents an image.

The following tables list the members exposed by the Image type.

Public Constructors

	Name	Description

	Image	Initializes a new instance of the **Image** class.

Public Fields

	Name	Description
	bounds	The object's bounds.
	image	The image to be displayed.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

See Also

Image Class
MindFusion.Drawing Namespace

Image Fields

Image.bounds Field

The object's bounds.

Namespace: MindFusion.Drawing
File: Image.js

Syntax

JavaScript
`var bounds`

See Also

Image Members
Image Class
MindFusion.Drawing Namespace

Image.image Field

The image to be displayed.

Namespace: MindFusion.Drawing
File: Image.js

Syntax

```
JavaScript
var image
```

See Also

Image Members
Image Class
MindFusion.Drawing Namespace

Image Constructor

Initializes a new instance of the Image class.

Namespace: MindFusion.Drawing
File: Image.js

Syntax

```
JavaScript
function Image (bounds)
```

Parameters

bounds
Rect. The bounds of the image.

See Also

Image Class
MindFusion.Drawing Namespace

ImageAlign Enumeration

Specifies the position and alignment of a picture in a node, or that of the background image.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
JavaScript
// enum
ImageAlign = {}
```

Members

	Member name	Description
	BottomCenter	The image is centered horizontally and aligned to the bottom side.
	BottomLeft	The image is aligned to the bottom left corner of the node or the diagram.
	BottomRight	The image is aligned to the bottom right corner of the node or the diagram.
	Center	The image is centered in the node or diagram.
	Fit	The image is resized to fit the size of the object or the component's client area.
	FitBottom	The image is resized to fit the size of the object and aligned to bottom side.
	FitLeft	The image is resized to fit the size of the object and aligned to left side.
	FitRight	The image is resized to fit the size of the object and aligned to right side.
	FitTop	The image is resized to fit the size of the object and aligned to top side.
	MiddleLeft	The image is centered vertically and aligned to the left-hand side.
	MiddleRight	The image is centered vertically and aligned to the right-hand side.
	Stretch	The image is stretched to fill the object or the component's client area.
	Tile	The image is tiled to cover the node or the diagram.
	TopCenter	The image is centered horizontally and aligned to the top side.
	TopLeft	The image is aligned to the top left corner of the node or the diagram.
	TopRight	The image is aligned to the top right corner of the node or the diagram.

See Also

MindFusion.Drawing Namespace

LayoutAlignment Enumeration

Specifies alignment of components.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

JavaScript
`// enum` `LayoutAlignment = {}`

Members

	Member name	Description
	Center	Center alignment.
	Far	Right ot bottom alignment
	Near	Left or top alignment.
	Stretch	Stretch the component.

See Also

MindFusion.Drawing Namespace

LineJoin Enumeration

Specifies how to join consecutive line or curve segments in a figure.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

JavaScript
`// enum` `LineJoin = {}`

Members

	Member name	Description
	Bevel	Produces a diagonal corner.
	Miter	Produces a sharp corner or a clipped corner, depending on whether the length of the miter exceeds the miter limit.
	Round	Produces a smooth, circular arc between the lines.

See Also

MindFusion.Drawing Namespace

Path Class

Represents a path.

Syntax

JavaScript

```
// class
Path.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
  MindFusion.Drawing.Path

See Also

Path Members
MindFusion.Drawing Namespace

Path Members

Represents a path.

The following tables list the members exposed by the Path type.

Public Constructors

	Name	Description
	Path	Initializes a new instance of the **Path** class.

Public Properties

	Name	Description
	brush	Gets or sets the brush used to fill a closed path.
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	pen	Gets or sets the pen used to draw the path.
	text	Gets or sets the text displayed inside this path.
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	addEllipse	Adds an ellipse figure to the path.
	addRect	Adds a rectangle figure to the path.
	arcTo	Draws an arc.
	bezierTo	Draws a cubic bezier curve.
	clone	Creates a **Path** object identical to the current object.
	close	Closes the path.
	done	Ends the path definition.
	empty	Gets a value indicating whether this **Path** is empty.
	getBounds	Gets the bounding rect of the path.
	init	Begins a path or resets the current path.
	lineTo	Draws a line from the current point to the specified location.
	moveTo	Moves the path to the specified location.
	quadraticCurveTo	Draws a quadratic bezier curve.

See Also

Path Class
MindFusion.Drawing Namespace

Path Constructor

Initializes a new instance of the Path class.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript
`function Path ([pathString])`

Parameters

pathString
Optional.
String. A string representing the path figures.

See Also

Path Class
MindFusion.Drawing Namespace

Path Methods

Path.addEllipse Method

Adds an ellipse figure to the path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
function addEllipse (x, y, width, height)
```

Parameters

x
Number. The x-coordinate of the center of the ellipse.

y
Number. The y-coordinate of the center of the ellipse.

width
Number. The width of the ellipse.

height
Number. The height of the ellipse.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.addRect Method

Adds a rectangle figure to the path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
function addRect (x, y, width, height)
```

Parameters

x
Number. The x-coordinate of the rectangle.

y
Number. The y-coordinate of the rectangle.

width
Number. The width of the rectangle.

height

Number. The height of the rectangle.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.arcTo Method

Draws an arc.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function arcTo (x, y, radius, startAngle, endAngle, anticlockwise)
```

Parameters

x
Number. The x-coordinate of the center of the circle.

y
Number. The y-coordinate of the center of the circle.

radius
Number. The radius of the circle

startAngle
Number. The starting angle in radians.

endAngle
Number. The ending angle in radians.

anticlockwise
Boolean. Specifies whether the drawing should be counterclockwise or clockwise.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.bezierTo Method

Draws a cubic bezier curve.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function bezierTo (x1, y1, x2, y2, x3, y3)
```

Parameters

x1
Number. The x-coordinate of the first bezier control point.

y1
Number. The y-coordinate of the first bezier control point.

x2
Number. The x-coordinate of the second bezier control point.

y2
Number. The y-coordinate of the second bezier control point.

x3
Number. The x-coordinate of the ending point.

y3
Number. The y-coordinate of the ending point.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.clone Method

Creates a Path object identical to the current object.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function clone ()
```

Return Value

The newly created Path object.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.close Method

Closes the path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function close ()
```

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.done Method

Ends the path definition.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function done ()
```

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.empty Method

Gets a value indicating whether this Path is empty.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function empty ()
```

Return Value

true if the path does not contain any figures; otherwise, **false**.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.getBounds Method

Gets the bounding rect of the path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
function getBounds ()
```

Return Value

Rect. The bounding rect.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.init Method

Begins a path or resets the current path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
function init ()
```

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.lineTo Method

Draws a line from the current point to the specified location.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
function lineTo (x, y)
```

Parameters

x
Number. The x-coordinate of the point.

y

Number. The y-coordinate of the point.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.moveTo Method

Moves the path to the specified location.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function moveTo (x, y)
```

Parameters

x
Number. The x-coordinate of the point.

y
Number. The y-coordinate of the point.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.quadraticCurveTo Method

Draws a quadratic bezier curve.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
function quadraticCurveTo (x1, y1, x, y)
```

Parameters

x1
Number. The x-coordinate of the bezier control point.

y1
Number. The y-coordinate of the bezier control point.

x
Number. The x-coordinate of the ending point.

y
Number. The y-coordinate of the ending point.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path Properties

Path.brush Property

Gets or sets the brush used to fill a closed path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
get brush() {}
set brush(value) {}
```

Property Value

Object

A color string or an object literal specifying gradient.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.pen Property

Gets or sets the pen used to draw the path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

```
JavaScript
get pen() {}
set pen(value) {}
```

Property Value

Object

A color string.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Path.text Property

Gets or sets the text displayed inside this path.

Namespace: MindFusion.Drawing
File: Path.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

A string containing path's text.

See Also

Path Members
Path Class
MindFusion.Drawing Namespace

Point Class

Represents a point.

Syntax

JavaScript

```
// class
Point.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Point

See Also

Point Members
MindFusion.Drawing Namespace

Point Members

Represents a point.

The following tables list the members exposed by the Point type.

Public Constructors

	Name	Description
	Point	Initializes a new instance of the **Point** class with the specified X and Y coordinates.

Public Properties

	Name	Description
	x	Gets or sets the x-coordinate of the point.
	y	Gets or sets the y-coordinate of the point.

Public Methods

	Name	Description
	empty	Gets a value indicating whether this **Point** is empty.
	equals	Specifies whether this **Point** contains the same coordinates as the specified point.

See Also

Point Class
MindFusion.Drawing Namespace

Point Constructor

Initializes a new instance of the Point class with the specified X and Y coordinates.

Namespace: MindFusion.Drawing
File: Point.js

Syntax

JavaScript
`function Point (x, y)`

Parameters

x
The X-coordinate of the Point.

y
The Y-coordinate of the **Point**.

See Also

Point Class
MindFusion.Drawing Namespace

Point Methods

Point.empty Method

Gets a value indicating whether this Point is empty.

Namespace: MindFusion.Drawing
File: Point.js

Syntax

```
JavaScript
function empty ()
```

Return Value

true if both x and y are 0; otherwise, **false**.

See Also

Point Members
Point Class
MindFusion.Drawing Namespace

Point.equals Method

Specifies whether this Point contains the same coordinates as the specified point.

Namespace: MindFusion.Drawing
File: Point.js

Syntax

```
JavaScript
function equals ([point])
```

Parameters

point
Optional. The Point to test.

Return Value

true if *point* is a Point and has the same coordinates as this **Point**.

See Also

Point Members
Point Class
MindFusion.Drawing Namespace

Point Properties

Point.x Property

Gets or sets the x-coordinate of the point.

Namespace: MindFusion.Drawing
File: Point.js

Syntax

```
JavaScript
get x() {}
set x(value) {}
```

Property Value

Number

The x-coordinate of the point.

See Also

Point Members
Point Class
MindFusion.Drawing Namespace

Point.y Property

Gets or sets the y-coordinate of the point.

Namespace: MindFusion.Drawing
File: Point.js

Syntax

```
JavaScript
get y() {}
set y(value) {}
```

Property Value

Number

The y-coordinate of the point.

See Also

Point Members
Point Class
MindFusion.Drawing Namespace

Rect Class

Represents a rectangle.

Syntax

JavaScript
`// class` `Rect.prototype = {}`

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
MindFusion.Drawing.Rect

See Also

Rect Members
MindFusion.Drawing Namespace

Rect Members

Represents a rectangle.

The following tables list the members exposed by the Rect type.

Public Constructors

	Name	Description
	Rect	Overloaded.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	height	Gets or sets the height of the rectangle.
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)
	width	Gets or sets the width of the rectangle.
	x	Gets or sets the x-coordinate of the upper-left corner of the rectangle.
	y	Gets or sets the y-coordinate of the upper-left corner of the rectangle.

Public Methods

	Name	Description
	bottom	Gets the y-coordinate that is the sum of the y and height values of this **Rect** object.
	bottomLeft	Gets the bottom-left edge of this **Rect** object.
	bottomRight	Gets the bottom-right edge of this **Rect** object.
	center	Gets the center of this **Rect** object.
	clone	Creates a **Rect** object identical to the current object.
	contains	Determines if the rectangular region represented by *rect* is entirely contained within this **Rect** object.
	containsPoint	Determines if the specified point is contained within this **Rect** object.
	equals	Specifies whether this Rect contains the same coordinates as the specified rectangle.
S	fromLTRB	Creates a Rect object with the specified edge locations.
	intersect	Returns a **Rect** object representing the intersection of the current rectangle with the specified rectangle.
	intersectsWith	Checks if this **Rect** intersects with the specified rect.
	isEmpty	Gets a value indicating whether this Rect is empty.
	left	Gets the x-coordinate of the left edge of this **Rect** object.
	right	Gets the x-coordinate that is the sum of x and width values of this **Rect**.
	setCenter	Sets the center of this **Rect** object.
	setLocation	Sets the top-left edge of this **Rect** object.
	top	Gets the y-coordinate of the top edge of this **Rect** object.
	topLeft	Gets the top-left edge of this **Rect**.
	topMiddle	Gets the top-middle of this **Rect** object.
	topRight	Gets the top-right edge of this **Rect** object.
	union	Returns a **Rect** object representing the union of the current rectangle with the specified rectangle.

See Also

Rect Class
MindFusion.Drawing Namespace

Rect Constructor

Overload List

Name	Description
Rect	Initializes a new instance of the Rect class.
Rect	Initializes a new instance of the Rect class.

See Also

Rect Class
MindFusion.Drawing Namespace

Rect Constructor

Initializes a new instance of the Rect class.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript
`function Rect (x, y, [width, [height]])`

Parameters

x
Number | Point. The X-coordinate of the top left corner of the rectangle.

y
Number | Size. The Y-coordinate of the top left corner of the rectangle.

width
Optional.
Number. The width of the rectangle.

height
Optional.
Number. The height of the rectangle.

See Also

Rect Constructor Overload List
Rect Class
MindFusion.Drawing Namespace

Rect Constructor

Initializes a new instance of the Rect class.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript
`function Rect (x, y, width, height)`

Parameters

x
The X-coordinate of the top left corner of the rectangle.

y
The Y-coordinate of the top left corner of the rectangle.

width
The width of the rectangle.

height
The height of the rectangle.

See Also

Rect Constructor Overload List
Rect Class
MindFusion.Drawing Namespace

Rect Methods

Rect.bottom Method

Gets the y-coordinate that is the sum of the y and height values of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function bottom ()
```

Return Value

The y-coordinate that is the sum of y and height of this Rect.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.bottomLeft Method

Gets the bottom-left edge of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function bottomLeft ()
```

Return Value

The bottom-left edge, which is a Point with x that is equal to x and with y that is the sum of y and height.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.bottomRight Method

Gets the bottom-right edge of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function bottomRight ()
```

Return Value

The bottom-right edge, which is a Point with x that is the sum of x and width and with y that is the sum of y and height.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.center Method

Gets the center of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function center ()
```

Return Value

The center, which is a Point with x that is the sum of x and half of width and with y that is the sum of y and half of height.

See Also

Rect Members
Rect Class

Rect.clone Method

Creates a Rect object identical to the current object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function clone ()
```

Return Value

Rect. The newly created Rect object.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.contains Method

Determines if the rectangular region represented by *rect* is entirely contained within this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function contains (rect)
```

Parameters

rect
The Rect to test.

Return Value

true if the rectangular region represented by *rect* is entirely contained within this **Rect**; otherwise, **false**.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.containsPoint Method

Determines if the specified point is contained within this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function containsPoint (point)
```

Parameters

point
Point. The point to check.

Return Value

true if point is contained within the Rect; otherwise, **false**.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.equals Method

Specifies whether this Rect contains the same coordinates as the specified rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function equals (rect)
```

Parameters

rect
Rect. The Rect to test.

Return Value

Boolean. true if rect has the same coordinates as this Rect.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.fromLTRB Method

Creates a Rect object with the specified edge locations.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function fromLTRB (l, t, r, b)
```

Parameters

l
Number. The x-coordinate of the upper-left corner of this Rect.

t
Number. The y-coordinate of the upper-left corner of this Rect.

r
Number. The x-coordinate of the lower-right corner of this Rect.

b
Number. The y-coordinate of the lower-right corner of this Rect.

Return Value

Rect. A rectangle with the specified coordinates.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.intersect Method

Returns a Rect object representing the intersection of the current rectangle with the specified rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function intersect (rect)
```

Parameters

rect
Rect. The rectangle to intersect.

Return Value

A Rect object representing the intersection of the two rectangles.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.intersectsWith Method

Checks if this Rect intersects with the specified rect.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function intersectsWith (rect)
```

Parameters

rect
The other Rect.

Return Value

true if this **Rect** and the specified **Rect** intersect; otherwise, **false**.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.isEmpty Method

Gets a value indicating whether this Rect is empty.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function isEmpty ()
```

Return Value

true if both width and height are 0; otherwise, **false**.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.left Method

Gets the x-coordinate of the left edge of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function left ()
```

Return Value

The x-coordinate of the left edge of this Rect object.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.right Method

Gets the x-coordinate that is the sum of x and width values of this Rect.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function right ()
```

Return Value

The x-coordinate that is the sum of x and width of this rectangle.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.setCenter Method

Sets the center of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
```

```
function setCenter (point)
```

Parameters

point
The center of this Rect.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.setLocation Method

Sets the top-left edge of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function setLocation (point)
```

Parameters

point
The top-left edge of the Rect.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.top Method

Gets the y-coordinate of the top edge of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
function top ()
```

Return Value

Number. The y-coordinate of the top edge of this Rect object.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.topLeft Method

Gets the top-left edge of this Rect.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function topLeft ()
```

Return Value

The top-left edge of this Rect.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.topMiddle Method

Gets the top-middle of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function topMiddle ()
```

Return Value

The top-middle coordinate, which is a Point with x that is the sum of x and half of width and with y equal to y.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.topRight Method

Gets the top-right edge of this Rect object.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function topRight ()
```

Return Value

The top-right edge, which is a Point with x that is the sum of x and width and with y equal to y.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.union Method

Returns a Rect object representing the union of the current rectangle with the specified rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
function union (rect)
```

Parameters

rect
Rect. The rectangle to union.

Return Value

A Rect object that bounds the union of the two rectangles.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect Properties

Rect.height Property

Gets or sets the height of the rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
get height() {}
set height(value) {}
```

Property Value

Number

The height of the rectangle.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.width Property

Gets or sets the width of the rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
get width() {}
set width(value) {}
```

Property Value

Number

The width of the rectangle.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.x Property

Gets or sets the x-coordinate of the upper-left corner of the rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

```
JavaScript
get x() {}
set x(value) {}
```

Property Value

Number

The x-coordinate of the upper-left corner of the rectangle.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Rect.y Property

Gets or sets the y-coordinate of the upper-left corner of the rectangle.

Namespace: MindFusion.Drawing
File: Rect.js

Syntax

JavaScript

```
get y() {}
set y(value) {}
```

Property Value

Number

The x-coordinate of the upper-left corner of the rectangle.

See Also

Rect Members
Rect Class
MindFusion.Drawing Namespace

Size Class

Represents the size of 2D object.

Syntax

JavaScript

```
// class
Size.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Size

See Also

Size Members

Represents the size of 2D object.

The following tables list the members exposed by the Size type.

Public Constructors

	Name	Description
	Size	Initializes a new instance of the **Size** class with the specified width and height.

Public Properties

	Name	Description
	height	Gets or sets the size height.
	width	Gets or sets the size width.

Public Methods

	Name	Description
	empty	Gets a value indicating whether this Size is empty.

See Also

Size Class
MindFusion.Drawing Namespace

Size Constructor

Initializes a new instance of the Size class with the specified width and height.

Namespace: MindFusion.Drawing
File: Size.js

Syntax

JavaScript
`function Size (width, height)`

Parameters

width
Number. Specifies width.

height
Number. Specifies height.

See Also

Size Class
MindFusion.Drawing Namespace

Size Methods

Size.empty Method

Gets a value indicating whether this Size is empty.

Namespace: MindFusion.Drawing
File: Size.js

Syntax

JavaScript

```
function empty ()
```

Return Value

Boolean. true if both width and height are 0; otherwise, false.

See Also

Size Members
Size Class
MindFusion.Drawing Namespace

Size Properties

Size.height Property

Gets or sets the size height.

Namespace: MindFusion.Drawing
File: Size.js

Syntax

JavaScript

```
get height() {}
set height(value) {}
```

Property Value

Number

The size height.

See Also

Size Members
Size Class

Size.width Property

Gets or sets the size width.

Namespace: MindFusion.Drawing
File: Size.js

Syntax

JavaScript

```
get width() {}
set width(value) {}
```

Property Value

Number

The size width.

See Also

Size Members
Size Class
MindFusion.Drawing Namespace

StringAlignment Enumeration

Specifies the alignment of a text string relative to its layout rectangle.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

JavaScript

```
// enum
StringAlignment = {}
```

Members

	Member name	Description
	Center	The text is drawn in the center of the layout rectangle.
	Far	The text is drawn in the far corner of the layout rectangle.le.
	Near	The text is drawn in the near corner of the layout rectangle.

See Also

StringFormat Class

Encapsulates text layout information.

Syntax

```
JavaScript
// class
StringFormat.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.StringFormat

See Also

StringFormat Members
MindFusion.Drawing Namespace

StringFormat Members

Encapsulates text layout information.

The following tables list the members exposed by the StringFormat type.

Public Constructors

	Name	Description
	StringFormat	Initializes a new instance of the StringFormat class.

Public Properties

	Name	Description
	alignment	Gets or sets the horizontal alignment of the text.
	lineAlignment	Gets or sets the vertical alignment of the text.

See Also

StringFormat Class
MindFusion.Drawing Namespace

StringFormat Constructor

Initializes a new instance of the StringFormat class.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
JavaScript
function StringFormat ()
```

See Also

StringFormat Class
MindFusion.Drawing Namespace

StringFormat Properties

StringFormat.alignment Property

Gets or sets the horizontal alignment of the text.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
JavaScript
get alignment() {}
set alignment(value) {}
```

Property Value

StringAlignment

See Also

StringFormat Members
StringFormat Class
MindFusion.Drawing Namespace

StringFormat.lineAlignment Property

Gets or sets the vertical alignment of the text.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

```
JavaScript
get lineAlignment() {}
set lineAlignment(value) {}
```

Property Value

StringAlignment

See Also

Text Class

Represents a text container.

Syntax

```
JavaScript
// class
Text.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
MindFusion.Drawing.Text

See Also

Text Members
MindFusion.Drawing Namespace

Text Members

Represents a text container.

The following tables list the members exposed by the Text type.

Public Constructors

	Name	Description
	Text	Initializes a new instance of the **Text** class.

Public Fields

	Name	Description
	bounds	The object's bounds.

Public Properties

	Name	Description
	font	Gets or sets the font used to render text.
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)

	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	lineAlignment	Gets or sets the vertical alignment of this text.
	rotationAngle	Gets a value indicating the rotation of the text container.
	text	Gets or sets the text to render.
	textAlignment	Gets or sets the horizontal alignment of this text.
	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	clone	Creates a **Text** object identical to the current object.
	getBounds	Gets a value indicating the bounding rectangle of the text container.
	setBounds	Sets a value indicating the bounding rectangle of the text container.

See Also

Text Class
MindFusion.Drawing Namespace

Text Fields

Text.bounds Field

The object's bounds.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

JavaScript
`var bounds`

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text Constructor

Initializes a new instance of the Text class.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

```
JavaScript
function Text (text, bounds)
```

Parameters

text
String. The text to be displayed in the container.

bounds
Rect. The bounds of the container.

See Also

Text Class
MindFusion.Drawing Namespace

Text Methods

Text.clone Method

Creates a Text object identical to the current object.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

```
JavaScript
function clone ()
```

Return Value

The newly created Text object.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text.getBounds Method

Gets a value indicating the bounding rectangle of the text container.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

```
JavaScript
function getBounds ()
```

Return Value

Rect. The bounding rectangle.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text.setBounds Method

Sets a value indicating the bounding rectangle of the text container.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

```
JavaScript
function setBounds (bounds, [angle])
```

Parameters

bounds
Rect. The bounding rectangle.

angle
Optional.
Number. The rotation angle to be applied to the bounding rectangle.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text Properties

Text.font Property

Gets or sets the font used to render text.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

JavaScript

```
get font() {}
set font(value) {}
```

Property Value

Font

The Font of this text.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text.lineAlignment Property

Gets or sets the vertical alignment of this text.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

JavaScript

```
get lineAlignment() {}
set lineAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text.rotationAngle Property

Gets a value indicating the rotation of the text container.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

JavaScript

```
get rotationAngle() {}
set rotationAngle(value) {}
```

Property Value

Number

The angle.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text.text Property

Gets or sets the text to render.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

JavaScript

```
get text() {}
set text(value) {}
```

Property Value

String

The text.

See Also

Text Members
Text Class
MindFusion.Drawing Namespace

Text.textAlignment Property

Gets or sets the horizontal alignment of this text.

Namespace: MindFusion.Drawing
File: Text.js

Syntax

JavaScript

```
get textAlignment() {}
set textAlignment(value) {}
```

Property Value

Alignment

A member of the Alignment enumeration.

See Also

Thickness Class

Describes the thickness of a rectangular frame.

Syntax

```
// class
Thickness.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Thickness

See Also

Thickness Members
MindFusion.Drawing Namespace

Thickness Members

Describes the thickness of a rectangular frame.

The following tables list the members exposed by the Thickness type.

Public Constructors

	Name	Description
	Thickness	Overloaded.

Public Properties

	Name	Description
	bottom	Gets or sets the width of the bottom side of the frame.
	isRelative	Gets or sets a flag indicating whether the thickness properties are expressed as relative or absolute quantities.
	left	Gets or sets the width of the left side of the frame.
	right	Gets or sets the width of the right side of the frame.
	top	Gets or sets the width of the top side of the frame.

Public Methods

	Name	Description

	addToRect	Applies the current thickness to the specified rectangle by inflating the rectangle.
	applyTo	Applies the current thickness to the specified rectangle by deflating the rectangle.
	toAbsolute	Converts the current thickness to absolute value relative to the specified size.

See Also

Thickness Class
MindFusion.Drawing Namespace

Thickness Constructor

Overload List

Name	Description
Thickness	Initializes a new instance of the Thickness class with the specified widths.
Thickness	Initializes a new instance of the Thickness class with the specified widths.

See Also

Thickness Class
MindFusion.Drawing Namespace

Thickness Constructor

Initializes a new instance of the Thickness class with the specified widths.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

JavaScript

```
function Thickness (left, top, right, bottom)
```

Parameters

left
Specifies the width of the left side of the frame.

top
Specifies the width of the top side of the frame.

right
Specifies the width of the right side of the frame.

bottom

Specifies the width of the bottom side of the frame.

See Also

Thickness Constructor Overload List
Thickness Class
MindFusion.Drawing Namespace

Thickness Constructor

Initializes a new instance of the Thickness class with the specified widths.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

JavaScript

```
function Thickness (left, [top, [right, [bottom, [isRelative]]]])
```

Parameters

left
Number. Specifies the width of the left side of the frame.

top
Optional.
Number. Specifies the width of the top side of the frame.

right
Optional.
Number. Specifies the width of the right side of the frame.

bottom
Optional.
Number. Specifies the width of the bottom side of the frame.

isRelative
Optional.
Boolean. A flag indicating whether the thickness properties are expressed as relative or absolute quantities.

See Also

Thickness Constructor Overload List
Thickness Class
MindFusion.Drawing Namespace

Thickness Methods

Thickness.addToRect Method

Applies the current thickness to the specified rectangle by inflating the rectangle.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
function addToRect (rect)
```

Parameters

rect
Rect. The rectangle to inflate.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness.applyTo Method

Applies the current thickness to the specified rectangle by deflating the rectangle.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
function applyTo (rect)
```

Parameters

rect
Rect. The rectangle to deflate.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness.toAbsolute Method

Converts the current thickness to absolute value relative to the specified size.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
function toAbsolute ()
```

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness Properties

Thickness.bottom Property

Gets or sets the width of the bottom side of the frame.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
get bottom() {}
set bottom(value) {}
```

Property Value

Number

The width of the bottom side of the frame.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness.isRelative Property

Gets or sets a flag indicating whether the thickness properties are expressed as relative or absolute quantities.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
get isRelative() {}
set isRelative(value) {}
```

Property Value

Boolean

true if the thickness is relative, or **false** otherwise.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness.left Property

Gets or sets the width of the left side of the frame.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
get left() {}
set left(value) {}
```

Property Value

Number

The width of the left side of the frame.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness.right Property

Gets or sets the width of the right side of the frame.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
get right() {}
set right(value) {}
```

Property Value

Number

The width of the right side of the frame.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Thickness.top Property

Gets or sets the width of the top side of the frame.

Namespace: MindFusion.Drawing
File: Thickness.js

Syntax

```
JavaScript
get top() {}
set top(value) {}
```

Property Value

Number

The width of the top side of the frame.

See Also

Thickness Members
Thickness Class
MindFusion.Drawing Namespace

Vector Class

Represents displacement in 2D space.

Syntax

```
JavaScript
// class
Vector.prototype = {}
```

Inheritance Hierarchy

MindFusion.Drawing.Vector

See Also

Vector Members
MindFusion.Drawing Namespace

Vector Members

Represents displacement in 2D space.

The following tables list the members exposed by the Vector type.

Public Constructors

	Name	Description

	Vector	Initializes a new instance of the Vector class with the specified X and Y coordinates.

Public Fields

	Name	Description
	x	The x-coordinate of the vector.
	y	The y-coordinate of the vector.

See Also

Vector Class
MindFusion.Drawing Namespace

Vector Fields

Vector.x Field

The x-coordinate of the vector.

Namespace: MindFusion.Drawing
File: Vector.js

Syntax

JavaScript
`var x`

See Also

Vector Members
Vector Class
MindFusion.Drawing Namespace

Vector.y Field

The y-coordinate of the vector.

Namespace: MindFusion.Drawing
File: Vector.js

Syntax

JavaScript
`var y`

See Also

Vector Members
Vector Class

MindFusion.Drawing Namespace

Vector Constructor

Initializes a new instance of the Vector class with the specified X and Y coordinates.

Namespace: MindFusion.Drawing
File: Vector.js

Syntax

JavaScript

```
function Vector (x, y)
```

Parameters

x
Number. The X-coordinate of the Vector.

y
Number. The Y-coordinate of the Vector.

See Also

Vector Class
MindFusion.Drawing Namespace

Video Class

A component that displays video stream.

Syntax

JavaScript

```
// class
Video.prototype = {}
```

Remarks

The video does not play automatically. To start playing it, call the component's play method in response to user input events.

If autoProperty is enabled for a **Video** component in CompositeNode template, the template loader creates getter and setter wrapping the component's MediaLocation property.

Example

The following template fragment adds a Video component to the node:

JavaScript

```
{
    component: "Video",
    name: "Video",
    autoProperty: true,
    mediaLocation: "video.webm",
```

```
    margin: "1"
},
```

It automatically adds getVideo and setVideo methods to the node class, which can be used to get or set the video stream URL:

JavaScript

```
var VideoNode = CompositeNode.classFromTemplate(
    "VideoNode", templateJson);
var node = new VideoNode();
node.setVideo("video2.webm");
diagram.addItem(node);
```

Inheritance Hierarchy

MindFusion.Drawing.ComponentBase
MindFusion.Drawing.Video

See Also

Video Members
MindFusion.Drawing Namespace

Video Members

A component that displays video stream.

The following tables list the members exposed by the Video type.

Public Constructors

	Name	Description
	Video	Initializes a new instance of the Video class.

Public Fields

	Name	Description
	bounds	The object's bounds.

Public Properties

	Name	Description
	gridColumn	Gets or sets the index of the grid column this component is contained in. (Inherited from ComponentBase.)
	gridRow	Gets or sets the index of the grid row this component is contained in. (Inherited from ComponentBase.)
	horizontalAlignment	Gets or sets the component's horizontal alignment. (Inherited from ComponentBase.)
	mediaLocation	Gets or sets URL of the video stream displayed in this component.

	verticalAlignment	Gets or sets the component's vertical alignment. (Inherited from ComponentBase.)
	visibility	Gets or sets the visibility of this component. (Inherited from ComponentBase.)

Public Methods

	Name	Description
	clone	Creates a Video object identical to the current object.
	isPlaying	Detects if video is playing.
	pause	Pauses the video.
	play	Starts playing the video stream.

See Also

Video Class
MindFusion.Drawing Namespace

Video Fields

Video.bounds Field

The object's bounds.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript
`var bounds`

See Also

Video Members
Video Class
MindFusion.Drawing Namespace

Video Constructor

Initializes a new instance of the Video class.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript

```
function Video (bounds)
```

Parameters

bounds
Rect. The bounds of the Video.

See Also

Video Class
MindFusion.Drawing Namespace

Video Methods

Video.clone Method

Creates a Video object identical to the current object.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript

```
function clone ()
```

Return Value

See Also

Video Members
Video Class
MindFusion.Drawing Namespace

Video.isPlaying Method

Detects if video is playing.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript

```
function isPlaying ()
```

See Also

Video Members
Video Class
MindFusion.Drawing Namespace

Video.pause Method

Pauses the video.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript

```
function pause ()
```

See Also

Video Members
Video Class
MindFusion.Drawing Namespace

Video.play Method

Starts playing the video stream.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript

```
function play ()
```

See Also

Video Members
Video Class
MindFusion.Drawing Namespace

Video Properties

Video.mediaLocation Property

Gets or sets URL of the video stream displayed in this component.

Namespace: MindFusion.Drawing
File: Video.js

Syntax

JavaScript

```
get mediaLocation() {}
set mediaLocation(value) {}
```

Property Value

String

A URL string that identifies the video to play.

See Also

Video Members
Video Class
MindFusion.Drawing Namespace

Visibility Enumeration

Specifies visibility for components.

Namespace: MindFusion.Drawing
File: Enum.js

Syntax

JavaScript

```
// enum
Visibility = {}
```

Members

	Member name	Description
	Collapsed	Specifies that the component is hidden, and the space it occupies in the layout is not preserved.
	Hidden	Specifies that the component is hidden, but the space it occupies in the layout is preserved.
	Visible	Specifies that the component is visible.

See Also

MindFusion.Drawing Namespace

MindFusion.Graphs Namespace

Anchoring Enumeration

Defines values that specify how automatic layout algorithms align links to anchor points.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

JavaScript

```
// enum
Anchoring = {}
```

Members

	Member name	Description
	Custom	Custom alignment logic.
	Ignore	Anchor points are ignored.
	Keep	Links are aligned to their original anchor points.
	Reassign	Links are aligned to new anchor points, depending on the positions of graph nodes after a layout is applied.

Remarks

Members of this enumeration can be assigned to the anchoring property of layout classes.

See Also

MindFusion.Graphs Namespace

AssistantPosition Enumeration

Specifies the final position of an assistant node. Assistants with type left or right are always positioned to the left or right respectively. The position of the "normal" assistants depends on their index in the children list.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

JavaScript

```
// enum
AssistantPosition = {}
```

Members

	Member name	Description
	Left	The node is distributed to the left of its parent.
	Right	The node is distributed to the right of its parent.

See Also

MindFusion.Graphs Namespace

AssistantType Enumeration

Indicates the type of an assistant node in a tree.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

```
JavaScript
// enum
AssistantType = {}
```

Members

	Member name	Description
	Left	The node is an assistant and it is always positioned to the left of its parent.
	Normal	The node is an assistant and its relative position is determinedautomatically by the layout.
	Right	The node is an assistant and it is always positioned to the right of its parent.

Remarks

Members of this enumeration can be assigned to the **treeLayoutAssistant** key of node's layoutTraits.

See Also

MindFusion.Graphs Namespace

BorderedTreeLayout Class

Implements algorithms for arranging tree structures.

Syntax

```
JavaScript
// class
BorderedTreeLayout.prototype = {}
```

Remarks

To apply this layout to a diagram, create a **BorderedTreeLayout** instance, set its members and invoke the arrange method of the Diagram class. The method lays out all subtrees in a diagram.

As long as the graph is a tree, the algorithm can find the root node automatically. The distance between the root and the second level and between all subsequent levels can be set through the levelDistance property. The distance between adjacent nodes in the same level is set through nodeDistance.

In addition to arranging the tree nodes, **BorderedTreeLayout** also sets the shape of links as set through its linkType property.

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.BorderedTreeLayout

See Also

BorderedTreeLayout Members
MindFusion.Graphs Namespace

BorderedTreeLayout Members

Implements algorithms for arranging tree structures.

The following tables list the members exposed by the BorderedTreeLayout type.

Public Constructors

	Name	Description
	BorderedTreeLayout	Initializes a new instance of the BorderedTreeLayout class.

Public Properties

	Name	Description
	anchoring	Gets or sets a value indicating how to align links to the anchor points of nodes.
	direction	Gets or sets the orientation of the arranged graph.
	keepGroupLayout	Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.
	keepRootPosition	Gets or sets a value indicating whether to keep the root node at its original position.
	levelDistance	Gets or sets the space to leave between adjacent levels of the tree.
	linkType	Gets or sets the type of links in the arranged tree.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.
	nodeDistance	Gets or sets the space to leave between adjacent nodes on the same level.
	reversedLinks	Gets or sets a value indicating the link direction that relates parent nodes to child nodes.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout Constructor

Initializes a new instance of the BorderedTreeLayout class.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
function BorderedTreeLayout ()
```

See Also

BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout Methods

BorderedTreeLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
function arrange (graph)
```

Parameters

graph
The Graph to arrange.

Remarks

This method is called internally by Diagram.arrange and ContainerNode.arrange.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout Properties

BorderedTreeLayout.anchoring Property

Gets or sets a value indicating how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the Anchoring enumeration. The default value is **Ignore**.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.direction Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
get direction() {}
set direction(value) {}
```

Property Value

LayoutDirection

A member of the LayoutDirection enumeration. The default value is **TopToBottom**.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.keepGroupLayout Property

Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

```
JavaScript
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group of attached nodes should be treated as a single vertex in the arranged graph, or **false** otherwise.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.keepRootPosition Property

Gets or sets a value indicating whether to keep the root node at its original position.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

```
JavaScript
get keepRootPosition() {}
set keepRootPosition(value) {}
```

Property Value

Boolean

true if the root node should be kept at its original position, or **false** otherwise.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.levelDistance Property

Gets or sets the space to leave between adjacent levels of the tree.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

```
JavaScript
```

```
get levelDistance() {}
set levelDistance(value) {}
```

Property Value

Number

The space to leave between adjacent levels of the tree.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.linkType Property

Gets or sets the type of links in the arranged tree.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
get linkType() {}
set linkType(value) {}
```

Property Value

TreeLayoutLinkType

A member of the TreeLayoutLinkType enumeration. The default value is **Default**,

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.multipleGraphsPlacement Property

Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration. The default value is **Horizontal**.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.nodeDistance Property

Gets or sets the space to leave between adjacent nodes on the same level.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
get nodeDistance() {}
set nodeDistance(value) {}
```

Property Value

Number

The space to leave between adjacent nodes on the same level.

See Also

BorderedTreeLayout Members
BorderedTreeLayout Class
MindFusion.Graphs Namespace

BorderedTreeLayout.reversedLinks Property

Gets or sets a value indicating the link direction that relates parent nodes to child nodes.

Namespace: MindFusion.Graphs
File: BorderedTreeLayout.js

Syntax

JavaScript

```
get reversedLinks() {}
set reversedLinks(value) {}
```

Property Value

Boolean

true to indicate reversed direction (links point to parent nodes), or **false** otherwise.

See Also

Edge Class

Represents an edge connecting two vertices in a graph.

Syntax

```
// class
Edge.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Edge

See Also

Edge Members
MindFusion.Graphs Namespace

Edge Members

Represents an edge connecting two vertices in a graph.

The following tables list the members exposed by the Edge type.

Public Constructors

	Name	Description
	Edge	Initializes a new instance of the **Edge** class.

Public Fields

	Name	Description
	destination	The destination vertex.
	origin	The origin vertex.

Public Methods

	Name	Description
	adjacentTo	Checks if the current edge is adjacent to the specified edge, that is, they have common vertex.
	changeDestination	Changes the destination of the current edge to the specified vertex.

	changeOrigin	Changes the origin of the current edge to the specified vertex.
	createReverseEdge	Creates a new edge similar to the current edge but with reversed direction.
	getCommonVertex	Returns a common vertex for the current edge and the specified edge.
	getEnds	Returns an array containing the origin and destination vertices of this edge.
	getOtherEnd	Returns the vertex connected by this edge that is not the specified vertex.
	incidentWith	Checks if the current edge is incident with the specified vertex.
	joins	Checks if the current edge connects the specified vertices.
	reverse	Reverses this edge.

See Also

Edge Class
MindFusion.Graphs Namespace

Edge Fields

Edge.destination Field

The destination vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
var destination
```

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.origin Field

The origin vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
var origin
```

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge Constructor

Initializes a new instance of the Edge class.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
function Edge (origin, destination)
```

Parameters

origin
Vertex. The origin vertex.

destination
Vertex. The destination vertex.

See Also

Edge Class
MindFusion.Graphs Namespace

Edge Methods

Edge.adjacentTo Method

Checks if the current edge is adjacent to the specified edge, that is, they have common vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
function adjacentTo (edge)
```

Parameters

edge
Edge. The edge to check.

Return Value

true if the current edge is adjacent to the specified edge; otherwise, **false**.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.changeDestination Method

Changes the destination of the current edge to the specified vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
function changeDestination (vertex)
```

Parameters

vertex
Vertex. The new destination of the edge.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.changeOrigin Method

Changes the origin of the current edge to the specified vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
function changeOrigin (vertex)
```

Parameters

vertex
Vertex. The new origin of the edge.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.createReverseEdge Method

Creates a new edge similar to the current edge but with reversed direction.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

```
JavaScript
function createReverseEdge ()
```

Return Value

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.getCommonVertex Method

Returns a common vertex for the current edge and the specified edge.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

```
JavaScript
function getCommonVertex (edge)
```

Parameters

edge
Edge. The edge for which to find common vertex.

Return Value

The common vertex if it exists, otherwise **null**.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.getEnds Method

Returns an array containing the origin and destination vertices of this edge.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

```
JavaScript
function getEnds ()
```

Return Value

Array. An array containing the related vertices.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.getOtherEnd Method

Returns the vertex connected by this edge that is not the specified vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

```
JavaScript
function getOtherEnd (vertex)
```

Parameters

vertex
Vertex. One of the vertices connected by this edge.

Return Value

Vertex. The other vertex connected by this edge.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.incidentWith Method

Checks if the current edge is incident with the specified vertex.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
function incidentWith (vertex)
```

Parameters

vertex
Vertex. The vertex to check.

Return Value

true if the edge is incident with the vertex; otherwise, **false**.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.joins Method

Checks if the current edge connects the specified vertices.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

JavaScript

```
function joins (v1, v2)
```

Parameters

v1
Vertex. The first vertex.

v2
Vertex. The second vertex.

Return Value

true if the current edge connects the specified vertices; otherwise, **false**.

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

Edge.reverse Method

Reverses this edge.

Namespace: MindFusion.Graphs
File: Edge.js

Syntax

```
JavaScript
function reverse ()
```

See Also

Edge Members
Edge Class
MindFusion.Graphs Namespace

FlowchartLayout Class

FlowchartLayout can be used to arrange flowcharts representing program source code.

Syntax

```
JavaScript
// class
FlowchartLayout.prototype = {}
```

Remarks

The algorithm recognizes program code -like patterns in the graph, such as loops, sequences and if/switch branchings, and arranges them recursively. **FlowchartLayout** could be used to arrange other types of graphs as well, though with some restrictions. For example it treats all back links as loops in the code, and expects that they are nested - loop links starting closer to the stop node should end closer to the start node. Another similar restriction is that there shouldn't be any cross-links that connect different branches of a decision sub-graph.

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.FlowchartLayout

See Also

FlowchartLayout Members
MindFusion.Graphs Namespace

FlowchartLayout Members

FlowchartLayout can be used to arrange flowcharts representing program source code.

The following tables list the members exposed by the **FlowchartLayout** type.

Public Constructors

	Name	Description
	FlowchartLayout	Initializes a new instance of the **FlowchartLayout** class.

Public Properties

	Name	Description
	anchoring	Gets or sets a value indicating how to align links to the anchor points of nodes.
	branchPadding	Gets or sets how much space to leave between adjacent decision branches in the flowchart.
	keepGroupLayout	Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.
	linkPadding	Gets or sets how much space to leave between adjacent back links designating nested loops.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.
	nodeDistance	Gets or sets the distance between consecutive nodes in the flowchart.
	orientation	Gets or sets the orientation of the arranged graph.

Public Methods

	Name	Description
	init	Returns a Builder object used to configure this **FlowchartLayout** instance.
S	With	Returns a Builder object used to configure and create new **FlowchartLayout** instances.

See Also

FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout Constructor

Initializes a new instance of the FlowchartLayout class.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

JavaScript
`function FlowchartLayout ()`

See Also

FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout Methods

FlowchartLayout.init Method

Returns a Builder object used to configure this FlowchartLayout instance.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

```
JavaScript
function init ()
```

Return Value

A Builder instance.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.With Method

Returns a Builder object used to configure and create new FlowchartLayout instances.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

```
JavaScript
function With ()
```

Return Value

A Builder instance.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout Properties

FlowchartLayout.anchoring Property

Gets or sets a value indicating how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

```
JavaScript
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the Anchoring enumeration.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.branchPadding Property

Gets or sets how much space to leave between adjacent decision branches in the flowchart.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

```
JavaScript
get branchPadding() {}
set branchPadding(value) {}
```

Property Value

Number

A numeric value specifying the distance between adjacent branches.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.keepGroupLayout Property

Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

```
JavaScript
```

```
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group of attached nodes should be treated as a single vertex in the arranged graph, or **false** otherwise.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.linkPadding Property

Gets or sets how much space to leave between adjacent back links designating nested loops.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

JavaScript

```
get linkPadding() {}
set linkPadding(value) {}
```

Property Value

Number

A numeric value specifying the distance between adjacent back links.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.multipleGraphsPlacement Property

Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

JavaScript

```
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.nodeDistance Property

Gets or sets the distance between consecutive nodes in the flowchart.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

JavaScript

```
get nodeDistance() {}
set nodeDistance(value) {}
```

Property Value

Number

A numeric value specifying the distance between neighbor nodes in the same sequence.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FlowchartLayout.orientation Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: FlowchartLayout.js

Syntax

JavaScript

```
get orientation() {}
set orientation(value) {}
```

Property Value

Orientation

A member of the Orientation enumeration.

See Also

FlowchartLayout Members
FlowchartLayout Class
MindFusion.Graphs Namespace

FractalLayout Class

The FractalLayout tree layout algorithm places child nodes symmetrically around their parent node. Nodes at the lowest level are arranged directly in a circle around their parent. At the upper level, the already arranged nodes form branches that are arranged in a circle around the new parent node. The algorithm is recursively repeated till the highest level is reached.

Syntax

JavaScript

```
// class
FractalLayout.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.FractalLayout

See Also

FractalLayout Members
MindFusion.Graphs Namespace

FractalLayout Members

The FractalLayout tree layout algorithm places child nodes symmetrically around their parent node. Nodes at the lowest level are arranged directly in a circle around their parent. At the upper level, the already arranged nodes form branches that are arranged in a circle around the new parent node. The algorithm is recursively repeated till the highest level is reached.

The following tables list the members exposed by the FractalLayout type.

Public Constructors

	Name	Description
	FractalLayout	Initializes a new instance of the FractalLayout class.

Public Properties

	Name	Description
	anchoring	Gets or sets a value indicating how to align links to the anchor points of nodes.
	keepGroupLayout	Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)

	multipleGraphsPlacement	Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.
	root	Gets or sets the node that should be placed at the center of the tree.

See Also

FractalLayout Class
MindFusion.Graphs Namespace

FractalLayout Fields

FractalLayout Constructor

Initializes a new instance of the FractalLayout class.

Namespace: MindFusion.Graphs
File: FractalLayout.js

Syntax

JavaScript

```
function FractalLayout ()
```

See Also

FractalLayout Class
MindFusion.Graphs Namespace

FractalLayout Properties

FractalLayout.anchoring Property

Gets or sets a value indicating how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: FractalLayout.js

Syntax

JavaScript

```
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the Anchoring enumeration. The default value is **Ignore**.

See Also

FractalLayout Members
FractalLayout Class
MindFusion.Graphs Namespace

FractalLayout.keepGroupLayout Property

Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: FractalLayout.js

Syntax

JavaScript

```
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group of attached nodes should be treated as a single vertex in the arranged graph, or **false** otherwise.

See Also

FractalLayout Members
FractalLayout Class
MindFusion.Graphs Namespace

FractalLayout.multipleGraphsPlacement Property

Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: FractalLayout.js

Syntax

JavaScript

```
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration. The default value is **Horizontal**.

See Also

FractalLayout Members
FractalLayout Class
MindFusion.Graphs Namespace

FractalLayout.root Property

Gets or sets the node that should be placed at the center of the tree.

Namespace: MindFusion.Graphs
File: FractalLayout.js

Syntax

```
JavaScript
get root() {}
set root(value) {}
```

Property Value

DiagramNode

The node that should be placed at the center of the tree.

See Also

FractalLayout Members
FractalLayout Class
MindFusion.Graphs Namespace

Graph Class

Represents a graph.

Syntax

```
JavaScript
// class
Graph.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Graph

See Also

Graph Members
MindFusion.Graphs Namespace

Graph Members

Represents a graph.

The following tables list the members exposed by the Graph type.

Public Constructors

	Name	Description
	Graph	Initializes a new instance of the **Graph** class.

Public Fields

	Name	Description
	edges	A list containing all edges in the graph.
	vertices	A list containing all vertices in the graph.

Public Methods

	Name	Description
	addEdge	Adds an existing edge to the graph.
	clone	Creates an exact copy of the current graph.
	createEdge	Creates a new edge connecting the specified vertices and adds it to the graph.
	createVertex	Creates a new vertex with the specified position and size and adds it to the graph.
	getConnectedComponents	Returns a list with the connected subgraphs in the current graph.
	getUpwardEmbedding	Should be called on a planar graph.
	makeAcyclic	Makes the graph acyclic by reversing the direction of selected edges.
	removeEdge	Removes an edge from the graph.
	tree	Overloaded.

See Also

Graph Class
MindFusion.Graphs Namespace

Graph Fields

Graph.edges Field

A list containing all edges in the graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
var edges
```

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.vertices Field

A list containing all vertices in the graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
var vertices
```

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph Constructor

Initializes a new instance of the Graph class.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function Graph ()
```

See Also

Graph Class
MindFusion.Graphs Namespace

Graph Methods

Graph.addEdge Method

Adds an existing edge to the graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function addEdge (edge)
```

Parameters

edge
Edge. The edge to add.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.clone Method

Creates an exact copy of the current graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function clone ([saveMapping])
```

Parameters

saveMapping
Optional.
Boolean. Indicates whether to store vertex and edge mapping information in the copy.

Return Value

Graph. The newly created copy.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.createEdge Method

Creates a new edge connecting the specified vertices and adds it to the graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function createEdge (origin, destination, [owner])
```

Parameters

origin
Vertex. The origin vertex.

destination
Vertex. The destination vertex.

owner
Optional.
Object. The owner of the new edge.

Return Value

Edge. The newly created edge.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.createVertex Method

Creates a new vertex with the specified position and size and adds it to the graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function createVertex (layoutRect, [owner])
```

Parameters

layoutRect
Rect. A rectangle specifying the position and size of the vertex.

owner
Optional.
Object. The owner of the new vertex.

Return Value

Vertex. The newly created vertex.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.getConnectedComponents Method

Returns a list with the connected subgraphs in the current graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function getConnectedComponents ()
```

Return Value

An array with connected Graph objects.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.getUpwardEmbedding Method

Should be called on a planar graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function getUpwardEmbedding ([embedding, [embeddingInfo]])
```

Parameters

embedding
Optional.
Map. Receives the embedding lists.

embeddingInfo
Optional.
Map. Optional.

Return Value

Vertex. The t node in st ordering.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.makeAcyclic Method

Makes the graph acyclic by reversing the direction of selected edges.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function makeAcyclic ()
```

Return Value

Array. A list with all edges that were reversed in order to make the graph acyclic.

Remarks

Must be called on a simple connected graph.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.removeEdge Method

Removes an edge from the graph.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function removeEdge (edge)
```

Parameters

edge
Edge. The edge to remove.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.tree Method

Overload List

Name	Description
Graph.tree	Returns a new graph object containing a subset of the vertices and edges of the original graph, such that the new graph is a tree.
Graph.tree	Returns a new graph object containing a subset of the vertices and edges of the original graph, such that the new graph is a tree.

See Also

Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.tree Method

Returns a new graph object containing a subset of the vertices and edges of the original graph, such that the new graph is a tree.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript
`function tree (root)`

Parameters

root
The desired root vertex of the tree.

Return Value

The newly created tree.

See Also

tree Method Overload List
Graph Members
Graph Class
MindFusion.Graphs Namespace

Graph.tree Method

Returns a new graph object containing a subset of the vertices and edges of the original graph, such that the new graph is a tree.

Namespace: MindFusion.Graphs
File: Graph.js

Syntax

JavaScript

```
function tree (root, enableAssistants, compactAssistants)
```

Parameters

root
Vertex. The desired root vertex of the tree.

enableAssistants

compactAssistants

Return Value

Graph. The newly created tree.

See Also

tree Method Overload List
Graph Members
Graph Class
MindFusion.Graphs Namespace

GridLayout Class

Implements a grid graph layout algorithm.

Syntax

JavaScript

```
// class
GridLayout.prototype = {}
```

Remarks

GridLayout arranges diagram nodes in a grid, keeping connected nodes close to each other. The algorithm strives to achieve a small number of link crossings. It is based on an iterative process whose initial steps shuffle the grid nodes randomly. That can lead to very different results each time the algorithm is run.

To apply the layout to a diagram, create a **GridLayout** instance, set its property values, and invoke the arrange method.

The distance between points in the layout grid can be changed through gridSize. Nodes to be placed at the upper-left and lower-right corners of the grid can be assigned to startNode and endNode.

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.GridLayout

See Also

GridLayout Members
MindFusion.Graphs Namespace

GridLayout Members

Implements a grid graph layout algorithm.

The following tables list the members exposed by the GridLayout type.

Public Constructors

	Name	Description
	GridLayout	Initializes a new instance of the GridLayout class.

Public Properties

	Name	Description
	anchoring	Gets or sets how to align links to the anchor points of nodes. (Inherited from Layout.)
	endNode	Gets or sets the node that is placed at the lower right corner of the grid.
	gridSize	Gets or sets the desired distance between adjacent grid points.
	iterations	Gets or sets for how many iterations to run the grid layout algorithm.
	keepGroupLayout	Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph. (Inherited from Layout.)
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other. (Inherited from Layout.)
	startNode	Gets or sets the node that is placed at the upper left corner of the grid.
	timeLimit	Gets or sets a time-out value for the path-finding part of the grid layout algorithm.
	useLongestPath	Gets or sets a value indicating whether the algorithm should arrange nodes around the longest path in the graph.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

GridLayout Class
MindFusion.Graphs Namespace

GridLayout Constructor

Initializes a new instance of the GridLayout class.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

JavaScript

```
function GridLayout ()
```

See Also

GridLayout Class
MindFusion.Graphs Namespace

GridLayout Methods

GridLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

JavaScript

```
function arrange (graph)
```

Parameters

graph
Graph. The graph to arrange.

Remarks

This method is called internally by Diagram.arrange.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace

GridLayout Properties

GridLayout.endNode Property

Gets or sets the node that is placed at the lower right corner of the grid.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

```
JavaScript
get endNode() {}
set endNode(value) {}
```

Property Value

DiagramNode

An instance of the DiagramNode class. Default value is **null**.

Remarks

If you assign a node to this property, then startNode cannot be **null** and must be set too. GridLayout usually produces better layouts if both **startNode** and **endNode** are **null** references; in such a case the algorithm automatically selects suitable nodes.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace

GridLayout.gridSize Property

Gets or sets the desired distance between adjacent grid points.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

```
JavaScript
get gridSize() {}
set gridSize(value) {}
```

Property Value

Number

A number specifying the grid size. The default is 25.

Remarks

When placing a node at a grid position, the node center point is aligned to the grid point at that position.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace

GridLayout.iterations Property

Gets or sets for how many iterations to run the grid layout algorithm.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

```
JavaScript
get iterations() {}
set iterations(value) {}
```

Property Value

Number

An integer number specifying the number of iterations. The default is 3000.

Remarks

More iterations would yield better layouts but require more time to complete.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace

GridLayout.startNode Property

Gets or sets the node that is placed at the upper left corner of the grid.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

```
JavaScript
get startNode() {}
set startNode(value) {}
```

Property Value

DiagramNode

An instance of the DiagramNode class. The default is **null**.

Remarks

If you assign a node to this property, then endNode cannot be **null** and must be set too. GridLayout usually produces better layouts if both **startNode** and **endNode** are **null** references; in such a case the algorithm automatically selects suitable nodes.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace

GridLayout.timeLimit Property

Gets or sets a time-out value for the path-finding part of the grid layout algorithm.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

```
JavaScript
get timeLimit() {}
set timeLimit(value) {}
```

Property Value

Number

An integer value specifying time-out in milliseconds. The default is 10000 (10 seconds).

Remarks

The grid layout algorithm arranges graph nodes and links along the longest path in the graph. Large graphs with many links could contain hundreds of thousands of possible paths, so finding the longest one would be very time-consuming. When the time-out set with this property occurs, the path-finding routine stops and the layout algorithm uses the longest path known at that moment. Usually, for graphs containing several hundreds of nodes, 1-2 seconds are enough to find a path long enough to produce a nice layout.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace
useLongestPath Property

GridLayout.useLongestPath Property

Gets or sets a value indicating whether the algorithm should arrange nodes around the longest path in the graph.

Namespace: MindFusion.Graphs
File: GridLayout.js

Syntax

```
JavaScript
get useLongestPath() {}
set useLongestPath(value) {}
```

Property Value

Boolean

true to look for the longest path in the graph and arrange nodes around it; otherwise, **false**.

Remarks

Initially, the algorithm places nodes and links from the longest path in the graph at the grid diagonal, and arranges the rest of the nodes around the diagonal. Large graphs with many links could contain hundreds of thousands of possible paths, so finding the longest one would be very time-consuming. Disabling this property will let you skip the path-finding part of the algorithm, which will make it work much faster.

See Also

GridLayout Members
GridLayout Class
MindFusion.Graphs Namespace
timeLimit Property

LayeredLayout Class

Implements a layered graph layout algorithm.

Syntax

JavaScript

```
// class
LayeredLayout.prototype = {}
```

Remarks

This algorithm assigns diagram nodes to distinct horizontal or vertical layers. While arranging the layers, the layout routine seeks to meet the following criteria:

- connected nodes must be placed close together;
- links must flow in one direction if possible;
- links must cross as few layers as possible;
- links must not cross other links;

To apply this layout to a diagram, create a **LayeredLayout** instance, set its members and invoke the arrange method of the Diagram class. The method lays out all subgraphs in a diagram.

The final appearance of arranged graphs depends mostly on the direction value, which defines the overall orientation of the layout. The layerDistance and nodeDistance values specify how much space to leave between adjacent layers and adjacent nodes in a layer.

Example

The following example applies a **LayeredLayout** to a diagram:

JavaScript

```
var layout = new MindFusion.Graphs.LayeredLayout();
layout.direction = MindFusion.Graphs.LayoutDirection.TopToBottom;
layout.siftingRounds = 3;
diagram.arrange(layout);
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
MindFusion.Graphs.LayeredLayout

See Also

LayeredLayout Members
MindFusion.Graphs Namespace

LayeredLayout Members

Implements a layered graph layout algorithm.

The following tables list the members exposed by the LayeredLayout type.

Public Constructors

	Name	Description
	LayeredLayout	Initializes a new instance of the **LayeredLayout** class.

Public Properties

	Name	Description
	anchoring	Gets or sets a value indicating how to align links to the anchor points of nodes.
	direction	Gets or sets the orientation of the arranged graph.
	endNode	Gets or sets the node that should be placed in the last layer of the arranged graph.
	keepGroupLayout	Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.
	layerDistance	Gets or sets the desired distance between layer axis lines.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.
	nodeDistance	Gets or sets the space to leave between adjacent nodes on the same level.
	siftingRounds	Gets or sets how many times to perform pair-swapping iterations on each layer.
	startNode	Gets or sets the node that should be placed in the first layer of the arranged graph.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout Constructor

Initializes a new instance of the LayeredLayout class.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

JavaScript

```
function LayeredLayout ()
```

See Also

LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout Methods

LayeredLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

JavaScript

```
function arrange (graph)
```

Parameters

graph
The Graph to arrange.

Remarks

This method is called internally by Diagram.arrange and ContainerNode.arrange.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout Properties

LayeredLayout.anchoring Property

Gets or sets a value indicating how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

```
JavaScript
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the Anchoring enumeration. The default value is **Ignore**.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.direction Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

```
JavaScript
get direction() {}
set direction(value) {}
```

Property Value

LayoutDirection

A member of the LayoutDirection enumeration. The default value is **TopToBottom**.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.endNode Property

Gets or sets the node that should be placed in the last layer of the arranged graph.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

```
JavaScript
get endNode() {}
set endNode(value) {}
```

Property Value

DiagramNode

A DiagramNode representing the end node.

Remarks

This value is honored only if startNode is specified too.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.keepGroupLayout Property

Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

```
JavaScript
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group of attached nodes should be treated as a single vertex in the arranged graph, or **false** otherwise.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.layerDistance Property

Gets or sets the desired distance between layer axis lines.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

```
JavaScript
get layerDistance() {}
set layerDistance(value) {}
```

Property Value

Number

The desired distance between layer axis lines.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.multipleGraphsPlacement Property

Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

```
JavaScript
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration. The default value is **Horizontal**.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.nodeDistance Property

Gets or sets the space to leave between adjacent nodes on the same level.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

JavaScript

```
get nodeDistance() {}
set nodeDistance(value) {}
```

Property Value

Number

The space to leave between adjacent nodes on the same level.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.siftingRounds Property

Gets or sets how many times to perform pair-swapping iterations on each layer.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

JavaScript

```
get siftingRounds() {}
set siftingRounds(value) {}
```

Property Value

Number

An integer value specifying how many pair-swapping iterations to perform. The default is 1.

Remarks

The algorithm swaps node pairs on each layer aiming to decrease the number of crossing links. Assigning a higher value to this property helps decrease the number of crossings, at the cost of longer execution times.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace

LayeredLayout.startNode Property

Gets or sets the node that should be placed in the first layer of the arranged graph.

Namespace: MindFusion.Graphs
File: LayeredLayout.js

Syntax

JavaScript

```
get startNode() {}
set startNode(value) {}
```

Property Value

DiagramNode

A DiagramNode representing the start node.

Remarks

If there is no start node specified, the algorithm selects one automatically based on graph's topological order.

See Also

LayeredLayout Members
LayeredLayout Class
MindFusion.Graphs Namespace
endNode Property

Layout Class

Superclass of graph layout algorithms that defines their common properties.

Syntax

JavaScript

```
// class
Layout.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
MindFusion.Graphs.BorderedTreeLayout
MindFusion.Graphs.FlowchartLayout
MindFusion.Graphs.FractalLayout
MindFusion.Graphs.GridLayout
MindFusion.Graphs.LayeredLayout
MindFusion.Graphs.OrthogonalLayout
MindFusion.Graphs.RadialTreeLayout
MindFusion.Graphs.SpringLayout
MindFusion.Graphs.TopologicalLayout
MindFusion.Graphs.TreeLayout
MindFusion.Graphs.TreeMapLayout

See Also

Layout Members
MindFusion.Graphs Namespace

Layout Members

Superclass of graph layout algorithms that defines their common properties.

The following tables list the members exposed by the Layout type.

Public Properties

	Name	Description
	anchoring	Gets or sets how to align links to the anchor points of nodes.
	keepGroupLayout	Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object.
	multipleGraphsPlacement	Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other.

See Also

Layout Class
MindFusion.Graphs Namespace

Layout Properties

Layout.anchoring Property

Gets or sets how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: Layout.js

Syntax

JavaScript

```
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the LayoutDirection enumeration.

See Also

Layout Members
Layout Class
MindFusion.Graphs Namespace

Layout.keepGroupLayout Property

Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: Layout.js

Syntax

```
JavaScript
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group should be processed as an integral graph node, or false otherwise. The default is false.

See Also

Layout Members
Layout Class
MindFusion.Graphs Namespace

Layout.margins Property

Gets or sets the margins' size around individual subgraphs processed by this layout object.

Namespace: MindFusion.Graphs
File: Layout.js

Syntax

```
JavaScript
get margins() {}
set margins(value) {}
```

Property Value

Number

A number specifying size of margins around arranged graphs.

See Also

Layout Members
Layout Class
MindFusion.Graphs Namespace

Layout.multipleGraphsPlacement Property

Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: Layout.js

Syntax

JavaScript

```
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration.

See Also

Layout Members
Layout Class
MindFusion.Graphs Namespace

LayoutDirection Enumeration

Specifies in what direction to place nodes processed by a layout algorithm.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

JavaScript

```
// enum
LayoutDirection = {}
```

Members

	Member name	Description
	BottomToTop	Indicates a bottom-to-top layout direction.
	LeftToRight	Indicates a left-to-right layout direction.
	RightToLeft	Indicates a right-to-left layout direction.
	TopToBottom	Indicates a top-to-bottom layout direction.

Remarks

Members of this enumeration can be assigned to the direction property of layout classes.

See Also

MindFusion.Graphs Namespace

MultipleGraphsPlacement Enumeration

Specifies placement of graph connected components relatively to each other.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

JavaScript
`// enum` `MultipleGraphsPlacement = {}`

Members

	Member name	Description
	Horizontal	Indicates that subgraphs should be placed in a row.
	MinimalArea	Indicates that subgraphs should be packed in a square-like area.
	Vertical	Indicates that subgraphs should be placed in a column.

Remarks

Members of this enumeration can be assigned to the multipleGraphsPlacement property of layout classes.

See Also

MindFusion.Graphs Namespace

Orientation Enumeration

Specifies general layout orientation.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

JavaScript
`// enum` `Orientation = {}`

Members

	Member name	Description
	Horizontal	Specifies horizonal orientation.
	Vertical	Specifies vertical orientation.

Remarks

Members of this enumeration can be assigned to the orientation property of layout classes.

See Also

MindFusion.Graphs Namespace

OrthogonalLayout Class

Implements orthogonal graph layout algorithm.Each link is drawn as a chain of alternating horizontal and vertical segments.Nodes are placed in a way that facilitates few links bends and crossings.

Syntax

JavaScript

```
// class
OrthogonalLayout.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.OrthogonalLayout

See Also

OrthogonalLayout Members
MindFusion.Graphs Namespace

OrthogonalLayout Members

Implements orthogonal graph layout algorithm.Each link is drawn as a chain of alternating horizontal and vertical segments.Nodes are placed in a way that facilitates few links bends and crossings.

The following tables list the members exposed by the OrthogonalLayout type.

Public Constructors

	Name	Description
	OrthogonalLayout	Initializes a new instance of the OrthogonalLayout class.

Public Properties

	Name	Description
	anchoring	Gets or sets how to align links to the anchor points of nodes. (Inherited from Layout.)
	keepGroupLayout	Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph. (Inherited from Layout.)
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	minLaneSize	Gets or sets a value indicating the minimum size of a lane.
	multipleGraphsPlacement	Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other. (Inherited from Layout.)

	padding	Gets or sets a value indicating how much space to leave between nodes in adjacent lanes.
	refine	Gets or sets a value indicating whether to refine the layout.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

OrthogonalLayout Class
MindFusion.Graphs Namespace

OrthogonalLayout Constructor

Initializes a new instance of the OrthogonalLayout class.

Namespace: MindFusion.Graphs
File: OrthogonalLayout.js

Syntax

JavaScript

```
function OrthogonalLayout ()
```

See Also

OrthogonalLayout Class
MindFusion.Graphs Namespace

OrthogonalLayout Methods

OrthogonalLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: OrthogonalLayout.js

Syntax

JavaScript

```
function arrange (graph)
```

Parameters

graph
Graph. The graph to arrange.

See Also

OrthogonalLayout Members
OrthogonalLayout Class
MindFusion.Graphs Namespace

OrthogonalLayout Properties

OrthogonalLayout.minLaneSize Property

Gets or sets a value indicating the minimum size of a lane.

Namespace: MindFusion.Graphs
File: OrthogonalLayout.js

Syntax

JavaScript

```
get minLaneSize() {}
set minLaneSize(value) {}
```

Property Value

Number

The minimum size of a lane.

See Also

OrthogonalLayout Members
OrthogonalLayout Class
MindFusion.Graphs Namespace

OrthogonalLayout.padding Property

Gets or sets a value indicating how much space to leave between nodes in adjacent lanes.

Namespace: MindFusion.Graphs
File: OrthogonalLayout.js

Syntax

JavaScript

```
get padding() {}
set padding(value) {}
```

Property Value

Number

The space to leave between nodes in adjacent lanes.

See Also

OrthogonalLayout Members
OrthogonalLayout Class
MindFusion.Graphs Namespace

OrthogonalLayout.refine Property

Gets or sets a value indicating whether to refine the layout.

Namespace: MindFusion.Graphs
File: OrthogonalLayout.js

Syntax

JavaScript

```
get refine() {}
set refine(value) {}
```

Property Value

Boolean

True to refine the layout, or false otherwise.

See Also

OrthogonalLayout Members
OrthogonalLayout Class
MindFusion.Graphs Namespace

Path Class

Initializes a new instance of the Path class.

Syntax

JavaScript

```
// class
Path.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Path

See Also

Path Members
MindFusion.Graphs Namespace

Path Members

Initializes a new instance of the Path class.

The following tables list the members exposed by the Path type.

Public Constructors

	Name	Description
	Path	Initializes a new instance of the Path class.

See Also

Path Class
MindFusion.Graphs Namespace

Path Constructor

Initializes a new instance of the Path class.

Namespace: MindFusion.Graphs
File: Path.js

Syntax

JavaScript

```
function Path ()
```

See Also

Path Class
MindFusion.Graphs Namespace

PathFinder Class

Initializes a new instance of the PathFinder class.

Syntax

JavaScript

```
// class
PathFinder.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.PathFinder

See Also

PathFinder Members
MindFusion.Graphs Namespace

PathFinder Members

Initializes a new instance of the PathFinder class.

The following tables list the members exposed by the PathFinder type.

Public Constructors

	Name	Description
	PathFinder	Initializes a new instance of the PathFinder class.

Public Methods

	Name	Description
	findAllPaths	Finds and returns all paths starting from node 'from' andending at node 'to'. Returns empty collection if nopath exists.

See Also

PathFinder Class
MindFusion.Graphs Namespace

PathFinder Constructor

Initializes a new instance of the PathFinder class.

Namespace: MindFusion.Graphs
File: PathFinder.js

Syntax

JavaScript

```
function PathFinder ()
```

See Also

PathFinder Class
MindFusion.Graphs Namespace

PathFinder Methods

PathFinder.findAllPaths Method

Finds and returns all paths starting from node 'from' andending at node 'to'. Returns empty collection if nopath exists.

Namespace: MindFusion.Graphs
File: PathFinder.js

Syntax

JavaScript

```
function findAllPaths ()
```

See Also

PathFinder Members
PathFinder Class
MindFusion.Graphs Namespace

RadialTreeLayout Class

Implements a radial tree layout algorithm.

Syntax

JavaScript

```
// class
RadialTreeLayout.prototype = {}
```

Remarks

This layout class arranges tree levels in concentric circles around the root. The levelDistance property specifies the distance between adjacent circles. The direction property specifies placement of first child node. Set stretchFactor to a value larger than 1 in order to stretch level circles into ellipses, providing more space for the layout when using wider nodes.

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.RadialTreeLayout

See Also

RadialTreeLayout Members
MindFusion.Graphs Namespace

RadialTreeLayout Members

Implements a radial tree layout algorithm.

The following tables list the members exposed by the RadialTreeLayout type.

Public Constructors

	Name	Description
	RadialTreeLayout	Initializes a new instance of the RadialTreeLayout class.

Public Properties

	Name	Description
	anchoring	Gets or sets how to align links to the anchor points of nodes. (Inherited from Layout.)
	direction	Gets or sets the orientation of the arranged graph.

	keepGroupLayout	Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph. (Inherited from Layout.)
	keepRootPosition	Gets or sets a value indicating whether to keep the root node at its original position.
	levelDistance	Gets or sets the space to leave between adjacent levels of the tree.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other. (Inherited from Layout.)
	reversedLinks	Gets or sets a value indicating the link direction that relates parent nodes to child nodes.
	stretchFactor	Gets or sets the stretch factor in radial layouts.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout Constructor

Initializes a new instance of the RadialTreeLayout class.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

JavaScript

```
function RadialTreeLayout ()
```

See Also

RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout Methods

RadialTreeLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

```
JavaScript
function arrange (graph)
```

Parameters

graph
Graph. The graph to arrange.

See Also

RadialTreeLayout Members
RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout Properties

RadialTreeLayout.direction Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

```
JavaScript
get direction() {}
set direction(value) {}
```

Property Value

LayoutDirection

A member of the LayoutDirection enumeration.

Remarks

In radial layouts, the **direction** property specifies placement of first child node relatively to the root.

See Also

RadialTreeLayout Members
RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout.keepRootPosition Property

Gets or sets a value indicating whether to keep the root node at its original position.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

```
JavaScript
get keepRootPosition() {}
set keepRootPosition(value) {}
```

Property Value

Boolean

true if the root node should be kept at its original position, or **false** otherwise.

See Also

RadialTreeLayout Members
RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout.levelDistance Property

Gets or sets the space to leave between adjacent levels of the tree.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

```
JavaScript
get levelDistance() {}
set levelDistance(value) {}
```

Property Value

Number

The space to leave between adjacent levels of the tree.

See Also

RadialTreeLayout Members
RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout.reversedLinks Property

Gets or sets a value indicating the link direction that relates parent nodes to child nodes.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

JavaScript

```
get reversedLinks() {}
set reversedLinks(value) {}
```

Property Value

Boolean

true to indicate reversed direction (links point to parent nodes), or **false** otherwise.

See Also

RadialTreeLayout Members
RadialTreeLayout Class
MindFusion.Graphs Namespace

RadialTreeLayout.stretchFactor Property

Gets or sets the stretch factor in radial layouts.

Namespace: MindFusion.Graphs
File: RadialTreeLayout.js

Syntax

JavaScript

```
get stretchFactor() {}
set stretchFactor(value) {}
```

Property Value

Number

A stretch factor that lets you set ellipse eccentricity. The default value is 1.

Remarks

Set **stretchFactor** to a value larger than 1 in order to stretch level circles into ellipses, providing more space for the layout when using wider nodes.

See Also

RadialTreeLayout Members
RadialTreeLayout Class
MindFusion.Graphs Namespace

SpringLayout Class

Implements the Spring-Embedder graph layout algorithm.

Syntax

JavaScript

```
// class
SpringLayout.prototype = {}
```

Remarks

To apply this layout to a diagram, create a **SpringLayout** instance, set its members and invoke the arrange method of the Diagram class. The method lays out all subgraphs in the diagram.

Spring-Embedder produces layouts having uniform distribution of nodes by simulating a physical system in which nodes repulse each other and the links between them act as confining springs. Nodes are moved around in an iterative process. The forces that act on a node are calculated, taking in account the positions of surrounding nodes and links from the previous iteration.

The final appearance of arranged graphs depends on the nodeDistance value that specifies how nodes are distributed by the layout routine. The layout routine runs as much iterations as set in iterations, calculating better node positions at each step.

Example

The following example applies a **SpringLayout** to a diagram:

JavaScript

```
var layout = new MindFusion.Graphs.SpringLayout();
layout.iterations = 100;
diagram.arrange(layout);
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.SpringLayout

See Also

SpringLayout Members
MindFusion.Graphs Namespace

SpringLayout Members

Implements the Spring-Embedder graph layout algorithm.

The following tables list the members exposed by the SpringLayout type.

Public Constructors

	Name	Description
	SpringLayout	Initializes a new instance of the **SpringLayout** class.

Public Properties

	Name	Description
	anchoring	Gets or sets a value indicating how to align links to the anchor points of nodes.
	iterations	Gets or sets the number of iterations to run the layout routine.

	keepGroupLayout	Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.
	nodeDistance	Gets or sets the desired distance between nodes.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout Constructor

Initializes a new instance of the SpringLayout class.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
function SpringLayout ()
```

See Also

SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout Methods

SpringLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
function arrange (graph)
```

Parameters

graph
The Graph to arrange.

Remarks

This method is called internally by Diagram.arrange and ContainerNode.arrange.

See Also

SpringLayout Members
SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout Properties

SpringLayout.anchoring Property

Gets or sets a value indicating how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the Anchoring enumeration. The default value is **Ignore**.

See Also

SpringLayout Members
SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout.iterations Property

Gets or sets the number of iterations to run the layout routine.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
get iterations() {}
set iterations(value) {}
```

Property Value

Number

The number of iterations to run the layout routine.

See Also

SpringLayout Members
SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout.keepGroupLayout Property

Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group of attached nodes should be treated as a single vertex in the arranged graph, or **false** otherwise.

See Also

SpringLayout Members
SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout.multipleGraphsPlacement Property

Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration. The default value is **Horizontal**.

See Also

SpringLayout Members
SpringLayout Class
MindFusion.Graphs Namespace

SpringLayout.nodeDistance Property

Gets or sets the desired distance between nodes.

Namespace: MindFusion.Graphs
File: SpringLayout.js

Syntax

JavaScript

```
get nodeDistance() {}
set nodeDistance(value) {}
```

Property Value

Number

The desired distance between nodes.

See Also

SpringLayout Members
SpringLayout Class
MindFusion.Graphs Namespace

TopologicalLayout Class

Applies topological ordering to the graph. This layout makes it easy to discern graph features such as overall flow direction, cycles and nested cycles.

Syntax

JavaScript

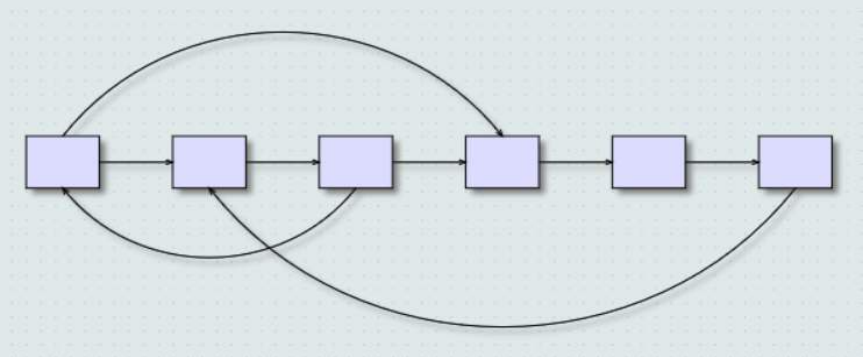
```
// class
TopologicalLayout.prototype = {}
```

Remarks

Topological ordering of graph nodes is one where link's origin node is placed before link's destination. TopologicalLayout arranges nodes in a row or a column (depending on the value of the direction property) in such a way that there are no backward links when the graph is acyclic. If the graph contains cycles, the algorithms selects ordering with as few backward links as possible.

Links that connect non-adjacent nodes are rendered as arcs. Links connecting adjacent nodes can be rendered either as arcs or as straight lines, depending on the bendAdjacentLinks property. The amplitude of link arcs is proportional to the distance between respective connected nodes. All forward

links are rendered at one side of the nodes and all back links are rendered at the opposite side. Thus, all bent links follow the same rotation direction.



Example

The following example applies TopologicalLayout to the diagram:

JavaScript

```
var layout = new MindFusion.Graphs.TopologicalLayout();
layout.direction = MindFusion.Graphs.LayoutDirection.LeftToRight;
diagram.arrange(layout);
diagram.resizeToFitItems();
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.TopologicalLayout

See Also

TopologicalLayout Members
MindFusion.Graphs Namespace

TopologicalLayout Members

Applies topological ordering to the graph. This layout makes it easy to discern graph features such as overall flow direction, cycles and nested cycles.

The following tables list the members exposed by the TopologicalLayout type.

Public Constructors

	Name	Description
	TopologicalLayout	Initializes a new instance of the **TopologicalLayout** class.

Public Properties

	Name	Description
	anchoring	Gets or sets how to align links to the anchor points of nodes. (Inherited from Layout.)
	bendAdjacentLinks	Gets or sets a value indicating whether to bend links connecting adjacent nodes or draw them as straight lines.

	direction	Gets or sets the orientation of the arranged graph.
	keepGroupLayout	Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph. (Inherited from Layout.)
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other. (Inherited from Layout.)
	nodeDistance	Gets or sets the desired distance between adjacent nodes.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.
	init	Returns a Builder object used to configure this **TopologicalLayout** instance.
S	With	Returns a Builder object used to configure and create new **TopologicalLayout** instances.

See Also

TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout Constructor

Initializes a new instance of the TopologicalLayout class.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

JavaScript

```
function TopologicalLayout ()
```

See Also

TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout Methods

TopologicalLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

```
JavaScript
function arrange (graph)
```

Parameters

graph
The Graph to arrange.

Remarks

This method is called internally by Diagram.arrange and ContainerNode.arrange.

See Also

TopologicalLayout Members
TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout.init Method

Returns a Builder object used to configure this TopologicalLayout instance.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

```
JavaScript
function init ()
```

Return Value

A Builder instance.

See Also

TopologicalLayout Members
TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout.With Method

Returns a Builder object used to configure and create new TopologicalLayout instances.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

JavaScript

```
function With ()
```

Return Value

A Builder instance.

See Also

TopologicalLayout Members
TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout Properties

TopologicalLayout.bendAdjacentLinks Property

Gets or sets a value indicating whether to bend links connecting adjacent nodes or draw them as straight lines.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

JavaScript

```
get bendAdjacentLinks() {}
set bendAdjacentLinks(value) {}
```

Property Value

Boolean

true to bend links connecting adjacent nodes, or false otherwise.

See Also

TopologicalLayout Members
TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout.direction Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

JavaScript

```
get direction() {}
set direction(value) {}
```

Property Value

LayoutDirection

A member of the LayoutDirection enumeration.

See Also

TopologicalLayout Members
TopologicalLayout Class
MindFusion.Graphs Namespace

TopologicalLayout.nodeDistance Property

Gets or sets the desired distance between adjacent nodes.

Namespace: MindFusion.Graphs
File: TopologicalLayout.js

Syntax

JavaScript

```
get nodeDistance() {}
set nodeDistance(value) {}
```

Property Value

Number

The desired distance between adjacent nodes.

See Also

TopologicalLayout Members
TopologicalLayout Class
MindFusion.Graphs Namespace

TreeLayout Class

Implements algorithms for arranging tree structures.

Syntax

JavaScript

```
// class
TreeLayout.prototype = {}
```

Remarks

To apply this layout to a diagram, create a **TreeLayout** instance, set its members and invoke the arrange method of the Diagram class. The method lays out all subtrees in a diagram.

As long as the graph is a tree, the algorithm can find the root node automatically. The distance between the root and the second level and between all subsequent levels can be set through the levelDistance field. The distance between adjacent nodes in the same level is set through nodeDistance.

In addition to arranging the tree nodes, **TreeLayout** also sets the shape of links as set through its linkType property.

Example

The following example applies a **TreeLayout** to a diagram:

JavaScript

```
var layout = new MindFusion.Graphs.TreeLayout();
layout.direction = MindFusion.Graphs.LayoutDirection.TopToBottom;
linkType = MindFusion.Graphs.TreeLayoutLinkType.Cascading;
diagram.arrange(layout);
```

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.TreeLayout

See Also

TreeLayout Members
MindFusion.Graphs Namespace

TreeLayout Members

Implements algorithms for arranging tree structures.

The following tables list the members exposed by the TreeLayout type.

Public Constructors

	Name	Description
	TreeLayout	Initializes a new instance of the **TreeLayout** class.

Public Properties

	Name	Description
	anchoring	Gets or sets a value indicating how to align links to the anchor points of nodes.
	compactAssistants	Gets or sets a value indicating whether the "assistant" nodes on the same side of a single parentare arranged as close to each other as possible.
	direction	Gets or sets the orientation of the arranged graph.
	enableAssistants	Gets or sets a value Indicating whether the "assistant" trait is regarded when performing the layout.
	keepGroupLayout	Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.
	keepRootPosition	Gets or sets a value indicating whether to keep the root node at its original position.

	levelDistance	Gets or sets the space to leave between adjacent levels of the tree.
	linkType	Gets or sets the type of the links in the arranged tree.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.
	nodeDistance	Gets or sets the space to leave between adjacent nodes on the same level.
	reversedLinks	Gets or sets a value indicating the link direction that relates parent nodes to child nodes.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout Constructor

Initializes a new instance of the TreeLayout class.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
function TreeLayout ()
```

See Also

TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout Methods

TreeLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

```
JavaScript
function arrange (graph)
```

Parameters

graph
The Graph to arrange.

Remarks

This method is called internally by Diagram.arrange and ContainerNode.arrange.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout Properties

TreeLayout.anchoring Property

Gets or sets a value indicating how to align links to the anchor points of nodes.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

```
JavaScript
get anchoring() {}
set anchoring(value) {}
```

Property Value

Anchoring

A member of the Anchoring enumeration. The default value is **Ignore**.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.compactAssistants Property

Gets or sets a value indicating whether the "assistant" nodes on the same side of a single parentare arranged as close to each other as possible.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get compactAssistants() {}
set compactAssistants(value) {}
```

Property Value

Boolean

true if the "assistant" nodes on the same side of a single parentare arranged as close to each other as possible, or false otherwise.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.direction Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get direction() {}
set direction(value) {}
```

Property Value

LayoutDirection

A member of the LayoutDirection enumeration. The default value is **TopToBottom**.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.enableAssistants Property

Gets or sets a value Indicating whether the "assistant" trait is regarded when performing the layout.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get enableAssistants() {}
set enableAssistants(value) {}
```

Property Value

Boolean

true if the "assistant" trait is regarded when performing the layout, or false otherwise.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.keepGroupLayout Property

Gets or sets a value indicating whether to treat each group of nodes as a single vertex in the arranged graph.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get keepGroupLayout() {}
set keepGroupLayout(value) {}
```

Property Value

Boolean

true if each group of attached nodes should be treated as a single vertex in the arranged graph, or **false** otherwise.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.keepRootPosition Property

Gets or sets a value indicating whether to keep the root node at its original position.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get keepRootPosition() {}
set keepRootPosition(value) {}
```

Property Value

Boolean

true if the root node should be kept at its original position, or **false** otherwise.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.levelDistance Property

Gets or sets the space to leave between adjacent levels of the tree.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get levelDistance() {}
set levelDistance(value) {}
```

Property Value

Number

The space to leave between adjacent levels of the tree.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.linkType Property

Gets or sets the type of the links in the arranged tree.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get linkType() {}
set linkType(value) {}
```

Property Value

TreeLayoutLinkType

A member of the TreeLayoutLinkType enumeration. The default value is **Default**.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.multipleGraphsPlacement Property

Gets or sets a value indicating how multiple independent graphs in the diagram should be positioned relatively to each other.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get multipleGraphsPlacement() {}
set multipleGraphsPlacement(value) {}
```

Property Value

MultipleGraphsPlacement

A member of the MultipleGraphsPlacement enumeration.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.nodeDistance Property

Gets or sets the space to leave between adjacent nodes on the same level.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get nodeDistance() {}
set nodeDistance(value) {}
```

Property Value

Number

The space to leave between adjacent nodes on the same level.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayout.reversedLinks Property

Gets or sets a value indicating the link direction that relates parent nodes to child nodes.

Namespace: MindFusion.Graphs
File: TreeLayout.js

Syntax

JavaScript

```
get reversedLinks() {}
set reversedLinks(value) {}
```

Property Value

Boolean

true to indicate reversed direction (links point to parent nodes), or **false** otherwise.

See Also

TreeLayout Members
TreeLayout Class
MindFusion.Graphs Namespace

TreeLayoutLinkType Enumeration

Specifies the shape of the diagram links after they are laid out.

Namespace: MindFusion.Graphs
File: Enum.js

Syntax

JavaScript

```
// enum
TreeLayoutLinkType = {}
```

Members

	Member name	Description
	Cascading	Indicates that links will be arranged as cascading and attached to the middles of the adjoining node sides.
	Default	Indicates that links will be arranged to point to the centers of the related nodes.
	Straight	Indicates that links will be attached to the middles of the adjoining node sides.

Remarks

Members of this enumeration can be assigned to the linkType property of TreeLayout.

See Also

MindFusion.Graphs Namespace

TreeMapLayout Class

Performs tree-map layout on a graph.

Syntax

JavaScript

```
// class
TreeMapLayout.prototype = {}
```

Remarks

Tree maps represent hierarchies by nesting child nodes within their parents, where the areas of leaf nodes are proportional to their Weight values. Unlike other layout algorithms, **TreeMapLayout** expects hierarchies to be defined via grouping or containment (see attachTo method and ContainerNode class), and will ignore any links in the diagram.

The diagram area covered by the topmost nodes in a hierarchy is specified via the layoutArea property. By default, the layout tries to keep the ratio of node sides as close as possible to one. However this could make it hard to distinguish separate levels of the hierarchy. To alleviate that, set squarify to **false**, and child nodes will be arranged either as a row or a column inside their parent node, alternating directions for each level. The drawback is that when weight ratios differ greatly or nodes contain many children, some nodes could end up with very narrow rectangles.

The padding size between parent nodes and their outermost children is specified via the containerPadding property. The padding size between child nodes is specified via padding.

Inheritance Hierarchy

MindFusion.Graphs.Layout
  MindFusion.Graphs.TreeMapLayout

See Also

TreeMapLayout Members
MindFusion.Graphs Namespace

TreeMapLayout Members

Performs tree-map layout on a graph.

The following tables list the members exposed by the TreeMapLayout type.

Public Constructors

	Name	Description
	TreeMapLayout	Initializes a new instance of the **TreeMapLayout** class.

Public Properties

	Name	Description

	anchoring	Gets or sets how to align links to the anchor points of nodes. (Inherited from Layout.)
	containerPadding	Gets or sets the padding inside containers.
	keepGroupLayout	Gets or sets whether to treat each group of attached nodes as a single vertex in the arranged graph. (Inherited from Layout.)
	layoutArea	Gets or sets the rectangle in which the layout will try to arrange nodes.
	margins	Gets or sets the margins' size around individual subgraphs processed by this layout object. (Inherited from Layout.)
	multipleGraphsPlacement	Gets or sets how multiple independent graphs in the diagram should be positioned relatively to each other. (Inherited from Layout.)
	orientation	Gets or sets the orientation of the arranged graph.
	padding	Gets or sets the distance between adjacent nodes.
	squarify	Gets or sets a value indicating whether the layout should attempt to keep the dimension ratio of nodes closer to 1.

Public Methods

	Name	Description
	arrange	Applies the layout to the specified graph.

See Also

TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout Constructor

Initializes a new instance of the TreeMapLayout class.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

JavaScript

```
function TreeMapLayout ()
```

See Also

TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout Methods

TreeMapLayout.arrange Method

Applies the layout to the specified graph.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

```
JavaScript
function arrange (graph)
```

Parameters

graph
The Graph to arrange.

Remarks

This method is called internally by Diagram.arrange and ContainerNode.arrange.

See Also

TreeMapLayout Members
TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout Properties

TreeMapLayout.containerPadding Property

Gets or sets the padding inside containers.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

```
JavaScript
get containerPadding() {}
set containerPadding(value) {}
```

Property Value

Number

The padding inside containers.

See Also

TreeMapLayout Members
TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout.layoutArea Property

Gets or sets the rectangle in which the layout will try to arrange nodes.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

```
JavaScript
get layoutArea() {}
set layoutArea(value) {}
```

Property Value

Rect

A Rect instance specifying the area in which the layout will try to arrange nodes.

See Also

TreeMapLayout Members
TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout.orientation Property

Gets or sets the orientation of the arranged graph.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

```
JavaScript
get orientation() {}
set orientation(value) {}
```

Property Value

Orientation

A member of the Orientation enumeration.

See Also

TreeMapLayout Members
TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout.padding Property

Gets or sets the distance between adjacent nodes.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

JavaScript

```
get padding() {}
set padding(value) {}
```

Property Value

Number

The distance between adjacent nodes.

See Also

TreeMapLayout Members
TreeMapLayout Class
MindFusion.Graphs Namespace

TreeMapLayout.squarify Property

Gets or sets a value indicating whether the layout should attempt to keep the dimension ratio of nodes closer to 1.

Namespace: MindFusion.Graphs
File: TreeMapLayout.js

Syntax

JavaScript

```
get squarify() {}
set squarify(value) {}
```

Property Value

Boolean

true if the layout should attempt to keep the dimension ratio of nodes closer to 1, or **false** otherwise.

See Also

TreeMapLayout Members
TreeMapLayout Class
MindFusion.Graphs Namespace

Vertex Class

Represents a vertex in a graph.

Syntax

```
JavaScript
// class
Vertex.prototype = {}
```

Inheritance Hierarchy

MindFusion.Graphs.Vertex

See Also

Vertex Members
MindFusion.Graphs Namespace

Vertex Members

Represents a vertex in a graph.

The following tables list the members exposed by the Vertex type.

Public Constructors

	Name	Description
	Vertex	Initializes a new instance of the **Vertex** class.

Public Fields

	Name	Description
	edges	A list containing all edges incident with this vertex.
	inEdges	A list containing all incoming edges.
	outEdges	A list containing all outgoing edges.
	weight	The relative weight of this vertex.

Public Methods

	Name	Description
	adjacentTo	Checks if there is an edge connecting this vertex with the specified vertex.
	degree	Returns the number of incident edges.
	getCommonEdge	Finds the edge connecting the current vertex with the specified vertex.
	getNeighbors	Returns an array containing all vertices adjacent to the current vertex.
	incidentWith	Checks if the current vertex is incident with the specified edge.

See Also

Vertex Class
MindFusion.Graphs Namespace

Vertex Fields

Vertex.edges Field

A list containing all edges incident with this vertex.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
var edges
```

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.inEdges Field

A list containing all incoming edges.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
var inEdges
```

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.outEdges Field

A list containing all outgoing edges.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
var outEdges
```

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.weight Field

The relative weight of this vertex.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
var weight
```

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex Constructor

Initializes a new instance of the Vertex class.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
function Vertex ()
```

See Also

Vertex Class
MindFusion.Graphs Namespace

Vertex Methods

Vertex.adjacentTo Method

Checks if there is an edge connecting this vertex with the specified vertex.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
function adjacentTo (vertex)
```

Parameters

vertex
Vertex. The vertex to check.

Return Value

true if the vertices are connected; otherwise, **false**.

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.degree Method

Returns the number of incident edges.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
function degree ()
```

Return Value

Number. The number of incident edges.

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.getCommonEdge Method

Finds the edge connecting the current vertex with the specified vertex.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

```
JavaScript
function getCommonEdge (vertex)
```

Parameters

vertex
Vertex. The vertex to find an edge for.

Return Value

The edge connecting the vertices or **null**.

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.getNeighbors Method

Returns an array containing all vertices adjacent to the current vertex.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

JavaScript

```
function getNeighbors ()
```

Return Value

Array. A list with all neighbors.

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Vertex.incidentWith Method

Checks if the current vertex is incident with the specified edge.

Namespace: MindFusion.Graphs
File: Vertex.js

Syntax

JavaScript

```
function incidentWith (edge)
```

Parameters

edge
Edge. The edge to check.

Return Value

true if the vertex is incident with the edge; otherwise, **false**.

See Also

Vertex Members
Vertex Class
MindFusion.Graphs Namespace

Table of Predefined Shapes

Shape	Identifier	Common Uses
	Actor	A user An actor in use case model
	Alternative	An alternative in SDL An extraction in flowcharts Storage in quality management
	Arrow1	Use as desired
	Arrow2	Use as desired
	Arrow3	Transportation in quality management A reference point
	Arrow4	Use as desired
	Arrow5	Use as desired
	Arrow6	Use as desired
	Arrow7	A signal Electronic transmission of data
	Arrow8	Use as desired
	Arrow9	Use as desired
	BeginLoop	Begin of a loop

	BpmnComplex	BPMN Gateways: Complex Decision/Merge
	BpmnDataBasedXor	BPMN Gateways: Data-Based Exclusive Decision/Merge (XOR)
	BpmnEndCancel	BPMN Events: Cancel (End)
	BpmnEndCompensation	BPMN Events: Compensation (End)
	BpmnEndError	BPMN Events: Error (End)
	BpmnEndLink	BPMN Events: Link (End)
	BpmnEndMessage	BPMN Events: Message (End)
	BpmnEndMultiple	BPMN Events: Multiple (End)
	BpmnEndTerminate	BPMN Events: Terminate (End)
	BpmnEventBasedXor	BPMN Gateways: Event-Based Exclusive Decision/Merge (XOR)
	BpmnInclusive	BPMN Gateways: Inclusive Decision/Merge (OR)
	BpmnIntermediateCancel	BPMN Events: Cancel (Intermediate)
	BpmnIntermediateCompensation	BPMN Events: Compensation (Intermediate)
	BpmnIntermediateError	BPMN Events: Error (Intermediate)
	BpmnIntermediateLink	BPMN Events: Link (Intermediate)

	BpmnIntermediateMessage	BPMN Events: Message (Intermediate)
	BpmnIntermediateMultiple	BPMN Events: Multiple (Intermediate)
	BpmnIntermediateRule	BPMN Events: Rule (Intermediate)
	BpmnIntermediateTimer	BPMN Events: Timer (Intermediate)
	BpmnParallelFork	BPMN Gateways: Parallel Fork/Join (AND)
	BpmnStartLink	BPMN Events: Link (Start)
	BpmnStartMessage	BPMN Events: Message (Start)
	BpmnStartMultiple	BPMN Events: Multiple (Start)
	BpmnStartRule	BPMN Events: Rule (Start)
	BpmnStartTimer	BPMN Events: Timer (Start)
	Cloud	Object instance in data flow diagrams
	Collate	A collation in flowcharts
	ConeDown	Use as desired
	ConeUp	Use as desired
	ConnectedIssues	Connected issues in quality management

	CreateRequest	Request to create in SDL
	Cross	Use as desired
	Cube	Use as desired
	Cylinder	Use as desired
	Database	A database in flowcharts
	DataTransmition	Electronic transmission of data
	DDelay	Use as desired
	Decagon	Use as desired
	Decision	Decision in flowcharts A comparison in flowcharts Relationship in data flow diagrams
	Decision2	A decision in SDL
	Delay	A delay in process diagrams
	DInDelay	Use as desired
	DirectAccessStorage	A direct access storage
	DiskStorage	Data stored on disk in flowcharts
	Display	Data that is displayed for human use
	DividedEvent	An event
	DividedProcess	A process or document in flowcharts Entity in data flow diagrams

	Document	A document
	Donut	Use as desired
	DOutDelay	Use as desired
	Ellipse	A cross-reference within the same page of a flowchart Process in data flow diagrams State in data flow diagrams A continuation of the flowchart in SDL An use case Operation in quality management
	EndLoop	End of a loop
	ExternalOrganization	An external organization
	ExternalProcess	An external process
	File	A file
	FramedRectangle	Use as desired
	Gate	A gateway
	Heart	Use as desired
	Heptagon	Use as desired
	Hourglass	Use as desired
	Input	Data that is input by manual means
	InternalStorage	An internal storage
	Interrupt	An interrupt

	Interrupt2	An interrupt
	IsoProcess	Iso9000 process
	Junction	A summing junction A return in SDL
	Lightning	Use as desired
	LinedDocument	A document in flowcharts
	ManualOperation	Operation performed by a human
	Merge	A merger in flowcharts Inbound goods in quality management Non-electronic file Stop state in data flow diagrams
	MessageFromUser	A message from user side in SDL
	MessageToUser	A message to user side in SDL
	Microform	A microform
	MicroformProcessing	A microform processing
	MicroformRecording	A microform recording
	MultiDocument	A multiple document in flowcharts
	MultiProc	A multiple process or document in flowcharts
	Octagon	Use as desired
	OfflineStorage	An offline storage

	OffpageConnection	A cross-reference to another page of a flowchart
	OffpageReference	A cross-reference to another page of a flowchart A foreign module
	Or	An 'or' statement
	Pentagon	Use as desired
	Plaque	Use as desired
	PrimitiveFromCall	A primitive from call control in SDL
	PrimitiveToCall	A primitive to call control in SDL
	Procedure	Procedure in SDL
	Process	A process or document in flowcharts A file or data storage
	Prysm	Use as desired
	PunchedCard	A punched card
	PyramidDown	Use as desired
	PyramidUp	Use as desired
	Rectangle	A metric operation 'External interactor' in data flow diagrams Inspection in quality management
	RightTriangle	Use as desired
	RoundRect	A process or item in flowcharts An event An issue in quality management
	RSave	Input or output data in flowcharts

	Save	A save in SDL Input or output data in flowcharts
	Sort	A sort operation
	Sphere	Use as desired
	Star	Use as desired
	Star4Pointed	Use as desired
	Star5Pointed	Use as desired
	Star6Pointed	Use as desired
	Star7Pointed	Use as desired
	Star16Pointed	Use as desired
	Star24Pointed	Use as desired
	Star32Pointed	Use as desired
	Start	Start of a program flow A starting point in SDL
	Tape	Sequentially accessible data
	Terminator	Start or end of a program flow
	TransmittalTape	A transmittal tape
	Trapezoid	Use as desired

Licensing

For non-commercial use, you can disable evaluation mode by adding an https://mindfusion.dev/ link to your page, and assigning the link's id value to the **linkBackId** property of DiagramView. For example:

HTML

```
<div class="footer">
    ...
    <a id="mindfusionLink" href="https://mindfusion.dev/">h/t MindFusion</a>
</div>
```

JavaScript

```
diagramView.linkBackId = "mindfusionLink";
```

If you have a commercial license, copy your license key to a mindfusion_lic.txt file in same folder where the diagram html page is located. The file name and location can be changed by setting DiagramView's licenseLocation property. Alternatively, the key string can be assigned directly to view's **licenseKey** property.

MindFusion.Diagramming licenses are perpetual and royalty-free. When you purchase the component, you can find its license key string in confirmation e-mail, and also listed on Keys & Downloads page at our client portal.

Online Resources

Video Tutorials

- Create, edit, save and load diagrams using JavaScript and HTML
- A JavaScript application for server load monitoring
- Interactive org chart in JavaScript

Blog Posts

- A block diagram editor
- Organizational chart tutorial
- Using MindFusion JavaScript Components With WordPress Elementor Plugin
- Custom Diagram Nodes With Clipped Images

Technical Support

Detailed information about everything connected with software products offered by MindFusion - online documentation, demos, information about forthcoming products can be found at https://mindfusion.dev. The site is updated every day, so make sure you visit it regularly to catch up with the newest from us.

Apart from that, MindFusion employs a dedicated and technically proficient support team. This guarantees that all your questions and inquiries will be answered in quick and professional manner. They can be reached by e-mail at support@mindfusion.eu or by phone at +359 888 951737.

You can post a question on the MindFusion.Diagramming for JavaScript discussion board here:

https://mindfusion.dev/Forum/YaBB.pl?board=jsdiag_disc

or visit our Help Desk site:

https://mindfusion.dev/HelpDesk/index.php

We will be happy to hear any suggestions and recommendations you might have regarding MindFusion.Diagramming for JavaScript. We do take in account the opinion of every single person who has shown interest in our products.

MindFusion Products

MindFusion offers various programming components for different platforms. Below is a list of our products with links to their sections on our website as well link to download a trial version. Clients that have already purchased a product use an increasing discount for each subsequent one.

Diagramming	Links
Diagramming for **WPF**	Product Information Download
Diagramming for **WinForms**	Product Information Download
Diagramming for **Java Swing**	Product Information Download
Diagramming for **JavaScript**	Product Information Download
Diagramming for **ASP.NET**	Product Information Download
Diagramming for **ASP.NET MVC**	Product Information Download
Diagramming for **Xamarin**	Product Information Download
Diagramming for **UWP**	Product Information Download
Diagramming for **Android**	Product Information Download
DiagramViewer for **iOS/Cocoa Touch**	Product Information Download
Diagramming for **ActiveX**	Product Information Download
Diagramming for **Silverlight**	Product Information Download

Charting	Links
Charting for **WPF**	Product Information Download
Charting for **WinForms**	Product Information Download
Charting for **Java Swing**	Product Information Download
Charting for **JavaScript**	Product Information Download

Charting for **ASP.NET**	Product Information Download
Charting for **Xamarin**	Product Information Download
Charting for **Android**	Product Information Download
Charting for **Silverlight**	Product Information Download

Scheduling	Links
Scheduling for **WPF**	Product Information Download
Scheduling for **WinForms**	Product Information Download
Scheduling for **Java Swing**	Product Information Download
Scheduling for **JavaScript**	Product Information Download
Scheduling for **ASP.NET**	Product Information Download
Scheduling for **Xamarin**	Product Information Download
Scheduling for **Silverlight**	Product Information Download

Spreadsheet	Links
Spreadsheet for **WPF**	Product Information Download
Spreadsheet for **WinForms**	Product Information Download
Spreadsheet for **Java Swing**	Product Information Download

Reporting	Links
Reporting for **WPF**	Product Information Download
Reporting for **WinForms**	Product Information Download
Reporting for **Silverlight**	Product Information Download

Virtual Keyboard	Links
Virtual Keyboard for **WPF**	Product Information Download

Virtual Keyboard for **WinForms**	Product Information Download
Virtual Keyboard for **Java Swing**	Product Information Download
Virtual Keyboard for **JavaScript**	Product Information Download

Mapping	Links
Mapping for **WPF**	Product Information Download
Mapping for **WinForms**	Product Information Download
Mapping for **ASP.NET**	Product Information Download
Mapping for **Silverlight**	Product Information Download

Suites	Links
Pack for **WPF**	Product Information Download
Pack for **WinForms**	Product Information Download
Pack for **Java Swing**	Product Information Download
Pack for **JavaScript**	Product Information Download
Pack for **ASP.NET**	Product Information Download
Pack for **Xamarin**	Product Information Download
Pack for **Silverlight**	Product Information Download
Pack for **Diagramming**	Product Information Download
Pack for **Scheduling**	Product Information Download