

JSPortal

Extends

[JSComponent](#)

Property Summary

Number	anchors Enables a component to stick to a specific side of form and/or to grow or shrink when a window is resized.
String	background The background color of the component.
String	borderType The type, color and style of border of the component.
Boolean	enabled The enable state of the component, default true.
String	fontType The font type of the component.
String	foreground The foreground color of the component.
Number	formIndex The Z index of this component.
String	groupID A String representing a group ID for this component.
Number	height The height in pixels of the component.
String	initialSort The default sort order for the rows displayed in the portal.
Boolean	multiLine When set, portal rows can have a custom layout of fields, buttons, etc.
String	name The name of the component.
JSMETHOD	onDrag The method that is triggered when (non Design Mode) dragging occurs.
JSMETHOD	onDragEnd The method that is triggered when (non Design Mode) dragging end occurs.
JSMETHOD	onDragOver The method that is triggered when (non Design Mode) dragging over a component occurs.
JSMETHOD	onDrop The method that is triggered when (non Design Mode) dropping occurs.
JSMETHOD	onRender The method that is executed when the component is rendered.
Number	printSliding Enables an element to resize based on its content and/or move when printing.
Boolean	printable Flag that tells if the component should be printed or not when the form is printed.
String	relationName The name of the relationship between the table related to the currently active form and the table you want to show data from in the portal.
Boolean	reorderable When set, the portal rows can be re-ordered by dragging the column headers.
Boolean	resizable When set the portal rows can be resized by users.
Number	rowHeight The height of each row in pixels.
Number	scrollbars Scrollbar options for the vertical and horizontal scrollbars.
Boolean	showHorizontalLines When set, the portal displays horizontal lines between the rows.
Boolean	showVerticalLines When set the portal displays vertical lines between the columns.
Boolean	sortable When set, users can sort the contents of the portal by clicking on the column headings.
String	styleClass The name of the style class that should be applied to this component.
Number	tabSeq An index that specifies the position of the component in the tab sequence.

Boolean	transparent Flag that tells if the component is transparent or not.
Boolean	visible The visible property of the component, default true.
Number	width The width in pixels of the component.
Number	x The x coordinate of the component on the form.
Number	y The y coordinate of the component on the form.

Method Summary

JSButton	getButton(name) Retrieves a button from the portal based on the name of the button.
JSButton[]	getButtons() Retrieves an array with all buttons in the portal.
Object	getDesignTimeProperty() Get a design-time property of an element.
String[]	getDesignTimePropertyNames() Get the design-time properties of an element.
JSField	getField(name) Retrieves a field from this portal based on the name of the field.
JSField[]	getFields() Retrieves an array with all fields in a portal.
String	getIntercellSpacing() The additional spacing between cell rows.
JSLabel	getLabel(name) Retrieves a label from this portal based on the name of the label.
JSLabel[]	getLabels() Retrieves all labels from the portal.
UUID	getUUID() Returns the UUID of this component.
JSButton	newButton(text, x, width, height, action) Creates a new button on the portal with the given text, place, size and JSMETHOD as the onClick action.
JSField	newCalendar(dataprovider, x, width, height) Creates a new calendar field in the portal.
JSField	newCheck(dataprovider, x, width, height) Creates a new checkbox field in the portal.
JSField	newComboBox(dataprovider, x, width, height) Creates a new combobox field in the portal.
JSField	newField(dataprovider, displaytype, x, width, height) Creates a new field on this form.
JSField	newHtmlArea(dataprovider, x, width, height) Creates a new HTML Area field in the portal.
JSField	newImageMedia(dataprovider, x, width, height) Creates a new Image Media field in the portal.
JSLabel	newLabel(txt, x, width, height) Creates a new label on the form, with the given text, place and size.
JSLabel	newLabel(text, x, width, height, action) Creates a new label on the form, with the given text, place, size and an JSMETHOD as the onClick action.
JSField	newPassword(dataprovider, x, width, height) Creates a new password field in the portal.
JSField	newRadios(dataprovider, x, width, height) Creates a new radio buttons field in the portal.
JSField	newRtfArea(dataprovider, x, width, height) Creates a new RTF Area field in the portal.
JSField	newTextArea(dataprovider, x, width, height) Creates a new text area field in the portal.
JSField	newTextField(dataprovider, x, width, height) Creates a new text field in the portal.
JSField	newTypeAhead(dataprovider, x, width, height) Creates a new type ahead field in the portal.
Object	putDesignTimeProperty() Set a design-time property of an element.
Object	removeDesignTimeProperty() Clear a design-time property of an element.
void	setIntercellSpacing(width, height) The additional spacing between cell rows.

Property Details

anchors

Enables a component to stick to a specific side of form and/or to grow or shrink when a window is resized.

If opposite anchors are activated then the component with grow or shrink with the window. For example if Top and Bottom are activated, then the component will grow/shrink when the window is vertically resized. If Left and Right are activated then the component will grow/shrink when the window is horizontally resized.

If opposite anchors are not activated, then the component will keep a constant distance from the sides of the window which correspond to the activated anchors.

Returns

[Number](#)

Sample

```
var form = solutionModel.newForm('mediaForm', 'db:/example_data/parent_table', null, false, 400, 300);
var stretchAllDirectionsLabel = form.newLabel('Strech all directions', 10, 10, 380, 280);
stretchAllDirectionsLabel.background = 'red';
stretchAllDirectionsLabel.anchors = SM_ANCHOR.ALL;
var stretchVerticallyLabel = form.newLabel('Strech vertically', 10, 10, 190, 280);
stretchVerticallyLabel.background = 'green';
stretchVerticallyLabel.anchors = SM_ANCHOR.WEST | SM_ANCHOR.NORTH | SM_ANCHOR.SOUTH;
var stretchHorizontallyLabel = form.newLabel('Strech horizontally', 10, 10, 380, 140);
stretchHorizontallyLabel.background = 'blue';
stretchHorizontallyLabel.anchors = SM_ANCHOR.NORTH | SM_ANCHOR.WEST | SM_ANCHOR.EAST;
var stickToTopLeftCornerLabel = form.newLabel('Stick to top-left corner', 10, 10, 200, 100);
stickToTopLeftCornerLabel.background = 'orange';
stickToTopLeftCornerLabel.anchors = SM_ANCHOR.NORTH | SM_ANCHOR.WEST; // This is equivalent to SM_ANCHOR.DEFAULT
var stickToBottomRightCornerLabel = form.newLabel('Stick to bottom-right corner', 190, 190, 200, 100);
stickToBottomRightCornerLabel.background = 'pink';
stickToBottomRightCornerLabel.anchors = SM_ANCHOR.SOUTH | SM_ANCHOR.EAST;
```

background

The background color of the component.

Returns

[String](#)

Sample

```
// This property can be used on all types of components.
// Here it is illustrated only for labels and fields.
var greenLabel = form.newLabel('Green',10,10,100,50);
greenLabel.background = 'green'; // Use generic names for colors.
var redField = form.newField('parent_table_text',JSField.TEXT_FIELD,10,110,100,30);
redField.background = '#FF0000'; // Use RGB codes for colors.
```

borderType

The type, color and style of border of the component.

Returns

[String](#)

Sample

```
//HINT: To know exactly the notation of this property set it in the designer and then read it once out through the solution model.
var field = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.borderType = solutionModel.createLineBorder(1,'#ff0000');
```

enabled

The enable state of the component, default true.

Returns[Boolean](#)**Sample**

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.enabled = false;
```

fontType

The font type of the component.

Returns[String](#)**Sample**

```
var label = form.newLabel('Text here', 10, 50, 100, 20);
label.fontType = solutionModel.createFont('Times New Roman',1,14);
```

foreground

The foreground color of the component.

Returns[String](#)**Sample**

```
// This property can be used on all types of components.
// Here it is illustrated only for labels and fields.
var labelWithBlueText = form.newLabel('Blue text', 10, 10, 100, 30);
labelWithBlueText.foreground = 'blue'; // Use generic names for colors.
var fieldWithYellowText = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 50, 100, 20);
fieldWithYellowText.foreground = '#FFFF00'; // Use RGB codes for colors.
```

formIndex

The Z index of this component. If two components overlap, then the component with higher Z index is displayed above the component with lower Z index.

Returns[Number](#)**Sample**

```
var labelBelow = form.newLabel('Green', 10, 10, 100, 50);
labelBelow.background = 'green';
labelBelow.formIndex = 10;
var fieldAbove = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 30);
fieldAbove.background = '#FF0000';
fieldAbove.formIndex = 20;
```

groupID

A String representing a group ID for this component. If several components have the same group ID then they belong to the same group of components. Using the group itself, all components can be disabled/enabled or made invisible/visible.

The group id should be a javascript compatible identifier to allow access of the group in scripting.

Returns[String](#)

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 400, 300);
var label = form.newLabel('Green', 10, 10, 100, 20);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
label.groupID = 'someGroup';
field.groupID = 'someGroup';
forms['someForm'].elements.someGroup.enabled = false;
```

height

The height in pixels of the component.

Returns

Number

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original width: ' + field.width);
application.output('original height: ' + field.height);
field.width = 200;
field.height = 100;
application.output('modified width: ' + field.width);
application.output('modified height: ' + field.height);
```

initialSort

The default sort order for the rows displayed in the portal.

Returns

String

Sample

```
var childrenPortal = form.newPortal('pp',rel,10,10,620,460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text',100,100,20);
childrenPortal.newTextField('child_table_parent_id', 200, 100, 20);
childrenPortal.initialSort = 'child_table_text desc';
```

multiLine

When set, portal rows can have a custom layout of fields, buttons, etc. displayed for each matching row (rather than the default "grid").

Returns

Boolean

Sample

```
var childrenPortal = form.newPortal('pp',rel,10,10,620,460);
// Set the fields some distance apart horizontally. By default this distance
// is ignored and the components are put in a grid.
var idField = childrenPortal.newTextField('child_table_id', 0, 100, 20);
idField.background = 'yellow';
var textField = childrenPortal.newTextField('child_table_text',150,100,20);
textField.background = 'green';
var parentIdField = childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
parentIdField.background = 'orange';
// Disable the grid placing of components, and make the distance between components
// become active.
childrenPortal.multiLine = true;
```

name

The name of the component. Through this name it can also accessed in methods.

Returns

String

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 620, 300);
var label = form.newLabel('Label', 10, 10, 150, 150);
label.name = 'myLabel'; // Give a name to the component.
forms['someForm'].controller.show()
// Now use the name to access the component.
forms['someForm'].elements['myLabel'].text = 'Updated text';
```

onDrag

The method that is triggered when (non Design Mode) dragging occurs.

Returns

[JSMethod](#)

Sample

```
form.onDrag = form.newMethod('function onDrag(event) { application.output("onDrag intercepted from " + event.getSource()); }');
form.onDragEnd = form.newMethod('function onDragEnd(event) { application.output("onDragEnd intercepted from " + event.getSource()); }');
form.onDragOver = form.newMethod('function onDragOver(event) { application.output("onDragOver intercepted from " + event.getSource()); }');
form.onDrop = form.newMethod('function onDrop(event) { application.output("onDrop intercepted from " + event.getSource()); }');
```

onDragEnd

The method that is triggered when (non Design Mode) dragging end occurs.

Returns

[JSMethod](#)

Sample

```
form.onDrag = form.newMethod('function onDrag(event) { application.output("onDrag intercepted from " + event.getSource()); }');
form.onDragEnd = form.newMethod('function onDragEnd(event) { application.output("onDragEnd intercepted from " + event.getSource()); }');
form.onDragOver = form.newMethod('function onDragOver(event) { application.output("onDragOver intercepted from " + event.getSource()); }');
form.onDrop = form.newMethod('function onDrop(event) { application.output("onDrop intercepted from " + event.getSource()); }');
```

onDragOver

The method that is triggered when (non Design Mode) dragging over a component occurs.

Returns

[JSMethod](#)

Sample

```
form.onDrag = form.newMethod('function onDrag(event) { application.output("onDrag intercepted from " + event.getSource()); }');
form.onDragEnd = form.newMethod('function onDragEnd(event) { application.output("onDragEnd intercepted from " + event.getSource()); }');
form.onDragOver = form.newMethod('function onDragOver(event) { application.output("onDragOver intercepted from " + event.getSource()); }');
form.onDrop = form.newMethod('function onDrop(event) { application.output("onDrop intercepted from " + event.getSource()); }');
```

onDrop

The method that is triggered when (non Design Mode) dropping occurs.

Returns

[JSMethod](#)

Sample

```
form.onDrag = form.newMethod('function onDrag(event) { application.output("onDrag intercepted from " + event.getSource()); }');
form.onDragEnd = form.newMethod('function onDragEnd(event) { application.output("onDragEnd intercepted from " + event.getSource()); }');
form.onDragOver = form.newMethod('function onDragOver(event) { application.output("onDragOver intercepted from " + event.getSource()); }');
form.onDrop = form.newMethod('function onDrop(event) { application.output("onDrop intercepted from " + event.getSource()); }');
```

onRender

The method that is executed when the component is rendered.

Returns

[JSMethod](#)

Sample

```
portal.onRender = form.newMethod('function onRender(event) { event.getElement().bgcolor = \'#00ff00\' }');
```

printSliding

Enables an element to resize based on its content and/or move when printing.
The component can move horizontally or vertically and can grow or shrink in height and width, based on its content and the content of neighboring components.

Returns

[Number](#)

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var slidingLabel = form.newLabel('Some long text here', 10, 10, 5, 5);
slidingLabel.printSliding = SM_PRINT_SLIDING.GROW_HEIGHT | SM_PRINT_SLIDING.GROW_WIDTH;
slidingLabel.background = 'gray';
forms['printForm'].controller.showPrintPreview();
```

printable

Flag that tells if the component should be printed or not when the form is printed.

By default components are printable.

Returns

[Boolean](#)

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var printedField = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
var notPrintedField = form.newField('parent_table_id', JSField.TEXT_FIELD, 10, 40, 100, 20);
notPrintedField.printable = false; // This field won't show up in print preview and won't be printed.
forms['printForm'].controller.showPrintPreview();
```

relationName

The name of the relationship between the table related to the currently active form and the table you want to show data from in the portal.

Returns

[String](#)

Sample

```
// Create the portal based on one relation.
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
var idField = childrenPortal.newTextField('child_table_id', 0, 100, 20);
var textField = childrenPortal.newTextField('child_table_text', 150, 100, 20);
var parentIdField = childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
// Now make the portal be based on another relation.
childrenPortal.relationName = 'parent_to_smaller_children';
```

reorderable

When set, the portal rows can be re-ordered by dragging the column headers.

Returns

[Boolean](#)

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text', 150, 100, 20);
childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
childrenPortal.reorderable = true;
```

resizable

When set the portal rows can be resized by users.

Returns

[Boolean](#)

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text', 150, 100, 20);
childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
// Make the columns resizable. By default they are not resizable.
childrenPortal.resizable = true;
```

rowHeight

The height of each row in pixels. If 0 or not set, the height defaults to 10.

Returns

[Number](#)

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text', 150, 100, 20);
childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
childrenPortal.rowHeight = 30;
```

scrollbars

Scrollbar options for the vertical and horizontal scrollbars. Each of the vertical and horizontal scrollbars can be configured to display all the time, to display only when needed or to never display.

Returns

[Number](#)

Sample

```
var noScrollbars = form.newField('my_table_text', JSField.TEXT_AREA, 10, 10, 100, 100);
noScrollbars.scrollbars = SM_SCROLLBAR.HORIZONTAL_SCROLLBAR_NEVER | SM_SCROLLBAR.VERTICAL_SCROLLBAR_NEVER;
var neededScrollbars = form.newField('my_table_text', JSField.TEXT_AREA, 120, 10, 100, 100);
neededScrollbars.scrollbars = SM_SCROLLBAR.HORIZONTAL_SCROLLBAR_AS_NEEDED | SM_SCROLLBAR.
VERTICAL_SCROLLBAR_AS_NEEDED;
var alwaysScrollbars = form.newField('my_table_text', JSField.TEXT_AREA, 230, 10, 100, 100);
alwaysScrollbars.scrollbars = SM_SCROLLBAR.HORIZONTAL_SCROLLBAR_ALWAYS | SM_SCROLLBAR.
VERTICAL_SCROLLBAR_ALWAYS;
```

showHorizontalLines

When set, the portal displays horizontal lines between the rows.

NOTE:

In a multi-line portal, a horizontal line is only displayed in the selected row; to display a horizontal line in all rows, add a line to the portal.

Returns

[Boolean](#)

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text', 150, 100, 20);
childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
childrenPortal.showHorizontalLines = true;
childrenPortal.showVerticalLines = true;
```

showVerticalLines

When set the portal displays vertical lines between the columns.

NOTE:

In a multi-line portal, a vertical line is only displayed in the selected row; to display a vertical line in all rows, add a line to the portal.

Returns

[Boolean](#)

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text', 150, 100, 20);
childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
childrenPortal.showHorizontalLines = true;
childrenPortal.showVerticalLines = true;
```

sortable

When set, users can sort the contents of the portal by clicking on the column headings.

Returns

[Boolean](#)

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text', 150, 100, 20);
childrenPortal.newTextField('child_table_parent_id', 300, 100, 20);
childrenPortal.sortable = true;
```

styleClass

The name of the style class that should be applied to this component.

When defining style classes for specific component types, their names must be prefixed according to the type of the component. For example in order to define a class names 'fancy' for fields, in the style definition the class must be named 'field.fancy'. If it would be intended for labels, then it would be named 'label.fancy'. When specifying the class name for a component, the prefix is dropped however. Thus the field or the label will have its styleClass property set to 'fancy' only.

Returns

[String](#)

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
var style = solutionModel.newStyle('myStyle', 'field.fancy { background-color: yellow; }');
form.styleName = 'myStyle'; // First set the style on the form.
field.styleClass = 'fancy'; // Then set the style class on the field.
```

tabSeq

An index that specifies the position of the component in the tab sequence. The components are put into the tab sequence in increasing order of this property. A value of 0 means to use the default mechanism of building the tab sequence (based on their location on the form). A value of -2 means to remove the component from the tab sequence.

Returns

[Number](#)

Sample

```
// Create three fields. Based on how they are placed, by default they will come one
// after another in the tab sequence.
var fieldOne = form.newField('parent_table_id', JSField.TEXT_FIELD, 10, 10, 100, 20);
var fieldTwo = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
var fieldThree = form.newField('parent_table_id', JSField.TEXT_FIELD, 10, 70, 100, 20);
// Set the third field come before the first in the tab sequence, and remove the
// second field from the tab sequence.
fieldOne.tabSeq = 2;
fieldTwo.tabSeq = SM_DEFAULTS.IGNORE;
fieldThree.tabSeq = 1;
```

transparent

Flag that tells if the component is transparent or not.

The default value is "false", that is the components are not transparent.

Returns

[Boolean](#)

Sample

```
// Load an image from disk and create a Media object based on it.
var imageBytes = plugins.file.readFile('d:/ball.jpg');
var media = solutionModel.newMedia('ball.jpg', imageBytes);
// Put on the form a label with the image.
var image = form.newLabel('', 10, 10, 100, 100);
image.imageMedia = media;
// Put two fields over the image. The second one will be transparent and the
// image will shine through.
var nonTransparentField = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 20, 100, 20);
var transparentField = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 50, 100, 20);
transparentField.transparent = true;
```

visible

The visible property of the component, default true.

Returns

[Boolean](#)

Sample

```
var form = solutionModel.newForm('printForm', 'db:/example_data/parent_table', null, false, 400, 300);
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.visible = false;
```

width

The width in pixels of the component.

Returns

[Number](#)

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original width: ' + field.width);
application.output('original height: ' + field.height);
field.width = 200;
field.height = 100;
application.output('modified width: ' + field.width);
application.output('modified height: ' + field.height);
```

x

The x coordinate of the component on the form.

Returns

[Number](#)

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original location: ' + field.x + ', ' + field.y);
field.x = 90;
field.y = 90;
application.output('changed location: ' + field.x + ', ' + field.y);
```

y

The y coordinate of the component on the form.

Returns

[Number](#)

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
application.output('original location: ' + field.x + ', ' + field.y);
field.x = 90;
field.y = 90;
application.output('changed location: ' + field.x + ', ' + field.y);
```

Method Details**getButton**

[JSButton](#) **getButton** (name)

Retrieves a button from the portal based on the name of the button.

Parameters

{[String](#)} name - The name of the button to retrieve.

Returns

[JSButton](#) - A JSButton instance that corresponds to the button with the specified name.

Sample

```
var clickMethod = form.newMethod('function clickMe() { application.output("I was clicked!"); }');
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
var btn = childrenPortal.newButton('Click me!', 400, 100, 20, clickMethod);
btn.name = 'clickMeBtn'; // Give a name to the button, so we can retrieve it by name later.
// Retrieve the button by name and do something with it.
var btnBack = childrenPortal.getButton('clickMeBtn');
btnBack.background = 'yellow';
// Retrieve the button through the array of all buttons and do something with it.
var allButtons = childrenPortal.getButtons();
var btnBackAgain = allButtons[0];
btnBackAgain.foreground = 'red';
```

getButtons**JSButton[]** **getButtons ()**

Retrieves an array with all buttons in the portal.

Returns**JSButton[]** - An array with all buttons in the portal.**Sample**

```
var clickMethod = form.newMethod('function clickMe() { application.output("I was clicked!"); }');
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
var btn = childrenPortal.newButton('Click me!', 400, 100, 20, clickMethod);
btn.name = 'clickMeBtn'; // Give a name to the button, so we can retrieve it by name later.
// Retrieve the button by name and do something with it.
var btnBack = childrenPortal.getButton('clickMeBtn');
btnBack.background = 'yellow';
// Retrieve the button through the array of all buttons and do something with it.
var allButtons = childrenPortal.getButtons();
var btnBackAgain = allButtons[0];
btnBackAgain.foreground = 'red';
```

getDesignTimeProperty**Object** **getDesignTimeProperty ()**

Get a design-time property of an element.

Returns**Object****Sample**

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
var prop = fld.getDesignTimeProperty('myprop')
```

getDesignTimePropertyNames**String[]** **getDesignTimePropertyNames ()**

Get the design-time properties of an element.

Returns**String[]****Sample**

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
var propName = fld.getDesignTimePropertyNames()
```

getField**JSField** **getField (name)**

Retrieves a field from this portal based on the name of the field.

Parameters

{String} name - The name of the field to retrieve.

Returns

JSField - A JSField instance corresponding to the field with the specified name.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var cal = childrenPortal.newField('my_table_date', JSField.CALENDAR, 0, 60, 20);
var tfield = childrenPortal.newField('my_table_text', JSField.TEXT_FIELD, 60, 60, 20);
tfield.name = 'textField'; // Give a name to the field so we can retrieve it later by name.
// Retrieve the text field by its name and do something with it.
var textFieldBack = childrenPortal.getField('textField');
textFieldBack.background = 'yellow';
// Retrieve the calendar field through the array of all fields and do something with it.
var allFields = childrenPortal.getFields();
var calFieldBack = allFields[0];
calFieldBack.foreground = 'red';
```

getFields

JSField[] getFields ()

Retrieves an array with all fields in a portal.

Returns

JSField[] - An array with JSField instances corresponding to all fields in the portal.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var cal = childrenPortal.newField('my_table_date', JSField.CALENDAR, 0, 60, 20);
var tfield = childrenPortal.newField('my_table_text', JSField.TEXT_FIELD, 60, 60, 20);
tfield.name = 'textField'; // Give a name to the field so we can retrieve it later by name.
// Retrieve the text field by its name and do something with it.
var textFieldBack = childrenPortal.getField('textField');
textFieldBack.background = 'yellow';
// Retrieve the calendar field through the array of all fields and do something with it.
var allFields = childrenPortal.getFields();
var calFieldBack = allFields[0];
calFieldBack.foreground = 'red';
```

getInterCellSpacing

String getInterCellSpacing ()

The additional spacing between cell rows. Is composed from the horizontal spacing and the vertical spacing.

Returns

String - A java.awt.Dimension object holding the horizontal and vertical intercell spacing.

Sample

```
var spacing = childrenPortal.getInterCellSpacing();
application.output("horizontal spacing: " + spacing.width);
application.output("vertical spacing: " + spacing.height);
```

getLabel

JSLabel getLabel (name)

Retrieves a label from this portal based on the name of the label.

Parameters

{String} name - The name of the label to retrieve.

Returns

JSLabel - A JSLabel instance corresponding to the label with the specified name.

Sample

```
var callLabel = childrenPortal.newLabel('Date', 120, 60, 20);
var textLabel = childrenPortal.newLabel('Text', 180, 60, 20);
textLabel.name = 'textLabel'; // Give a name to this label, so we can retrieve it by name.
// Retrieve the second label by name.
var textLabelBack = childrenPortal.getLabel('textLabel');
textLabelBack.background = 'yellow';
// Retrieve the first label through the array of all labels.
var allLabels = childrenPortal.getLabels();
var callLabelBack = allLabels[0];
callLabelBack.foreground = 'red';
```

getLabels**JSLabel[]** getLabels ()

Retrieves all labels from the portal.

Returns**JSLabel[]** - An array of JSLabel instances corresponding to all labels in the portal.**Sample**

```
var callLabel = childrenPortal.newLabel('Date', 120, 60, 20);
var textLabel = childrenPortal.newLabel('Text', 180, 60, 20);
textLabel.name = 'textLabel'; // Give a name to this label, so we can retrieve it by name.
// Retrieve the second label by name.
var textLabelBack = childrenPortal.getLabel('textLabel');
textLabelBack.background = 'yellow';
// Retrieve the first label through the array of all labels.
var allLabels = childrenPortal.getLabels();
var callLabelBack = allLabels[0];
callLabelBack.foreground = 'red';
```

getUUID**UUID** getUUID ()

Returns the UUID of this component.

Returns**UUID****Sample**

```
var button_uuid = solutionModel.getForm("my_form").getButton("my_button").getUUID();
application.output(button_uuid.toString());
```

newButton**JSButton** newButton (text, x, width, height, action)

Creates a new button on the portal with the given text, place, size and JSMethod as the onClick action.

Parameters{**String**} text - The text to be displayed on the button.{**Number**} x - The x coordinate of the button. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.{**Number**} width - The width of the button.{**Number**} height - The height of the button. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.{**Object**} action - The JSMethod object that should be executed when the button is clicked.**Returns****JSButton** - A JSButton instance representing the newly created button.**Sample**

```
var clickMethod = form.newMethod('function clickMe() { application.output("I was clicked!"); }');
var childrenPortal = form.newPortal('pp', 'parent_to_child', 10, 10, 620, 460);
childrenPortal.newButton('Click me!', 400, 100, 20, clickMethod);
```

newCalendar

JSField **newCalendar** (dataprovder, x, width, height)

Creates a new calendar field in the portal. It is equivalent to calling "newField" with the type JSField.CALENDAR.

Parameters

{Object} dataprovder - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created calendar.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var cal = childrenPortal.newCalendar('my_table_date', 0, 60, 20);
```

newCheck

JSField **newCheck** (dataprovder, x, width, height)

Creates a new checkbox field in the portal. It is equivalent to calling "newField" with the type JSField.CHECKS.

Parameters

{Object} dataprovder - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created checkbox field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var chk = childrenPortal.newCheck('my_table_options', 60, 60, 50);
chk.valuelist = vlist;
```

newComboBox

JSField **newComboBox** (dataprovder, x, width, height)

Creates a new combobox field in the portal. It is equivalent to calling "newField" with the type JSField.COMBOBOX.

Parameters

{Object} dataprovder - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created combobox field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var cmb = childrenPortal.newComboBox('my_table_options', 120, 160, 20);
cmb.valuelist = vlist;
```

newField

JSField **newField** (dataprovder, displaytype, x, width, height)

Creates a new field on this form. The type of the field is specified by using one of the JSField constants like JSField.TEXT_FIELD.

Parameters

{Object} dataprovder - The data provider for this field. Can be either a column name, or an instance of JSVariable.

{Number} displaytype - The display type of the field. Use constants from JSField for this parameter.

{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.

{Number} width - The width of the field.

{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);

var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";

var cal = childrenPortal.newField('my_table_date', JSField.CALENDAR, 0, 60, 20);
var chk = childrenPortal.newField('my_table_options', JSField.CHECKS, 60, 60, 50);
chk.valuelist = vlist;
var cmb = childrenPortal.newField('my_table_options', JSField.COMBOBOX, 120, 160, 20);
cmb.valuelist = vlist;
var html = childrenPortal.newField('my_table_html', JSField.HTML_AREA, 180, 60, 50);
var img = childrenPortal.newField('my_table_image', JSField.IMAGE_MEDIA, 240, 60, 50);
var pwd = childrenPortal.newField('my_table_text', JSField.PASSWORD, 300, 60, 20);
var radio = childrenPortal.newField('my_table_options', JSField.RADIOS, 360, 60, 50);
radio.valuelist = vlist;
var rtf = childrenPortal.newField('my_table_rtf', JSField.RTF_AREA, 420, 60, 50);
var tarea = childrenPortal.newField('my_table_text', JSField.TEXT_AREA, 480, 60, 50);
var tfield = childrenPortal.newField('my_table_text', JSField.TEXT_FIELD, 540, 60, 20);
var tahead = childrenPortal.newField('my_table_text', JSField.TYPE_AHEAD, 600, 60, 20);
tahead.valuelist = vlist;
```

newHtmlArea

JSField **newHtmlArea** (dataprovder, x, width, height)

Creates a new HTML Area field in the portal. It is equivalent to calling "newField" with the type JSField.HTML_AREA.

Parameters

{Object} dataprovder - The data provider for this field. Can be either a column name, or an instance of JSVariable.

{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.

{Number} width - The width of the field.

{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created HTML Area field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var html = childrenPortal.newHtmlArea('my_table_html', 180, 60, 50);
```

newImageMedia

JSField **newImageMedia** (dataprovder, x, width, height)

Creates a new Image Media field in the portal. It is equivalent to calling "newField" with the type JSField.IMAGE_MEDIA.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created Image Media field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var img = childrenPortal.newImageMedia('my_table_image', 240, 60, 50);
```

newLabel

JSLabel newLabel (txt, x, width, height)

Creates a new label on the form, with the given text, place and size.

Parameters

{String} txt - The text that will be displayed in the label.
{Number} x - The x coordinate of the label. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the label.
{Number} height - The height of the label. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSLabel - A JSLabel instance that represents the newly created label.

Sample

```
var clickMethod = form.newMethod('function clickMe() { application.output("I was clicked!"); }');
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var callLabel = childrenPortal.newLabel('Date', 120, 60, 20);
// This will result in a button being actually created, because we specify an action.
var textLabel = childrenPortal.newLabel('Text', 180, 60, 20, clickMethod);
```

newLabel

JSLabel newLabel (text, x, width, height, action)

Creates a new label on the form, with the given text, place, size and an JSMETHOD as the onClick action.

Parameters

{String} text - The text that will be displayed in the label.
{Number} x - The x coordinate of the label. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the label.
{Number} height - The height of the label. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.
{Object} action - The JSMETHOD object that should be executed when the label is clicked.

Returns

JSLabel - A JSLabel instance that represents the newly created label.

Sample

```
var clickMethod = form.newMethod('function clickMe() { application.output("I was clicked!"); }');
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var callLabel = childrenPortal.newLabel('Date', 120, 60, 20);
// This will result in a button being actually created, because we specify an action.
var textLabel = childrenPortal.newLabel('Text', 180, 60, 20, clickMethod);
```

newPassword

JSField newPassword (dataprovider, x, width, height)

Creates a new password field in the portal. It is equivalent to calling "newField" with the type JSField.PASSWORD.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created password field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var pwd = childrenPortal.newPassword('my_table_text', 300, 60, 20);
```

newRadios

JSField newRadios (dataprovider, x, width, height)

Creates a new radio buttons field in the portal. It is equivalent to calling "newField" with the type JSField.RADIOS.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created radio buttons.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var radio = childrenPortal.newRadios('my_table_options', 360, 60, 50);
radio.valuelist = vlist;
```

newRtfArea

JSField newRtfArea (dataprovider, x, width, height)

Creates a new RTF Area field in the portal. It is equivalent to calling "newField" with the type JSField.RTF_AREA.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created RTF Area field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var rtf = childrenPortal.newRtfArea('my_table_rtf', 420, 60, 50);
```

newTextArea

JSField newTextArea (dataprovider, x, width, height)

Creates a new text area field in the portal. It is equivalent to calling "newField" with the type JSField.TEXT_AREA.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created text area field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var tarea = childrenPortal.newTextArea('my_table_text', 480, 60, 50);
```

newTextField

JSField newTextField (dataprovider, x, width, height)

Creates a new text field in the portal. It is equivalent to calling "newField" with the type JSField.TEXT_FIELD.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created text field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var tfield = childrenPortal.newTextField('my_table_text', 540, 60, 20);
```

newTypeAhead

JSField newTypeAhead (dataprovider, x, width, height)

Creates a new type ahead field in the portal. It is equivalent to calling "newField" with the type JSField.TYPE_AHEAD.

Parameters

{Object} dataprovider - The data provider for this field. Can be either a column name, or an instance of JSVariable.
{Number} x - The x coordinate of the field. If the portal does not have the "multiLine" property set, then the x coordinates are used only for determining the order of the columns in the grid. If the portal has the "multiLine" property set, then the components are actually displayed at the specified coordinates.
{Number} width - The width of the field.
{Number} height - The height of the field. In a portal the height of all components is set to the height of the first component, unless the "multiLine" property is set.

Returns

JSField - A JSField instance that corresponds to the newly created type ahead field.

Sample

```
var childrenPortal = form.newPortal('pp', 'parent_to_my_table', 10, 10, 1180, 780);
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var tahead = childrenPortal.newTypeAhead('my_table_text', 600, 60, 20);
tahead.valueList = vlist;
```

putDesignTimeProperty

Object putDesignTimeProperty ()

Set a design-time property of an element.

Returns

Object

Sample

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
fld.putDesignTimeProperty('myprop', 'strawberry')
```

removeDesignTimeProperty

Object **removeDesignTimeProperty** ()

Clear a design-time property of an element.

Returns

Object

Sample

```
var frm = solutionModel.getForm('orders')
var fld = frm.getField('fld')
fld.removeDesignTimeProperty('myprop')
```

setIntercellSpacing

void **setIntercellSpacing** (width, height)

The additional spacing between cell rows. Is composed from the horizontal spacing and the vertical spacing.

Parameters

{**Number**} width - The horizontal spacing between cells.

{**Number**} height - The vertical spacing between cells.

Returns

void

Sample

```
var childrenPortal = form.newPortal('pp',rel,10,10,620,460);
childrenPortal.newTextField('child_table_id', 0, 100, 20);
childrenPortal.newTextField('child_table_text',100,100,20);
childrenPortal.newTextField('child_table_parent_id', 200, 100, 20);
childrenPortal.setIntercellSpacing(5,10);
```