JSField

Extends

JSComponent

Constants Summary

Number	CALENDAR Constant for specifying the display type of a JSField.
Number	CHECKS Constant for specifying the display type of a JSField.
Number	COMBOBOX Constant for specifying the display type of a JSField.
Number	HTML_AREA Constant for specifying the display type of a JSField.
Number	IMAGE_MEDIA Constant for specifying the display type of a JSField.
Number	LISTBOX Constant for specifying the display type of a JSField.
Number	MULTISELECT_LISTBOX Constant for specifying the display type of a JSField.
Number	PASSWORD Constant for specifying the display type of a JSField.
Number	RADIOS Constant for specifying the display type of a JSField.
Number	RTF_AREA Constant for specifying the display type of a JSField.
Number	SPINNER Constant for specifying the display type of a JSField.
Number	TEXT_AREA Constant for specifying the display type of a JSField.
Number	TEXT_FIELD Constant for specifying the display type of a JSField.
Number	TYPE_AHEAD Constant for specifying the display type of a JSField.

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.
String	dataProviderID The dataprovider of the component.
Number	displayType The type of display used by the field.
Boolean	displaysTags Flag that enables or disables merging of data inside components using tags (placeholders).
Boolean	editable Flag that tells if the content of the field can be edited or not.
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	format The format that should be applied when displaying the data in the component.
String	groupID A String representing a group ID for this component.

Number	height The height in pixels of the component.
Number	horizontalAlignment Horizontal alignment of the text inside the component.
String	margin The margins of the component.
String	name The name of the component.
JSMethod	onAction The method that is executed when the component is clicked.
JSMethod	onDataChange Method that is executed when the data in the component is successfully changed.
JSMethod	onFocusGained The method that is executed when the component gains focus.
JSMethod	onFocusLost The method that is executed when the component looses focus.
JSMethod	onRender The method that is executed when the component is rendered.
JSMethod	onRightClick The method that is executed when the component is right clicked.
String	placeholderText The text that is displayed in field when the field doesn't have a text value.
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.
Number	scrollbars Scrollbar options for the vertical and horizontal scrollbars.
Boolean	selectOnEnter Flag that tells if the content of the field should be automatically selected when the field receives focus.
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.
String	titleText The text that is displayed in the column header associated with the component when the form is in table view.
String	toolTipText The text displayed when hovering over the component with a mouse cursor.
Boolean	transparent Flag that tells if the component is transparent or not.
JSValueList	valuelist The valuelist that is used by this field when displaying data.
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

Object	getDesignTimeProperty() Get a design-time property of an element.
String[]	getDesignTimePropertyNames() Get the design-time properties of an element.
UUID	getUUID() Returns the UUID of this component.
Object	putDesignTimeProperty() Set a design-time property of an element.
Object	removeDesignTimeProperty() Clear a design-time property of an element.

Constants Details

CALENDAR

Constant for specifying the display type of a JSField. Sets the display type of the field to calendar. The field will show a formatted date and will have a button which pops up a calendar for date selection.

Returns

Number

Sample

```
var cal = form.newField('my_table_date', JSField.CALENDAR, 10, 10, 100, 20);
```

CHECKS

Constant for specifying the display type of a JSField. Sets the display type of the field to checkbox. The field will show a checkbox, or a list of checkboxes if the valuelist property is also set.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var chk = form.newField('my_table_options', JSField.CHECKS, 10, 40, 100, 50);
chk.valuelist = vlist;
```

COMBOBOX

Constant for specifying the display type of a JSField. Sets the display type of the field to combobox.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var cmb = form.newField('my_table_options', JSField.COMBOBOX, 10, 100, 100, 20);
cmb.valuelist = vlist;
```

HTML_AREA

Constant for specifying the display type of a JSField. Sets the display type of the field to HTML area. The field will display formatted HTML content.

Returns

Number

Sample

```
var html = form.newField('my_table_html', JSField.HTML_AREA, 10, 130, 100, 50);
```

IMAGE_MEDIA

Constant for specifying the display type of a JSField. Sets the display type of the field to image. The field will display images.

Returns

Number

Sample

```
var img = form.newField('my_table_image', JSField.IMAGE_MEDIA, 10, 190, 100, 50);
```

LISTBOX

Constant for specifying the display type of a JSField. Sets the display type of the field to list box. The field will show a selection list with single choice selection.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var list = form.newField('my_table_list', JSField.LISTBOX, 10, 280, 100, 50);
list.valuelist = vlist;
```

MULTISELECT_LISTBOX

Constant for specifying the display type of a JSField. Sets the display type of the field to list box. The field will show a selection list with multiple choice selection.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var list = form.newField('my_table_options', JSField.MULTISELECT_LISTBOX, 10, 280, 100, 50);
list.valuelist = vlist;
```

PASSWORD

Constant for specifying the display type of a JSField. Sets the display type of the fiels to password. The field will allow the user to enter passwords, masking the typed characters.

Returns

Number

Sample

```
var pwd = form.newField('my_table_text', JSField.PASSWORD, 10, 250, 100, 20);
```

RADIOS

Constant for specifying the display type of a JSField. Sets the display type of the field to radio buttons. The field will show a radio button, or a list of them if the valuelist property is also set.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var radio = form.newField('my_table_options', JSField.RADIOS, 10, 280, 100, 50);
radio.valuelist = vlist;
```

RTF_AREA

Constant for specifying the display type of a JSField. Sets the display type of the field to RTF area. The field will display formatted RTF content.

Returns

Number

Sample

```
var rtf = form.newField('my_table_rtf', JSField.RTF_AREA, 10, 340, 100, 50);
```

SPINNER

Constant for specifying the display type of a JSField. Sets the display type of the field to spinner. The field will show a spinner.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var spinner = form.newField('my_spinner', JSField.SPINNER, 10, 460, 100, 20);
spinner.valuelist = vlist;
```

TEXT_AREA

Constant for specifying the display type of a JSField. Sets the display type of the field to text area. The field will show text on multiple lines.

Returns

Number

Sample

```
var tarea = form.newField('my_table_text', JSField.TEXT_AREA, 10, 400, 100, 50);
```

TEXT_FIELD

Constant for specifying the display type of a JSField. Sets the display type of the field to text field. The field will show regular text on a single line.

Returns

Number

Sample

```
var tfield = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 460, 100, 20);
```

TYPE_AHEAD

Constant for specifying the display type of a JSField. Sets the display type of the field to type ahead. The field will show regular text, but will have type ahead capabilities.

Returns

Number

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var tahead = form.newField('my_table_text', JSField.TYPE_AHEAD, 10, 490, 100, 20);
tahead.valuelist = vlist;
```

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 strechAllDirectionsLabel = form.newLabel('Strech all directions', 10, 10, 380, 280);
strechAllDirectionsLabel.background = 'red';
strechAllDirectionsLabel.anchors = SM_ANCHOR.ALL;
var strechVerticallyLabel = form.newLabel('Strech vertically', 10, 10, 190, 280);
strechVerticallyLabel.background = 'green';
strechVerticallyLabel.anchors = SM_ANCHOR.WEST | SM_ANCHOR.NORTH | SM_ANCHOR.SOUTH;
var strechHorizontallyLabel = form.newLabel('Strech horizontally', 10, 10, 380, 140);
strechHorizontallyLabel.background = 'blue';
strechHorizontallyLabel.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');
```

dataProviderID

The dataprovider of the component.

Returns

String

Sample

```
// Normally the dataprovider is specified when a component is created.
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
// But it can be modified later if needed.
field.dataProviderID = 'parent_table_id';
```

displayType

The type of display used by the field. Can be one of CALENDAR, CHECKS, COMBOBOX, HTML_AREA, IMAGE_MEDIA, PASSWORD, RADIOS, RTF_AREA, TEXT_AREA, TEXT_FIELD, TYPE_AHEAD, LIST_BOX, MULTISELECT_LISTBOX or SPINNER.

Returns

Number

Sample

```
// The display type is specified when the field is created.
var cal = form.newField('my_table_date', JSField.CALENDAR, 10, 10, 100, 20);
// But it can be changed if needed.
cal.dataProviderID = 'my_table_text';
cal.displayType = JSField.TEXT_FIELD;
```

displaysTags

Flag that enables or disables merging of data inside components using tags (placeholders). Tags (or placeholders) are words surrounded by %%on each side. There are data tags and standard tags. Data tags consist in names of dataproviders surrounded by%%. Standard tags are a set of predefined tags that are made available by the system.

See the "Merging data" section for more details about tags.

The default value of this flag is "false", that is merging of data is disabled by default.

Returns

Boolean

Sample

```
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
field.displaysTags = true;
```

editable

Flag that tells if the content of the field can be edited or not.
The default value of this flag is "true", that is the content can be edited.

Returns

Boolean

Sample

```
var field = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.editable = false;
```

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;
```

format

The format that should be applied when displaying the data in the component.
There are different options for the different dataprovider types that are assigned to this field.
For Integer fields, there is a display and an edit format, using http://docs.oracle.com/javase/7/docs/api/java/text/DecimalFormat.html and the max (string) length can be set.
For Text/String fields, there are options to force uppercase,lowercase or only numbers. Or a mask can be set that restrict the input the pattern chars can be found here: http://docs.oracle.com/javase/7/docs/api/javax/swing/text/MaskFormatter.html
A mask can have a placehoder (what is shown when there is no data) and if the data must be stored raw (without literals of the mask). A max text length can also be set to force
the max text length input, this doesn't work on mask because that max length is controlled with the mask.
For Date fields a display and edit format can be set by using a pattern from here: http://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html, you can also say this must behave like a mask (the edit format)
A mask only works with when the edit format is exactly that mask (1 char is 1 number/char), because for example MM then only 2 numbers are allowed MMM that displays the month as a string is not supported as a mask.
Some examples are "dd-MM-yyyy", "MM-dd-yyyy", etc.
The format property is also used to set the UI Converter, this means that you can convert the value object to something else before it gets set into the field, this can also result in a type change of the data.
So a string in scripting/db is converted to a integer in the ui, then you have to set an integer format.
This property is applicable only for types: TEXT_FIELD, COMBOBOX, TYPE_AHEAD, CALENDAR and SPINNER.

Returns

String

Sample

```
var field = form.newField('my_table_number', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.format = '$#.00';
```

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);
```

horizontalAlignment

Horizontal alignment of the text inside the component. Can be one of LEFT, CENTER or RIGHT.

Note that this property does not refer to the horizontal alignment of the component inside the form.

Returns

Number

Sample

```
var leftAlignedLabel = form.newLabel('LEFT', 10, 10, 300, 20);
leftAlignedLabel.horizontalAlignment = SM_ALIGNMENT.LEFT;
var hCenteredLabel = form.newLabel('CENTER', 10, 40, 300, 20);
hCenteredLabel.horizontalAlignment = SM_ALIGNMENT.CENTER;
var rightAlignedLabel = form.newLabel('RIGHT', 10, 70, 300, 20);
rightAlignedLabel.horizontalAlignment = SM_ALIGNMENT.RIGHT;
```

margin

The margins of the component. They are specified in this order, separated by commas: top, left, bottom, right.

Returns

String

Sample

```
var label = form.newLabel('Label', 10, 10, 150, 150);
label.background = 'yellow';
label.margin = '10,20,30,40';
```

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';
```

onAction

The method that is executed when the component is clicked.

Returns

JSMethod

Sample

```
var doNothingMethod = form.newMethod('function doNothing() { application.output("Doing nothing."); }');
var onClickMethod = form.newMethod('function onClick(event) { application.output("I was clicked at " + event.
getTimestamp()); }');
var onDoubleClickMethod = form.newMethod('function onDoubleClick(event) { application.output("I was double-
clicked at " + event.getTimestamp()); }');
var onRightClickMethod = form.newMethod('function onRightClick(event) { application.output("I was right-
clicked at " + event.getTimestamp()); }');
// At creation the button has the 'doNothing' method as onClick handler, but we'll change that later.
var btn = form.newButton('I am a button', 10, 40, 200, 20, doNothingMethod);
btn.onAction = onClickMethod;
btn.onDoubleClick = onDoubleClickMethod;
btn.onRightClick = onRightClickMethod;
```

onDataChange

Method that is executed when the data in the component is successfully changed.

Returns

JSMethod

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 620, 300);
var onDataChangeMethod = form.newMethod('function onDataChange(oldValue, newValue, event) { application.
output("Data changed from " + oldValue + " to " + newValue + " at " + event.getTimestamp()); }');
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.onDataChange = onDataChangeMethod;
forms['someForm'].controller.show();
```

onFocusGained

The method that is executed when the component gains focus.
NOTE: Do not call methods that will influence the focus itself.

Returns

JSMethod

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 620, 300);
var onFocusLostMethod = form.newMethod('function onFocusLost(event) { application.output("Focus lost at " +
event.getTimestamp()); }');
var onFocusGainedMethod = form.newMethod('function onFocusGained(event) { application.output("Focus gained
at " + event.getTimestamp()); }');
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.onFocusGained = onFocusGainedMethod;
field.onFocusLost = onFocusLostMethod;
forms['someForm'].controller.show()
```

onFocusLost

The method that is executed when the component looses focus.

Returns

JSMethod

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table', null, false, 620, 300);
var onFocusLostMethod = form.newMethod('function onFocusLost(event) { application.output("Focus lost at " +
event.getTimestamp()); }');
var onFocusGainedMethod = form.newMethod('function onFocusGained(event) { application.output("Focus gained
at " + event.getTimestamp()); }');
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.onFocusGained = onFocusGainedMethod;
field.onFocusLost = onFocusLostMethod;
forms['someForm'].controller.show()
```

onRender

The method that is executed when the component is rendered.

Returns

JSMethod

Sample

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

onRightClick

The method that is executed when the component is right clicked.

Returns

JSMethod

Sample

```
var doNothingMethod = form.newMethod('function doNothing() { application.output("Doing nothing."); }');
var onClickMethod = form.newMethod('function onClick(event) { application.output("I was clicked at " + event.
getTimestamp()); }');
var onDoubleClickMethod = form.newMethod('function onDoubleClick(event) { application.output("I was double-
clicked at " + event.getTimestamp()); }');
var onRightClickMethod = form.newMethod('function onRightClick(event) { application.output("I was right-
clicked at " + event.getTimestamp()); }');
// At creation the button has the 'doNothing' method as onClick handler, but we'll change that later.
var btn = form.newButton('I am a button', 10, 40, 200, 20, doNothingMethod);
btn.onAction = onClickMethod;
btn.onDoubleClick = onDoubleClickMethod;
btn.onRightClick = onRightClickMethod;
```

placeholderText

The text that is displayed in field when the field doesn't have a text value.

Returns

String

Sample

```
field.placeholderText = 'Search';
```

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()
```

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;
```

selectOnEnter

Flag that tells if the content of the field should be automatically selected when the field receives focus. The default value of this field is "false".

Returns

Boolean

Sample

```
// Create two fields and set one of them to have "selectOnEnter" true. As you tab
// through the fields you can notice how the text inside the second field gets
// automatically selected when the field receives focus.
var fieldNoSelect = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
var fieldSelect = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 40, 100, 20);
fieldSelect.selectOnEnter = 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;
```

titleText

The text that is displayed in the column header associated with the component when the form is in table view.

Returns

String

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/my_table', null, false, 640, 480);
var field = form.newField('my_table_number', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.titleText = 'Column Title';
form.view = JSForm.LOCKED_TABLE_VIEW;
forms['someForm'].controller.show()
```

toolTipText

The text displayed when hovering over the component with a mouse cursor.

NOTE:
HTML should be used for multi-line tooltips; you can also use any valid HTML tags to format tooltip text. For example:
<html>This includesbolded text and
BLUE text as well.</html>

Returns

String

Sample

```
var label = form.newLabel('Stop the mouse over me!', 10, 10, 200, 20);
label.toolTipText = 'I\'m the tooltip. Do you see me?';
```

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 an 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;
```

valuelist

The valuelist that is used by this field when displaying data. Can be used with fields of type CHECKS, COMBOBOX, RADIOS and TYPE_AHEAD.

Returns

JSValueList

Sample

```
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "one\ntwo\nthree\nfour";
var cmb = form.newField('my_table_options', JSField.COMBOBOX, 10, 100, 100, 20);
cmb.valuelist = vlist;
```

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

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 propNames = fld.getDesignTimePropertyNames()
```

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());
```

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')
```