

# JSTabPanel

## Property Summery

Number	<a href="#">#anchors</a> Enables a component to stick to a specific side of form and/or to grow or shrink when a window is resized.
String	<a href="#">#background</a> The background color of the component.
String	<a href="#">#borderType</a> The type, color and style of border of the component.
String	<a href="#">#fontType</a> The font type of the component.
String	<a href="#">#foreground</a> The foreground color of the component.
Number	<a href="#">#formIndex</a> The Z index of this component.
String	<a href="#">#groupID</a> A String representing a group ID for this component.
Number	<a href="#">#height</a> The height in pixels of the component.
String	<a href="#">#name</a> The name of the component.
JSMETHOD	<a href="#">#onTabChange</a> Method to be executed when the selected tab is changed in the tab panel.
Number	<a href="#">#printSliding</a> Enables an element to resize based on its content and/or move when printing.
Boolean	<a href="#">#printable</a> Flag that tells if the component should be printed or not when the form is printed.
Boolean	<a href="#">#scrollTabs</a> Flag that tells how to arrange the tabs if they don't fit on a single line.
String	<a href="#">#styleClass</a> The name of the style class that should be applied to this component.
Number	<a href="#">#tabOrientation</a> The position of the tabs related to the tab panel.
Number	<a href="#">#tabSeq</a> An index that specifies the position of the component in the tab sequence.
Boolean	<a href="#">#transparent</a> Flag that tells if the component is transparent or not.
Number	<a href="#">#width</a> The width in pixels of the component.
Number	<a href="#">#x</a> The x coordinate of the component on the form.
Number	<a href="#">#y</a> The y coordinate of the component on the form.

## Method Summery

JSTab	<a href="#">#getTab(name)</a> Returns a JSTab instance representing the tab which has the specified name.
JSTab[]	<a href="#">#getTabs()</a> Returns an array of JSTab instances holding the tabs of the tab panel.
JSTab	<a href="#">#newTab(name, text, form)</a>
JSTab	<a href="#">#newTab(name, text, form, [relation])</a> Adds a new tab with the text label and JSForm and JSRelation (can be null for unrelated).

## 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', '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 = 'LineBorder,2,#FF0000';
```

#### fontType

The font type 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 label = form.newLabel('Text here', 10, 50, 100, 20);
label.fontType = '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.

#### Returns

[String](#)

#### Sample

```
var form = solutionModel.newForm('someForm', '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);
```

name

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

**Returns**

String

**Sample**

```
var form = solutionModel.newForm('someForm', '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';
```

onTabChange

Method to be executed when the selected tab is changed in the tab panel.

**Returns**

JSMMethod

**Sample**

```
var onTabChangeMethod = form.newFormMethod('function onTabChange(previousIndex, event) { application.output
("Tab changed from previous index " + previousIndex + " at " + event.getTimestamp()); }');
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.onTabChange = onTabChangeMethod;
```

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', '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', '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()
```

#### scrollTabs

Flag that tells how to arrange the tabs if they don't fit on a single line.

If this flag is set, then the tabs will stay on a single line, but there will be the possibility to scroll them to the left and to the right. If this flag is not set, then the tabs will be arranged on multiple lines.

#### Returns

Boolean

#### Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 200, 200);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.scrollTabs = 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', '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.
```

#### tabOrientation

The position of the tabs related to the tab panel. Can be one of TOP, RIGHT, BOTTOM, LEFT, HIDE, SPLIT\_HORIZONTAL, SPLIT\_VERTICAL. The HIDE option makes the tabs invisible, SPLIT\_HORIZONTAL makes the tab panel horizontal split pane, SPLIT\_VERTICAL makes the tab panel vertical split pane.

#### Returns

Number

#### Sample

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
// The SM_ALIGNMENT constants TOP, RIGHT, BOTTOM and LEFT can be used to put the
// tabs into the needed position. Use SM_DEFAULTS.NONE to hide the tabs.
// The SM_ALIGNMENT constants SPLIT_HORIZONTAL, SPLIT_VERTICAL can be used to create a split pane
// where the first tab will be left component and the second tab will be the right component.
tabPanel.tabOrientation = SM_ALIGNMENT.BOTTOM;
```

#### 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;
```

#### 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**

getTab

**JSTab** getTab(name)

Returns a JSTab instance representing the tab which has the specified name.

**Parameters**

{String} name – The name of the tab that should be returned.

**Returns**

**JSTab** – A JSTab instance represented the requested tab.

**Sample**

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
tabPanel.getTab('tab2').text = 'Child Two Changed';
```

getTabs

**JSTab[]** getTabs()

Returns an array of JSTab instances holding the tabs of the tab panel.

**Returns**

**JSTab[]** – An array of JSTab instances representing all tabs of this tabpanel.

**Sample**

```
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne);
tabPanel.newTab('tab2', 'Child Two', childTwo);
var tabs = tabPanel.getTabs();
for (var i=0; i<tabs.length; i++)
    application.output("Tab " + i + " has text " + tabs[i].text);
```

newTab

**JSTab** newTab(name, text, form, [relation])

Adds a new tab with the text label and JSForm and JSRelation (can be null for unrelated).

### Parameters

[{String}](#) name – The name of the new tab.

[{String}](#) text – The text to be displayed on the new tab.

[{JSForm}](#) form – The JSForm instance that should be displayed in the new tab.

[{Object}](#) [relation] – A JSRelation object that relates the parent form with the form that will be displayed in the new tab.

### Returns

[JSTab](#) – A JSTab instance representing the newly created and added tab.

### Sample

```
// Create a parent form.
var form = solutionModel.newForm('parentForm', 'example_data', 'parent_table', 'null', false, 640, 480);
// Create a first child form.
var childOne = solutionModel.newForm('childOne', 'example_data', 'child_table', 'null', false, 400, 300);
childOne.newField('child_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
// Create a relation to link the parent form to the first child form.
var parentToChild = solutionModel.newRelation
('parentToChild', 'example_data', 'parent_table', 'example_data', 'child_table', JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id', '=', 'child_table_parent_id');
// Create a second child form.
var childTwo = solutionModel.newForm('childTwo', 'example_data', 'my_table', 'null', false, 400, 300);
childTwo.newField('my_table_image', JSField.IMAGE_MEDIA, 10, 10, 100, 100);
// Create a tab panel and add two tabs to it, with the two child forms.
var tabPanel = form.newTabPanel('tabs', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child Two', childOne, parentToChild); // The first form uses the relation.
tabPanel.newTab('tab2', 'Child Two', childTwo);
```