

JSLabel

Extends

[JSComponent](#)

Property Summary

String	dataProviderID The dataprovider of the component.
Boolean	displaysTags Flag that enables or disables merging of data inside components using tags (placeholders).
Boolean	enabled The enable state of the component, default true.
String	format The format that should be applied when displaying data(using dataProviderID) in the label/button.
Number	labelSize Size property for a label, values 1 to 60 correspond to header class h1 to h6
String	name The name of the component.
String	styleClass The name of the style class that should be applied to this component.
String	text The text that is displayed inside the component.
Boolean	visible The visible property of the component, default true.
Number	x The x coordinate of the component on the form.
Number	y The y coordinate of the component on the form.

Method Summary

JSTitle	getTitle() Get title label for the label.
-------------------------	--

Property Details

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';
```

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 label = form.newLabel('You are viewing record no. %%parent_table_id%%. You are running on server %%
serverURL%%.',
                        10, 10, 600, 100);

label.displaysTags = true;
```

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;
```

format

The format that should be applied when displaying data(using dataProviderID) in the label/button. Some examples are "%%", "dd-MM-yyyy", "MM-dd-yyyy", etc.

Returns

[String](#)

Sample

```
var label = form.newLabel('', 10, 10, 100, 100);
label.format = '$#.00';
```

labelSize

Size property for a label, values 1 to 60 correspond to header class h1 to h6

Returns

[Number](#)

Sample

```
var label = form.newLabel('Hello', 1);
label.labelSize = 2 // corresponds to header class h2
```

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';
```

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.
```

text

The text that is displayed inside the component.

Returns

[String](#)

Sample

```
// In general the text is specified when creating the component.
var label = form.newLabel('Initial text', 10, 10, 100, 20);
// But it can be changed later if needed.
label.text = 'Changed text';
```

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;
```

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

getTitle

[JSTitle](#) `getTitle()`

Get title label for the label.

Returns

[JSTitle](#)

Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table');
var label = form.newLabel('Customers', 1);
label.getTitle().text = 'Some text'
forms['someForm'].controller.show()
```