

# JSTextArea

## Property Summary

String	<a href="#">dataProviderID</a> The dataprovider of the component.
Number	<a href="#">displayType</a> The type of display used by the field.
Boolean	<a href="#">enabled</a> The enable state of the component, default true.
String	<a href="#">format</a> The format that should be applied when displaying the data in the component.
String	<a href="#">name</a> The name of the component.
JSMETHOD	<a href="#">onAction</a> The method that is executed when the component is clicked.
JSMETHOD	<a href="#">onDataChange</a> Method that is executed when the data in the component is successfully changed.
String	<a href="#">placeholderText</a> The text that is displayed in field when the field doesn't have a text value.
Boolean	<a href="#">visible</a> The visible property of the component, default true.
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 Summary

JSTitle	<a href="#">getTitle()</a> Get title label for the field or 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';
```

### 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;
```

## 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 the data in the component.

There are different options for the different data provider 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 placeholder (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';
```

## name

The name of the component. Through this name it can also be 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();
```

## placeholderText

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

## Returns

[String](#)

## Sample

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

## 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 field or label.

### Returns

[JSTitle](#)

### Sample

```
var form = solutionModel.newForm('someForm', 'db:/example_data/parent_table');
var field = form.newField('parent_table_text', JSField.TEXT_FIELD, 1);
field.getTitle().text = 'Parent table'
forms['someForm'].controller.show()
```