

# JSForm

## Constants Summery

|        |                                                                                                   |
|--------|---------------------------------------------------------------------------------------------------|
| Number | <a href="#">#LIST_VIEW</a><br>The constants to set or get the view property of a JSForm.          |
| Number | <a href="#">#LOCKED_LIST_VIEW</a><br>The constants to set or get the view property of a JSForm.   |
| Number | <a href="#">#LOCKED_RECORD_VIEW</a><br>The constants to set or get the view property of a JSForm. |
| Number | <a href="#">#LOCKED_TABLE_VIEW</a><br>The constants to set or get the view property of a JSForm.  |
| Number | <a href="#">#RECORD_VIEW</a><br>The constants to set or get the view property of a JSForm.        |

## Property Summery

|        |                                                                                                                                                                                                             |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| String | <a href="#">#borderType</a><br>The type, color and style of border of the component.                                                                                                                        |
| String | <a href="#">#dataSource</a><br>The names of the database server and table that this form is linked to.                                                                                                      |
| String | <a href="#">#defaultPageFormat</a><br>The default page format for the form.                                                                                                                                 |
| JSForm | <a href="#">#extendsForm</a><br>A JSForm instance representing the super form of this form, if this form has a super form.                                                                                  |
| String | <a href="#">#initialSort</a><br>The default sort order only when the form loads.                                                                                                                            |
| String | <a href="#">#name</a><br>The name of the form.                                                                                                                                                              |
| String | <a href="#">#namedFoundSet</a><br>Property that tells the form to use a named foundset instead of the default foundset.                                                                                     |
| JSForm | <a href="#">#navigator</a><br>The navigator (previously named "controller") that is used to control/navigate to the form.                                                                                   |
| JSMeth | <a href="#">#onDeleteAllRecordsCmd</a><br>od The method that overrides the Servoy menu item Select > Delete All.                                                                                            |
| JSMeth | <a href="#">#onDeleteRecordCmd</a><br>od The method that overrides the Servoy menu item Select > Delete Record (or keyboard shortcut).                                                                      |
| JSMeth | <a href="#">#onDrag</a><br>od The method that is triggered when (non Design Mode) dragging occurs.                                                                                                          |
| JSMeth | <a href="#">#onDragOver</a><br>od The method that is triggered when (non Design Mode) dragging over a component occurs.                                                                                     |
| JSMeth | <a href="#">#onDrop</a><br>od The method that is triggered when (non Design Mode) dropping occurs.                                                                                                          |
| JSMeth | <a href="#">#onDuplicateRecordCmd</a><br>od The method that overrides the Servoy menu item Select > Duplicate Record (or keyboard shortcut).                                                                |
| JSMeth | <a href="#">#onElementFocusGained</a><br>od The method that is triggered when focus is gained by a component inside the form.                                                                               |
| JSMeth | <a href="#">#onElementFocusLost</a><br>od The method that gets triggered when focus is lost by a component inside the form.                                                                                 |
| JSMeth | <a href="#">#onFindCmd</a><br>od The method that overrides the Servoy menu item Select > Find (or keyboard shortcut) in Data (ready) mode.                                                                  |
| JSMeth | <a href="#">#onHide</a><br>od The method that is triggered when another form is being activated.                                                                                                            |
| JSMeth | <a href="#">#onInvertRecordsCmd</a><br>od The method that overrides the Servoy menu item Select > Invert Records.                                                                                           |
| JSMeth | <a href="#">#onLoad</a><br>od The method that is triggered when a form is loaded/reloaded from the repository; used to alter elements, set globals, hide toolbars, etc; onShow method can also be assigned. |
| JSMeth | <a href="#">#onNewRecordCmd</a><br>od The method that overrides the Servoy menu item Select > New Record (or keyboard shortcut).                                                                            |
| JSMeth | <a href="#">#onNextRecordCmd</a><br>od The method that overrides the Servoy menu item Select > Next Record.                                                                                                 |
| JSMeth | <a href="#">#onOmitRecordCmd</a><br>od The method that overrides the Servoy menu item Select > Omit Record.                                                                                                 |
| JSMeth | <a href="#">#onPreviousRecordCmd</a><br>od The method that overrides the Servoy menu item Select > Previous Record.                                                                                         |
| JSMeth | <a href="#">#onPrintPreviewCmd</a><br>od The method that overrides the Servoy menu item File > Print Preview.                                                                                               |

**JSMeth** [#onRecordEditStart](#)  
**od** The method that is triggered when a user clicks into a column on the form.  
**JSMeth** [#onRecordEditStop](#)  
**od** The method that is triggered when a record is being saved.  
**JSMeth** [#onRecordSelection](#)  
**od** The method that is triggered each time a record is selected.  
**JSMeth** [#onResize](#)  
**od** The method that gets triggered when resize occurs.  
**JSMeth** [#onSearchCmd](#)  
**od** The method that overrides the Servoy menu item Select > Search (or keyboard shortcut) in Find mode.  
**JSMeth** [#onShow](#)  
**od** The method that is triggered EVERY TIME the form is displayed; an argument must be passed to the method if this is the first time the form is displayed.  
**JSMeth** [#onShowAllRecordsCmd](#)  
**od** The method that overrides the Servoy menu item Select > Show All (or keyboard shortcut).  
**JSMeth** [#onShowOmittedRecordsCmd](#)  
**od** The method that overrides the Servoy menu item Select > Show Omitted Records.  
**JSMeth** [#onSortCmd](#)  
**od** The method that overrides the Servoy menu item Select > Sort.  
**JSMeth** [#onUnLoad](#)  
**od** The method that is triggered when a form is unloaded from the repository.  
**Number** [#paperPrintScale](#)  
The percentage value the printed page is enlarged or reduced to; the size of the printed form is inversely proportional.  
**String** [#rowBGColorCalculation](#)  
The calculation dataprovider used to add background color and highlight selected or alternate rows.  
**Number** [#scrollbars](#)  
Scrollbar options for the vertical and horizontal scrollbars.  
**String** [#serverName](#)  
Get the server name used by this form.  
**Boolean** [#showInMenu](#)  
When set, the form is displayed under the Window menu.  
**String** [#styleClass](#)  
The Cascading Style Sheet (CSS) class name applied to the form.  
**String** [#styleName](#)  
The name of the Servoy style that is being used on the form.  
**String** [#tableName](#)  
The [name of the table/SQL view].  
**String** [#titleText](#)  
The text that displays in the title bar of the form window.  
**Boolean** [#transparent](#)  
When set, the form is transparent.  
**Number** [#view](#)  
The default form view mode.  
**Number** [#width](#)  
The width of the form in pixels.

## Method Summery

**JSPart** [#getBodyPart\(\)](#)  
Retrieves the Body part of the form.  
**JSButt** [#getButton\(name\)](#)  
**on** Returns a JSButton that has the given name.  
**JSButt** [#getButtons\(\)](#)  
**on[]** Returns all JSButtons of this form, including the ones without a name.  
**JSCo** [#getComponent\(name\)](#)  
**mpone** Returns a JSCoComponent that has the given name; if found it will be a JSField, JSLabel, JSButton, JSPortal or JSTabPanel.  
**nt**  
**JSCo** [#getComponents\(\)](#)  
**mpone** Returns a array of all the JSCoComponents that a form has; they are of type JSField, JSLabel, JSButton, JSPortal or JSTabPanel.  
**nt[]**  
**JSField** [#getField\(name\)](#)  
The field with the specified name.  
**JSField** [#getFields\(\)](#)  
**[]** Returns all JSField objects of this form, including the ones without a name.  
**JSPart** [#getFooterPart\(\)](#)  
Retrieves the Footer part of the form.  
**JSMet** [#getFormMethod\(name\)](#)  
**hod** Gets an existing form method for the given name.  
**JSMet** [#getFormMethods\(\)](#)  
**hod[]** Returns all existing form methods for this form.

**JSVari** **#getFormVariable**(name)  
**able** Gets an existing form variable for the given name.

**JSVari** **#getFormVariables**()  
**able[]** An array consisting of all form variables for this form.

**JSPart** **#getHeaderPart**()  
 Retrieves the Header part of the form.

**JSLab** **#getLabel**(name)  
**el** Returns a JSLabel that has the given name.

**JSLab** **#getLabels**()  
**el[]** Returns all JSLabels of this form, including the ones without a name.

**JSPart** **#getLeadingGrandSummaryPart**()  
 Retrieves the Leading Grand Summary part of the form.

**JSPart** **#getLeadingSubSummaryParts**()  
**[]** Gets an array of the Leading Subsummary parts of the form, ordered by their height from top == 0 to bottom.

**JSPart** **#getPart**(type)  
**#getPart**(type, [height])  
**JSPart** Gets a part of the form from the given type (see JSPart constants).

**Numb** **#getPartYOffset**(type)  
**er** Returns the Y offset of a given part (see JSPart) of the form.

**JSPart** **#getParts**()  
**[]** Gets all the parts from the form, ordered by there height (lowerbound) property, from top == 0 to bottom.

**JSPort** **#getPortal**(name)  
**al** Returns a JSPortal that has the given name.

**JSPort** **#getPortals**()  
**al[]** Returns all JSPortal objects of this form, including the ones without a name.

**JSTab** **#getTabPanel**(name)  
**Panel** Returns a JSTabPanel that has the given name.

**JSTab** **#getTabPanels**()  
**Panel[]** Returns all JSTabPanels of this form, including the ones without a name.

**JSPart** **#getTitleFooterPart**()  
 Retrieves the Title Footer part of the form.

**JSPart** **#getTitleHeaderPart**()  
 Retrieves the Title Header part of the form.

**JSPart** **#getTrailingGrandSummaryPart**()  
 Retrieves the Trailing Grand Summary part of the form.

**JSPart** **#getTrailingSubSummaryParts**()  
**[]** Gets an array of the Trailing Subsummary parts of the form, ordered by their height from top == 0 to bottom.

**JSbutt** **#newButton**(text, x, y, width, height, jsmethod)  
**on** Creates a new button on the form with the given text, place, size and JSMMethod as the onAction event triggered action.

**JSField** **#newCalendar**(dataprovidername/jsvariable, x, y, width, height)  
 Creates a new JSField object on the form with the displayType of CALENDAR - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSField** **#newCheck**(dataprovidername/jsvariable, x, y, width, height)  
 Creates a new JSField object on the form with the displayType of CHECK (checkbox) - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSField** **#newComboBox**(dataprovidername/jsvariable, x, y, width, height)  
 Creates a new JSField object on the form with the displayType of COMBOBOX - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSField** **#newField**(dataprovidername/jsvariable, displaytype, x, y, width, height)  
 Creates a new JSField object on the form - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSPart** **#newFooterPart**()  
 Creates a new Footer part on the form.

**JSMet** **#newFormMethod**(code)  
**hod** Creates a new form JSMMethod - based on the specified code.

**JSVari** **#newFormVariable**(name, type)  
**able** Creates a new form JSVariable - based on the name of the variable object and the number type, uses the SolutionModel JSVariable constants.

**JSPart** **#newHeaderPart**()  
 Creates a new Header part on the form.

**JSField** **#newHtmlArea**(dataprovidername/jsvariable, x, y, width, height)  
 Creates a new JSField object on the form with the displayType of HTML\_AREA - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSField** **#newImageMedia**(dataprovidername/jsvariable, x, y, width, height)  
 Creates a new JSField object on the form with the displayType of IMAGE\_MEDIA - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSLab** **#newLabel**(txt, x, y, width, height)  
**el**

**JSLab** **#newLabel**(text, x, y, width, height, [jsmethod])  
**el** Creates a new JSLabel object on the form - including the text of the label, the "x" and "y" position of the label object in pixels, the width and height of the label object in pixels and the optional JSMMethod action such as the method for an onAction event.

**JSPart** **#newLeadingGrandSummaryPart**()  
 Creates a new Leading Grand Summary part on the form.

**JSPart** [#newLeadingSubSummaryPart\(\)](#)  
Creates a new Leading Subsummary part on the form.

**JSPart** [#newPart\(type, height\)](#)  
Creates a new part on the form.  
[#newPassword\(dataprovidername/jsvariable, x, y, width, height\)](#)

**JSField** [#newPortal\(name, relation, x, y, width, height\)](#)  
Creates a new JSField object on the form with the displayType of PASSWORD - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSPortal** [#newRadios\(dataprovidername/jsvariable, x, y, width, height\)](#)  
Creates a new JSPortal object on the form - including the name of the JSPortal object; the relation the JSPortal object is based on, the "x" and "y" position of the JSPortal object in pixels, as well as the width and height of the JSPortal object in pixels.

**JSField** [#newRtfArea\(dataprovidername/jsvariable, x, y, width, height\)](#)  
Creates a new JSField object on the form with the displayType of RTF\_AREA (enables more than one line of text to be displayed in a field) - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSTabPanel** [#newTabPanel\(name, x, y, width, height\)](#)  
Creates a new JSTabPanel object on the form - including the name of the JSTabPanel object, the "x" and "y" position of the JSTabPanel object in pixels, as well as the width and height of the JSTabPanel object in pixels.  
[#newTextArea\(dataprovidername/jsvariable, x, y, width, height\)](#)

**JSField** [#newTextField\(dataprovidername/jsvariable, x, y, width, height\)](#)  
Creates a new JSField object on the form with the displayType of TEXT\_AREA - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**JSField** [#newTitleFooterPart\(\)](#)  
Creates a new Title Footer part on the form.

**JSPart** [#newTitleHeaderPart\(\)](#)  
Creates a new Title Header part on the form.

**JSPart** [#newTrailingGrandSummaryPart\(\)](#)  
Creates a new Trailing Grand Summary part on the form.

**JSPart** [#newTrailingSubSummaryPart\(\)](#)  
Creates a new Trailing Subsummary part on the form.  
[#newTypeAhead\(dataprovidername/jsvariable, x, y, width, height\)](#)

**JSField** [#newTypeAhead\(dataprovidername/jsvariable, x, y, width, height\)](#)  
Creates a new JSField object on the form with the displayType of TYPE\_AHEAD - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

**Boole** [#removeButton\(name\)](#)  
**an** Removes a JSButton that has the specified name.

**Boole** [#removeComponent\(name\)](#)  
**an** Removes a component (JSLabel, JSButton, JSField, JSPortal, JSTabpanel) that has the given name.

**Boole** [#removeField\(name\)](#)  
**an** Removes a JSField that has the given name.

**Boole** [#removeLabel\(name\)](#)  
**an** Removes a JSLabel that has the given name.

**Boole** [#removePart\(type\)](#)  
**an**

**Boole** [#removePart\(type, \[height\]\)](#)  
**an** Removes a JSPart of the given type.

**Boole** [#removePortal\(name\)](#)  
**an** Removes a JSPortal that has the given name.

**Boole** [#removeTabPanel\(name\)](#)  
**an** Removes a JSTabPanel that has the given name.

## Constants Details

## LIST\_VIEW

The constants to set or get the view property of a JSForm.

They are as follows: JSForm.LIST\_VIEW, JSForm.LOCKED\_LIST\_VIEW, JSForm.LOCKED\_RECORD\_VIEW, JSForm.LOCKED\_TABLE\_VIEW, JSForm.RECORD\_VIEW.

### Returns

Number

### Sample

```
var myListViewForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', 'myStyleName', false, 800, 600);
myListViewForm.view = JSForm.LIST_VIEW;

var myLockedListViewForm = solutionModel.newForm('newForm2', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedListViewForm.view = JSForm.LOCKED_LIST_VIEW;

var myLockedRecordViewForm = solutionModel.newForm('newForm3', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedRecordViewForm.view = JSForm.LOCKED_RECORD_VIEW;

var myLockedTableViewForm = solutionModel.newForm('newForm4', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedTableViewForm.view = JSForm.LOCKED_TABLE_VIEW;

var myRecordViewForm = solutionModel.newForm('newForm5', 'myServer', 'myTable', 'myStyleName', false,
800, 600);
myRecordViewForm.view = JSForm.RECORD_VIEW;
```

## LOCKED\_LIST\_VIEW

The constants to set or get the view property of a JSForm.

They are as follows: JSForm.LIST\_VIEW, JSForm.LOCKED\_LIST\_VIEW, JSForm.LOCKED\_RECORD\_VIEW, JSForm.LOCKED\_TABLE\_VIEW, JSForm.RECORD\_VIEW.

### Returns

Number

### Sample

```
var myListViewForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', 'myStyleName', false, 800, 600);
myListViewForm.view = JSForm.LIST_VIEW;

var myLockedListViewForm = solutionModel.newForm('newForm2', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedListViewForm.view = JSForm.LOCKED_LIST_VIEW;

var myLockedRecordViewForm = solutionModel.newForm('newForm3', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedRecordViewForm.view = JSForm.LOCKED_RECORD_VIEW;

var myLockedTableViewForm = solutionModel.newForm('newForm4', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedTableViewForm.view = JSForm.LOCKED_TABLE_VIEW;

var myRecordViewForm = solutionModel.newForm('newForm5', 'myServer', 'myTable', 'myStyleName', false,
800, 600);
myRecordViewForm.view = JSForm.RECORD_VIEW;
```

### LOCKED\_RECORD\_VIEW

The constants to set or get the view property of a JSForm.

They are as follows: JSForm.LIST\_VIEW, JSForm.LOCKED\_LIST\_VIEW, JSForm.LOCKED\_RECORD\_VIEW, JSForm.LOCKED\_TABLE\_VIEW, JSForm.RECORD\_VIEW.

#### Returns

Number

#### Sample

```
var myListViewForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', 'myStyleName', false, 800, 600);
myListViewForm.view = JSForm.LIST_VIEW;

var myLockedListViewForm = solutionModel.newForm('newForm2', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedListViewForm.view = JSForm.LOCKED_LIST_VIEW;

var myLockedRecordViewForm = solutionModel.newForm('newForm3', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedRecordViewForm.view = JSForm.LOCKED_RECORD_VIEW;

var myLockedTableViewForm = solutionModel.newForm('newForm4', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedTableViewForm.view = JSForm.LOCKED_TABLE_VIEW;

var myRecordViewForm = solutionModel.newForm('newForm5', 'myServer', 'myTable', 'myStyleName', false,
800, 600);
myRecordViewForm.view = JSForm.RECORD_VIEW;
```

### LOCKED\_TABLE\_VIEW

The constants to set or get the view property of a JSForm.

They are as follows: JSForm.LIST\_VIEW, JSForm.LOCKED\_LIST\_VIEW, JSForm.LOCKED\_RECORD\_VIEW, JSForm.LOCKED\_TABLE\_VIEW, JSForm.RECORD\_VIEW.

#### Returns

Number

#### Sample

```
var myListViewForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', 'myStyleName', false, 800, 600);
myListViewForm.view = JSForm.LIST_VIEW;

var myLockedListViewForm = solutionModel.newForm('newForm2', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedListViewForm.view = JSForm.LOCKED_LIST_VIEW;

var myLockedRecordViewForm = solutionModel.newForm('newForm3', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedRecordViewForm.view = JSForm.LOCKED_RECORD_VIEW;

var myLockedTableViewForm = solutionModel.newForm('newForm4', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
myLockedTableViewForm.view = JSForm.LOCKED_TABLE_VIEW;

var myRecordViewForm = solutionModel.newForm('newForm5', 'myServer', 'myTable', 'myStyleName', false,
800, 600);
myRecordViewForm.view = JSForm.RECORD_VIEW;
```

## RECORD\_VIEW

The constants to set or get the view property of a JSForm.

They are as follows: JSForm.LIST\_VIEW, JSForm.LOCKED\_LIST\_VIEW, JSForm.LOCKED\_RECORD\_VIEW, JSForm.LOCKED\_TABLE\_VIEW, JSForm.RECORD\_VIEW.

### Returns

Number

### Sample

```
var myListViewForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', 'myStyleName', false, 800, 600);
    myListViewForm.view = JSForm.LIST_VIEW;

    var myLockedListViewForm = solutionModel.newForm('newForm2', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
    myLockedListViewForm.view = JSForm.LOCKED_LIST_VIEW;

    var myLockedRecordViewForm = solutionModel.newForm('newForm3', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
    myLockedRecordViewForm.view = JSForm.LOCKED_RECORD_VIEW;

    var myLockedTableViewForm = solutionModel.newForm('newForm4', 'myServer', 'myTable', 'myStyleName',
false, 800, 600);
    myLockedTableViewForm.view = JSForm.LOCKED_TABLE_VIEW;

    var myRecordViewForm = solutionModel.newForm('newForm5', 'myServer', 'myTable', 'myStyleName', false,
800, 600);
    myRecordViewForm.view = JSForm.RECORD_VIEW;
```

## Property Details

### 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';
```

### dataSource

The names of the database server and table that this form is linked to.

### Returns

String

### Sample

```
var myForm = solutionModel.newForm('newForm', 'aServer', 'aTable', 'aStyleName', false, 800, 600)
    myForm.dataSource = 'db:/anotherServerName/anotherTableName'
```

#### defaultPageFormat

The default page format for the form.

#### Returns

[String](#)

#### Sample

```
var form = solutionModel.getForm("someForm");
application.output(form.defaultPageFormat);
//HINT: To know exactly the notation of this property set it in the designer and then read it once out through
the solution model.
//It follows this convention:
//Orientation (0-Landscape,1-Portrait, 2-Reverse Landscape); Width; Height; ImageableX; ImageableY;
ImageableWidth; ImageableHeight
form.defaultPageFormat = 1;612.0;792.0;72.0;72.0;468.0;648.0
```

#### extendsForm

A JSForm instance representing the super form of this form, if this form has a super form.

#### Returns

[JSForm](#)

#### Sample

```
var subForm = solutionModel.newForm('childForm', 'myServer', 'myTable', null, true, 800, 600);
var superForm = solutionModel.newForm('childForm', 'myServer', 'myTable', null, true, 800, 600);
subForm.extendsForm = superForm;
```

#### initialSort

The default sort order only when the form loads.

This is applied each time an internal SQL query is being executed (find, find-all, open form); and is only executed when no other manual sort has been performed on the foundset.

#### Returns

[String](#)

#### Sample

```
var form = solutionModel.newForm('myForm', 'myServer', 'myTable', null, true, 800, 600);
form.initialSort = "column1 desc, column2 asc, column3 asc";
```

#### name

The name of the form.

#### Returns

[String](#)

#### Sample

```
var form = solutionModel.newForm('myForm', 'myServer', 'myTable', null, true, 800, 600);
var formName = form.name;
application.output(formName);
```

#### namedFoundSet

Property that tells the form to use a named foundset instead of the default foundset.

When "separate" as special value is specified the form will always create a copy of assigned foundset and therefor become separated from other foundsets

#### Returns

[String](#)

#### Sample

```
var firstForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var secondForm = solutionModel.newForm('newForm2', 'myServer', 'myTable', null, true, 800, 600);
forms['newForm2'].controller.find()
forms['newForm2'].columnTextDataProvider = 'aSearchValue';
forms['newForm2'].controller.search()
var myFoundSet = forms['newForm2'].foundset
firstForm.namedFoundSet = myFoundSet;
```

## navigator

The navigator (previously named "controller") that is used to control/navigate to the form. The navigator is shown at the left or at the right side of the form, depending on the page orientation.

The following options are available:

-none- - no navigator is assigned.

DEFAULT - the Servoy default navigator is assigned.

IGNORE - the navigator last assigned to a previous form.

Custom - a custom navigator based on a selected form.

### Returns

[JSForm](#)

### Sample

```
var aForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
// you can also use SM_DEFAULTS.IGNORE to just reuse the navigator that is already set, or SM_DEFAULTS.DEFAULT
to have the default servoy navigator.
// here we assign an other new form as the navigator.
var aNavigator = solutionModel.newForm('navForm', 'myServer', 'myTable', null, false, 800, 600);
// set the navigators navigator to NONE
aNavigator.navigator = SM_DEFAULTS.NONE; // Hide the navigator on the form.
myListViewForm.navigator = aNavigator;
application.output(myListViewForm.navigator.name);
```

## onDeleteAllRecordsCmd

The method that overrides the Servoy menu item Select > Delete All.

This property is automatically set to "DEFAULT" (no override) when the form is created.

### Returns

[JSMethod](#)

### Sample

```
form.onNewRecordCmd = form.newFormMethod('function onNewRecordCmd(event) { application.output("onNewRecordCmd
intercepted on " + event.getFormName()); }');
form.onDuplicateRecordCmd = form.newFormMethod('function onDuplicateRecordCmd(event) { application.output
("onDuplicateRecordCmd intercepted on " + event.getFormName()); }');
form.onDeleteRecordCmd = form.newFormMethod('function onDeleteRecordCmd(event) { application.output
("onDeleteRecordCmd intercepted on " + event.getFormName()); }');
form.onDeleteAllRecordsCmd = form.newFormMethod('function onDeleteAllRecordsCmd(event) { application.output
("onDeleteAllRecordsCmd intercepted on " + event.getFormName()); }');
```

## onDeleteRecordCmd

The method that overrides the Servoy menu item Select > Delete Record (or keyboard shortcut).

This property is automatically set to "DEFAULT" (no override) when the form is created.

### Returns

[JSMethod](#)

### Sample

```
form.onNewRecordCmd = form.newFormMethod('function onNewRecordCmd(event) { application.output("onNewRecordCmd
intercepted on " + event.getFormName()); }');
form.onDuplicateRecordCmd = form.newFormMethod('function onDuplicateRecordCmd(event) { application.output
("onDuplicateRecordCmd intercepted on " + event.getFormName()); }');
form.onDeleteRecordCmd = form.newFormMethod('function onDeleteRecordCmd(event) { application.output
("onDeleteRecordCmd intercepted on " + event.getFormName()); }');
form.onDeleteAllRecordsCmd = form.newFormMethod('function onDeleteAllRecordsCmd(event) { application.output
("onDeleteAllRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onDrag

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

#### Returns

[JSMethod](#)

#### Sample

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

#### onDuplicateRecordCmd

The method that overrides the Servoy menu item Select > Duplicate Record (or keyboard shortcut).

This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onNewRecordCmd = form.newFormMethod('function onNewRecordCmd(event) { application.output("onNewRecordCmd intercepted on " + event.getFormName()); }');
form.onDuplicateRecordCmd = form.newFormMethod('function onDuplicateRecordCmd(event) { application.output("onDuplicateRecordCmd intercepted on " + event.getFormName()); }');
form.onDeleteRecordCmd = form.newFormMethod('function onDeleteRecordCmd(event) { application.output("onDeleteRecordCmd intercepted on " + event.getFormName()); }');
form.onDeleteAllRecordsCmd = form.newFormMethod('function onDeleteAllRecordsCmd(event) { application.output("onDeleteAllRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onElementFocusGained

The method that is triggered when focus is gained by a component inside the form.

#### Returns

[JSMethod](#)

#### Sample

```
form.onElementFocusGained = form.newFormMethod('function onElementFocusGained(event) { application.output("onElementFocusGained intercepted from " + event.getSource()); }');
form.onElementFocusLost = form.newFormMethod('function onElementFocusLost(event) { application.output("onElementFocusLost intercepted from " + event.getSource()); }');
```

#### onElementFocusLost

The method that gets triggered when focus is lost by a component inside the form.

#### Returns

[JSMethod](#)

#### Sample

```
form.onElementFocusGained = form.newFormMethod('function onElementFocusGained(event) { application.output("onElementFocusGained intercepted from " + event.getSource()); }');
form.onElementFocusLost = form.newFormMethod('function onElementFocusLost(event) { application.output("onElementFocusLost intercepted from " + event.getSource()); }');
```

#### onFindCmd

The method that overrides the Servoy menu item Select > Find (or keyboard shortcut) in Data (ready) mode. This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onFindCmd = form.newFormMethod('function onFindCmd(event) { application.output("onFindCmd intercepted on " + event.getFormName()); }');
form.onSearchCmd = form.newFormMethod('function onSearchCmd(event) { application.output("onSearchCmd intercepted on " + event.getFormName()); }');
form.onShowAllRecordsCmd = form.newFormMethod('function onShowAllRecordsCmd(event) { application.output("onShowAllRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onHide

The method that is triggered when another form is being activated.

NOTE: If the onHide method returns false, the form can be prevented from hiding.

For example, when using onHide with showFormInDialog, the form will not close by clicking the dialog close box (X).

#### Returns

[JSMethod](#)

#### Sample

```
form.onShow = form.newFormMethod('function onShow(firstShow, event) { application.output("onShow intercepted on " + event.getFormName() + ". first show? " + firstShow); return false; }');
form.onHide = form.newFormMethod('function onHide(event) { application.output("onHide blocked on " + event.getFormName()); return false; }');
```

#### onInvertRecordsCmd

The method that overrides the Servoy menu item Select > Invert Records.  
This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onOmitRecordCmd = form.newFormMethod('function onOmitRecordCmd(event) { application.output  
("onOmitRecordCmd intercepted on " + event.getFormName()); }');  
form.onShowOmittedRecordsCmd = form.newFormMethod('function onShowOmittedRecordsCmd(event) { application.output  
("onShowOmittedRecordsCmd intercepted on " + event.getFormName()); }');  
form.onInvertRecordsCmd = form.newFormMethod('function onInvertRecordsCmd(event) { application.output  
("onInvertRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onLoad

The method that is triggered when a form is loaded/reloaded from the repository; used to alter elements, set globals, hide toolbars, etc; onShow method can also be assigned.

NOTE: onShow should be used to access current foundset dataproviders; onLoad cannot be used because the foundset data is not loaded until after the form is loaded.

NOTE: the onLoad event bubbles up, meaning that the onLoad event of a form displayed in a tabPanel is fired before the onLoad event of the parent.

#### Returns

[JSMethod](#)

#### Sample

```
form.onLoad = form.newFormMethod('function onLoad(event) { application.output("onLoad intercepted on " + event.  
getFormName()); }');  
form.onUnLoad = form.newFormMethod('function onUnLoad(event) { application.output("onUnLoad intercepted on " +  
event.getFormName()); }');
```

#### onNewRecordCmd

The method that overrides the Servoy menu item Select > New Record (or keyboard shortcut).  
This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onNewRecordCmd = form.newFormMethod('function onNewRecordCmd(event) { application.output("onNewRecordCmd  
intercepted on " + event.getFormName()); }');  
form.onDuplicateRecordCmd = form.newFormMethod('function onDuplicateRecordCmd(event) { application.output  
("onDuplicateRecordCmd intercepted on " + event.getFormName()); }');  
form.onDeleteRecordCmd = form.newFormMethod('function onDeleteRecordCmd(event) { application.output  
("onDeleteRecordCmd intercepted on " + event.getFormName()); }');  
form.onDeleteAllRecordsCmd = form.newFormMethod('function onDeleteAllRecordsCmd(event) { application.output  
("onDeleteAllRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onNextRecordCmd

The method that overrides the Servoy menu item Select > Next Record.  
This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onPreviousRecordCmd = form.newFormMethod('function onPreviousRecordCmd(event) { application.output  
("onPreviousRecordCmd intercepted on " + event.getFormName()); }');  
form.onNextRecordCmd = form.newFormMethod('function onNextRecordCmd(event) { application.output  
("onNextRecordCmd intercepted on " + event.getFormName()); }');
```

#### onOmitRecordCmd

The method that overrides the Servoy menu item Select > Omit Record.  
This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onOmitRecordCmd = form.newFormMethod('function onOmitRecordCmd(event) { application.output
("onOmitRecordCmd intercepted on " + event.getFormName()); }');
form.onShowOmittedRecordsCmd = form.newFormMethod('function onShowOmittedRecordsCmd(event) { application.output
("onShowOmittedRecordsCmd intercepted on " + event.getFormName()); }');
form.onInvertRecordsCmd = form.newFormMethod('function onInvertRecordsCmd(event) { application.output
("onInvertRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onPreviousRecordCmd

The method that overrides the Servoy menu item Select > Previous Record.  
This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onPreviousRecordCmd = form.newFormMethod('function onPreviousRecordCmd(event) { application.output
("onPreviousRecordCmd intercepted on " + event.getFormName()); }');
form.onNextRecordCmd = form.newFormMethod('function onNextRecordCmd(event) { application.output
("onNextRecordCmd intercepted on " + event.getFormName()); }');
```

#### onPrintPreviewCmd

The method that overrides the Servoy menu item File > Print Preview.  
This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onPrintPreviewCmd = form.newFormMethod('function onPrintPreviewCmd(event) { application.output
("onPrintPreviewCmd intercepted on " + event.getFormName()); }');
```

#### onRecordEditStart

The method that is triggered when a user clicks into a column on the form.  
NOTE: There is a small "e" displayed in the lower left side of the Servoy Client screen in the status area at the bottom of the window when the record is being edited.

#### Returns

[JSMethod](#)

#### Sample

```
form.onRecordEditStart = form.newFormMethod('function onRecordEditStart(event) { application.output
("onRecordEditStart intercepted on " + event.getFormName()); }');
form.onRecordEditStop = form.newFormMethod('function onRecordEditStop(record, event) { application.output
("onRecordEditStop intercepted on " + event.getFormName() + ". record is: " + record); }');
form.onRecordSelection = form.newFormMethod('function onRecordSelection(event) { application.output
("onRecordSelection intercepted on " + event.getFormName()); }');
```

#### onRecordEditStop

The method that is triggered when a record is being saved.

A record is saved when a user clicks out of it (for example on an empty part of the layout or to another form); can return false (for example as part of a validation) where a condition must be changed to return true.

NOTE: The name of this property has been changed from onRecordSave.

#### Returns

[JSMethod](#)

#### Sample

```
form.onRecordEditStart = form.newFormMethod('function onRecordEditStart(event) { application.output
("onRecordEditStart intercepted on " + event.getFormName()); }');
form.onRecordEditStop = form.newFormMethod('function onRecordEditStop(record, event) { application.output
("onRecordEditStop intercepted on " + event.getFormName() + ". record is: " + record); }');
form.onRecordSelection = form.newFormMethod('function onRecordSelection(event) { application.output
("onRecordSelection intercepted on " + event.getFormName()); }');
```

#### onRecordSelection

The method that is triggered each time a record is selected.

If a form is in List view or Special table view - when the user clicks on it.

In Record view - after the user navigates to another record using the slider or clicks up or down for next/previous record.

NOTE: Data and Servoy tag values are returned when the onRecordSelection method is executed.

#### Returns

[JSMethod](#)

#### Sample

```
form.onRecordEditStart = form.newFormMethod('function onRecordEditStart(event) { application.output
("onRecordEditStart intercepted on " + event.getFormName()); }');
form.onRecordEditStop = form.newFormMethod('function onRecordEditStop(record, event) { application.output
("onRecordEditStop intercepted on " + event.getFormName() + ". record is: " + record); }');
form.onRecordSelection = form.newFormMethod('function onRecordSelection(event) { application.output
("onRecordSelection intercepted on " + event.getFormName()); }');
```

#### onResize

The method that gets triggered when resize occurs.

#### Returns

[JSMethod](#)

#### Sample

```
form.onResize = form.newFormMethod('function onResize(event) { application.output("onResize intercepted on " +
event.getFormName()); }');
```

#### onSearchCmd

The method that overrides the Servoy menu item Select > Search (or keyboard shortcut) in Find mode.

This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onFindCmd = form.newFormMethod('function onFindCmd(event) { application.output("onFindCmd intercepted on "
+ event.getFormName()); }');
form.onSearchCmd = form.newFormMethod('function onSearchCmd(event) { application.output("onSearchCmd
intercepted on " + event.getFormName()); }');
form.onShowAllRecordsCmd = form.newFormMethod('function onShowAllRecordsCmd(event) { application.output
("onShowAllRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onShow

The method that is triggered EVERY TIME the form is displayed; an argument must be passed to the method if this is the first time the form is displayed.

NOTE: onShow can be used to access current foundset dataproviders; onLoad cannot be used because the foundset data is not loaded until after the form is loaded.

NOTE: the onShow event bubbles down, meaning that the onShow event of a form displayed in a tabPanel is fired after the onShow event of the parent.

#### Returns

[JSMethod](#)

#### Sample

```
form.onShow = form.newFormMethod('function onShow(firstShow, event) { application.output("onShow intercepted on " + event.getFormName() + ". first show? " + firstShow); return false; }');
form.onHide = form.newFormMethod('function onHide(event) { application.output("onHide blocked on " + event.getFormName()); return false; }');
```

#### onShowAllRecordsCmd

The method that overrides the Servoy menu item Select > Show All (or keyboard shortcut). This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onFindCmd = form.newFormMethod('function onFindCmd(event) { application.output("onFindCmd intercepted on " + event.getFormName()); }');
form.onSearchCmd = form.newFormMethod('function onSearchCmd(event) { application.output("onSearchCmd intercepted on " + event.getFormName()); }');
form.onShowAllRecordsCmd = form.newFormMethod('function onShowAllRecordsCmd(event) { application.output("onShowAllRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onShowOmittedRecordsCmd

The method that overrides the Servoy menu item Select > Show Omitted Records. This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onOmitRecordCmd = form.newFormMethod('function onOmitRecordCmd(event) { application.output("onOmitRecordCmd intercepted on " + event.getFormName()); }');
form.onShowOmittedRecordsCmd = form.newFormMethod('function onShowOmittedRecordsCmd(event) { application.output("onShowOmittedRecordsCmd intercepted on " + event.getFormName()); }');
form.onInvertRecordsCmd = form.newFormMethod('function onInvertRecordsCmd(event) { application.output("onInvertRecordsCmd intercepted on " + event.getFormName()); }');
```

#### onSortCmd

The method that overrides the Servoy menu item Select > Sort. This property is automatically set to "DEFAULT" (no override) when the form is created.

#### Returns

[JSMethod](#)

#### Sample

```
form.onSortCmd = form.newFormMethod('function onSortCmd(dataProviderID, asc, event) { application.output("onSortCmd intercepted on " + event.getFormName() + ". data provider: " + dataProviderID + ". asc: " + asc); }');
```

#### onUnLoad

The method that is triggered when a form is unloaded from the repository.

NOTE: Forms can be prevented from being removed from memory by referencing the form object in a global variable or inside an array inside a global variable. Do take care using this technique.

Forms take up memory and if too many forms are in memory and cannot be unloaded, there is a possibility of running out of memory.

#### Returns

[JSMethod](#)

#### Sample

```
form.onLoad = form.newFormMethod('function onLoad(event) { application.output("onLoad intercepted on " + event.getFormName()); }');
form.onUnLoad = form.newFormMethod('function onUnLoad(event) { application.output("onUnLoad intercepted on " + event.getFormName()); }');
```

#### paperPrintScale

The percentage value the printed page is enlarged or reduced to; the size of the printed form is inversely proportional. For example, if the paperPrintScale is 50, the printed form will be enlarged 200%.

#### Returns

[Number](#)

#### Sample

```
var form = solutionModel.newForm('myForm','myServer','myTable',null,true,800,600);
if (form.paperPrintScale < 100)
    form.paperPrintScale = 100;
```

#### rowBGColorCalculation

The calculation dataprovider used to add background color and highlight selected or alternate rows.

The default is -none-.

#### Returns

[String](#)

#### Sample

```
var form = solutionModel.newForm('myForm','myServer','myTable',null,true,800,600);
//assign the global method as a string. Or use a calculation name as the string.
form.rowBGColorCalculation = "globals.calculationDataProvider";
```

#### 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 form = solutionModel.newForm('myForm','myServer','myTable',null,true,1000,600);
form.scrollbars = SM_SCROLLBAR.VERTICAL_SCROLLBAR_NEVER;
forms['newForm1'].controller.show();
```

#### serverName

Get the server name used by this form.

#### Returns

[String](#)

#### Sample

```
var form = solutionModel.newForm('myForm','myServer','myTable',null,true,800,600);
form.serverName = 'anotherServerName';
var theServerName = form.getServerName();
application.output(theServerName);
```

#### showInMenu

When set, the form is displayed under the Window menu.

If it is not set, the form will be 'hidden'.

NOTE: This is only applicable for Servoy Client. Servoy Developer always shows all forms so that developers have access to all forms within a solution during development.

#### Returns

Boolean

#### Sample

```
var aForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var anotherForm= solutionModel.newForm('newForm2', 'myServer', 'myTable', null, true, 800, 600);
    //using 'anotherForm' as navigator for aForm
    anotherForm.showInMenu = false;
    anotherForm.navigator = null;
    aForm.navigator = anotherForm;
    application.output(aForm.navigator.name);
```

#### styleClass

The Cascading Style Sheet (CSS) class name applied to the form.

#### Returns

String

#### Sample

```
var aForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
if (aForm.styleClass == null)
    aForm.styleClass = someStyleClass;
else
    application.output("The Cascading Style Sheet (CSS) class name applied to this form is " + aForm.
styleClass);
```

#### styleName

The name of the Servoy style that is being used on the form.

#### Returns

String

#### Sample

```
var aForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
if (aForm.styleName == null)
    aForm.styleName = someServoyStyleName;
else
    application.output("The name of the Servoy style that is being used on the form is " + aForm.styleName);
```

#### tableName

The [name of the table/SQL view].[the name of the database server connection] the form is based on.

#### Returns

String

#### Sample

```
var aForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    aForm.tableName = 'anotherTableOfMine'
    if (forms['newForm1'].controller.find())
    {
        columnTextDataProvider = '=aSearchedValue'
        columnNumberDataProvider = '>10';
        forms['newForm1'].controller.search()
    }
```

#### titleText

The text that displays in the title bar of the form window.

NOTE: Data tags and Servoy tags can be used as part of the title text.

#### Returns

[String](#)

#### Sample

```
var myForm = solutionModel.newForm('newForm', 'aServer', 'aTable', 'aStyleName', false, 800, 600)
forms['newForm'].controller.show();
if (myForm.titleText == null)
{
    myForm.titleText = "My new title text should be really cool!"
    forms['newForm'].controller.recreateUI();
}
else
    application.output("My text text is already cool");
```

#### transparent

When set, the form is transparent.

#### Returns

[Boolean](#)

#### Sample

```
var form = solutionModel.newForm('myForm', 'myServer', 'myTable', null, true, 1000, 800);
if (form.transparent == false)
{
    var style = solutionModel.newStyle('myStyle', 'form { background-color: yellow; }');
    style.text = style.text + 'field { background-color: blue; }';
    form.styleName = 'myStyle';
}
var field = form.newField('columnTextDataProvider', JSField.TEXT_FIELD, 100, 100, 100, 50);
forms['myForm'].controller.show();
```

#### view

The default form view mode.

The view can be changed using a method at runtime. The following views are available:

- Record view
- List view
- Record view (locked)
- List view (locked)
- Table View (locked)

NOTE: Only Table View (locked) uses asynchronized related data loading.

This feature defers all related foundset data loading to the background - enhancing the visual display of a related foundset.

#### Returns

[Number](#)

#### Sample

```
var myForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
myForm.view = JSForm.RECORD_VIEW;
forms['newForm1'].controller.show();
```

#### width

The width of the form in pixels.

#### Returns

[Number](#)

#### Sample

```
var myForm = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
forms['newForm1'].controller.show();
myForm.width = 120;
forms['newForm1'].controller.recreateUI();
```

## Method Details

getBodyPart

[JSPart](#) **getBodyPart()**

Retrieves the Body part of the form.

### Returns

[JSPart](#) – A JSPart instance corresponding to the Body part of the form.

### Sample

```
form.getBodyPart().background = 'blue';
```

getButton

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

Returns a JSButton that has the given name.

### Parameters

[String](#) name – the specified name of the button

### Returns

[JSButton](#) – a JSButton object

### Sample

```
var btn = myForm.getButton("hello");
application.output(btn.text);
```

getButtons

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

Returns all JSButtons of this form, including the ones without a name.

### Returns

[JSButton](#)[] – the list of all JSbuttons on this forms

### Sample

```
var buttons = myForm.getButtons();
for (var b in buttons)
{
    if (buttons[b].name != null)
        application.output(buttons[b].name);
    else
        application.output(buttons[b].text + " has no name ");
}
```

getComponent

[JSComponent](#) **getComponent(name)**

Returns a JSComponent that has the given name; if found it will be a JSField, JSLabel, JSButton, JSPortal or JSTabPanel.

### Parameters

[String](#) name – the specified name of the component

### Returns

[JSComponent](#) – a JSComponent object (might be a JSField, JSLabel, JSButton, JSPortal or JSTabPanel)

### Sample

```
var frm = solutionModel.getForm("myForm");
var cmp = frm.getComponent("componentName");
application.output("Component type and name: " + cmp);
```

getComponents

[JSComponent](#)[] **getComponents()**

Returns a array of all the JSComponents that a form has; they are of type JSField,JSLabel,JSButton,JSPortal or JSTabPanel.

### Returns

[JSComponent](#)[] – an array of all the JSComponents on the form.

### Sample

```
var form = solutionModel.getForm("myForm");
    var components = form.getComponents();
    for (var i in components)
        application.output("Component type and name: " + components[i]);
```

### getField

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

The field with the specified name.

#### Parameters

{**String**} name – the specified name of the field

#### Returns

**JSField** – a JSField object

### Sample

```
var form = solutionModel.getForm("myForm");
    var field = form.getField("myField");
    application.output(field.dataProviderID);
```

### getFields

**JSField[]** **getFields**()

Returns all JSField objects of this form, including the ones without a name.

#### Returns

**JSField[]** – all JSField objects of this form

### Sample

```
var frm = solutionModel.getForm("myForm");
    var fields = frm.getFields();
    for (var f in fields)
    {
        var fname = fields[f].name;
        if (fname != null)
            application.output(fname);
    }
```

### getFooterPart

**JSPart** **getFooterPart**()

Retrieves the Footer part of the form.

#### Returns

**JSPart** – A JSPart instance corresponding to the Footer part of the form.

### Sample

```
form.getFooterPart().background = 'magenta';
```

### getFormMethod

**JSMethod** **getFormMethod**(name)

Gets an existing form method for the given name.

#### Parameters

{**String**} name – the specified name of the method

#### Returns

**JSMethod** – a JSMethod object (or null if the method with the specified name does not exist)

### Sample

```
var frm = solutionModel.getForm("myForm");
    var method = frm.getFormMethod("myMethod");
    application.output(method.code);
```

### getFormMethods

**JSMethod[]** **getFormMethods**()

Returns all existing form methods for this form.

**Returns**

[JSMethod\[\]](#) – all form methods for the form

**Sample**

```
var frm = solutionModel.getForm("myForm");
var methods = frm.getFormMethods();
for (var m in methods)
    application.output(methods[m].getName());
```

**getFormVariable**

[JSVariable](#) **getFormVariable**(name)

Gets an existing form variable for the given name.

**Parameters**

[{String}](#) name – the specified name of the variable

**Returns**

[JSVariable](#) – a JSVariable object

**Sample**

```
var frm = solutionModel.getForm("myForm");
var fvariable = frm.getFormVariable("myVarName");
application.output(fvariable.name + " has the default value of " + fvariable.defaultValue);
```

**getFormVariables**

[JSVariable\[\]](#) **getFormVariables**()

An array consisting of all form variables for this form.

**Returns**

[JSVariable\[\]](#) – an array of all variables on this form

**Sample**

```
var frm = solutionModel.getForm("myForm");
var variables = frm.getFormVariables();
for (var i in variables)
    application.output(variables[i].name);
```

**getHeaderPart**

[JSPart](#) **getHeaderPart**()

Retrieves the Header part of the form.

**Returns**

[JSPart](#) – A JSPart instance corresponding to the Header part of the form.

**Sample**

```
form.getHeaderPart().background = 'orange';
```

**getLabel**

[JSLabel](#) **getLabel**(name)

Returns a JSLabel that has the given name.

**Parameters**

[{String}](#) name – the specified name of the label

**Returns**

[JSLabel](#) – a JSLabel object (or null if the label with the specified name does not exist)

**Sample**

```
var frm = solutionModel.getForm("myForm");
var label = frm.getLabel("myLabel");
application.output(label.text);
```

**getLabels**

[JSLabel\[\]](#) **getLabels**()

Returns all JSLabels of this form, including the ones without a name.

**Returns**

[JSLabel\[\]](#) – all JSLabels on this form

### Sample

```
var frm = solutionModel.getForm("myForm");
var labels = frm.getLabels();
for (var i in labels)
{
    var lname = labels[i].name;
    if (lname != null)
        application.output(lname);
}
```

### getLeadingGrandSummaryPart

#### JSPart getLeadingGrandSummaryPart()

Retrieves the Leading Grand Summary part of the form.

#### Returns

[JSPart](#) – A JSPart instance corresponding to the Leading Grand Summary part of the form.

### Sample

```
form.getLeadingGrandSummaryPart().background = 'yellow';
```

### getLeadingSubSummaryParts

#### JSPart[] getLeadingSubSummaryParts()

Gets an array of the Leading Subsummary parts of the form, ordered by their height from top == 0 to bottom.

#### Returns

[JSPart\[\]](#) – An array of JSPart instances corresponding to the Leading Subsummary parts of the form.

### Sample

```
form.getLeadingSubSummaryParts()[0].background = 'green';
```

### getPart

#### JSPart getPart(type, [height])

Gets a part of the form from the given type (see JSPart constants).

Use the optional height if you want to get a specific LEADING\_SUBSUMMARY or TRAILING\_SUBSUMMARY.

#### Parameters

[{Number}](#) type – The type of the part to retrieve.

[{Number}](#) [height] – The height of the part to retrieve. Use this parameter when retrieving one of multiple Leading/Trailing Subsummary parts.

#### Returns

[JSPart](#) – A JSPart instance representing the retrieved form part.

### Sample

```
form.getPart(JSPart.HEADER).background = 'red';
form.getPart(JSPart.LEADING_SUBSUMMARY, 160).background = 'red';
```

### getPartYOffset

#### Number getPartYOffset(type, [height])

Returns the Y offset of a given part (see JSPart) of the form. This will include all the super forms parts if this form extends a form. Use the optional height for targetting one of multiple subsummary parts.

#### Parameters

[{Number}](#) type – The type of the part whose Y offset will be returned.

[{Number}](#) [height] – The height of the part whose Y offset will be returned. This is used when one of multiple Leading/Trailing Sumsummary parts is retrieved.

#### Returns

[Number](#) – A number holding the Y offset of the specified form part.

### Sample

```
// get the subform
var form = solutionModel.getForm('SubForm');
// get the start offset of the body
var height = form.getPartYOffset(JSPart.BODY);
// place a new button based on the start offset.
form.newButton('mybutton', 50, 50+height, 80, 20, solutionModel.getGlobalMethod('test'));
```

### getParts

**JSPart[] getParts()**

Gets all the parts from the form, ordered by there height (lowerbound) property, from top == 0 to bottom.

### Returns

**JSPart[]** – An array of JSPart instances corresponding to the parts of the form.

### Sample

```
var allParts = form.getParts()
for (var i=0; i<allParts.length; i++) {
    if (allParts[i].getPartType() == JSPart.BODY)
        application.output('body Y offset: ' + allParts[i].getPartYOffset());
}
```

### getPortal

**JSPortal getPortal(name)**

Returns a JSPortal that has the given name.

### Parameters

**{String} name** – the specified name of the portal

### Returns

**JSPortal** – a JSPortal object

### Sample

```
var frm = solutionModel.getForm("myForm");
var portal = frm.getPortal("myPortal");
portal.initialSort = 'my_table_text desc';
```

### getPortals

**JSPortal[] getPortals()**

Returns all JSPortal objects of this form, including the ones without a name.

### Returns

**JSPortal[]** – an array of all JSPortal objects on this form

### Sample

```
var frm = solutionModel.getForm("myForm");
var portals = frm.getPortals();
for (var i in portals)
{
    var p = portals[i];
    if (p.name != null)
        application.output(p.name);
    else
        application.output("unnamed portal detected");
}
```

### getTabPanel

**JSTabPanel getTabPanel(name)**

Returns a JSTabPanel that has the given name.

### Parameters

**{String} name** – the specified name of the tabpanel

### Returns

**JSTabPanel** – a JSTabPanel object

### Sample

```
var frm = solutionModel.getForm("myForm");
var tabPanel = frm.getTabPanel("myTabPanel");
var tabs = tabPanel.getTabs();
for (var i=0; i<tabs.length; i++)
    application.output("Tab " + i + " has text " + tabs[i].text);
```

getTabPanels

**JSTabPanel[] getTabPanels()**

Returns all JSTabPanels of this form, including the ones without a name.

#### Returns

**JSTabPanel[]** – an array of all JSTabPanel objects on this form

### Sample

```
var frm = solutionModel.getForm("myForm");
var tabPanels = frm.getTabPanels();
for (var i in tabPanels)
{
    var tp = tabPanels[i];
    if (tp.name != null)
        application.output("Tab " + tp.name + " has text " + tp.text);
    else
        application.output("Tab with text " + tp.text + " has no name");
}
```

getTitleFooterPart

**JSPart getTitleFooterPart()**

Retrieves the Title Footer part of the form.

#### Returns

**JSPart** – A JSPart instance corresponding to the Title Footer part of the form.

### Sample

```
form.getTitleFooterPart().background = 'gray';
```

getTitleHeaderPart

**JSPart getTitleHeaderPart()**

Retrieves the Title Header part of the form.

#### Returns

**JSPart** – A JSPart instance corresponding to the Title Header part of the form.

### Sample

```
form.getTitleHeaderPart().background = 'red';
```

getTrailingGrandSummaryPart

**JSPart getTrailingGrandSummaryPart()**

Retrieves the Trailing Grand Summary part of the form.

#### Returns

**JSPart** – A JSPart instance corresponding to the Trailing Grand Summary part of the form.

### Sample

```
form.getTrailingGrandSummaryPart().background = 'yellow';
```

getTrailingSubSummaryParts

**JSPart[] getTrailingSubSummaryParts()**

Gets an array of the Trailing Subsummary parts of the form, ordered by their height from top == 0 to bottom.

#### Returns

**JSPart[]** – An array of JSPart instances corresponding to the Trailing Subsummary parts of the form.

### Sample

```
form.getTrailingSubSummaryParts()[0].background = 'green';
```

### newButton

**JSButton** **newButton**(text, x, y, width, height, jsmethod)

Creates a new button on the form with the given text, place, size and JSMethod as the onAction event triggered action.

#### Parameters

text – the text on the button

{**Number**} x – the x coordinate of the button location on the form

{**Number**} y – the y coordinate of the button location on the form

{**Number**} width – the width of the button

{**Number**} height – the height of the button

jsmethod – the method assigned to handle an onAction event

#### Returns

**JSButton** – a new JSButton object

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var method = solutionModel.newFormMethod('function onAction(event) { application.output("onAction
intercepted on " + event.getFormName()); }');
    var button = solutionModel.newButton('myButton', 10, 10, 100, 30, method);
    application.output("The new button: " + button.name + " has the following onAction event handling
method assigned " + button.onAction.getName());
```

### newCalendar

**JSField** **newCalendar**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of CALENDAR - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

#### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{**Number**} x – the horizontal "x" position of the JSfield object in pixels

{**Number**} y – the vertical "y" position of the JSField object in pixels

{**Number**} width – the width of the JSField object in pixels

{**Number**} height – the height of the JSField object in pixels

#### Returns

**JSField** – a new JSField object on the form with the displayType of CALENDAR

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var calendar = form.newCalendar(myDataProvider, 100, 100, 200, 200);
    forms['newForm1'].controller.show();
```

### newCheck

**JSField** **newCheck**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of CHECK (checkbox) - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

#### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{**Number**} x – the horizontal "x" position of the JSfield object in pixels

{**Number**} y – the vertical "y" position of the JSField object in pixels

{**Number**} width – the width of the JSField object in pixels

{**Number**} height – the height of the JSField object in pixels

#### Returns

**JSField** – a new JSField object on the form with the displayType of CHECK (checkbox)

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var calendar = form.newCheck(myDataProvider, 100, 100, 200, 200);
    forms['newForm1'].controller.show();
```

### newComboBox

**JSField** **newComboBox**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of COMBOBOX - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{Number} x – the horizontal "x" position of the JSField object in pixels

{Number} y – the vertical "y" position of the JSField object in pixels

{Number} width – the width of the JSField object in pixels

{Number} height – the height of the JSField object in pixels

### Returns

JSField – a new JSField object on the form with the displayType of COMBOBOX

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var calendar = form.newComboBox(myDataProvider, 100, 100, 200, 200);
    forms['newForm1'].controller.show();
```

### newField

JSField **newField**(dataprovidername/jsvariable, displaytype, x, y, width, height)

Creates a new JSField object on the form - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

displaytype – the display type of the JSField object (see the Solution Model -> JSField node for display types)

{Number} x – the horizontal "x" position of the JSField object in pixels

{Number} y – the vertical "y" position of the JSField object in pixels

{Number} width – the width of the JSField object in pixels

{Number} height – the height of the JSField object in pixels

### Returns

JSField – a new JSField object (of the specified display type)

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var variable = form.newFormVariable('myVar', JSVariable.TEXT);
    variable.defaultValue = "'This is a default value (with tripple quotes)!'";
    var field = form.newField(variable, JSField.TEXT_FIELD, 100, 100, 200, 200);
    forms['newForm1'].controller.show();
```

### newFooterPart

JSPart **newFooterPart**()

Creates a new Footer part on the form.

### Returns

JSPart – A JSPart instance corresponding to the newly created Footer form part.

### Sample

```
var footer = form.newFooterPart(440);
```

### newFormMethod

JSMETHOD **newFormMethod**(code)

Creates a new form JSMETHOD - based on the specified code.

### Parameters

{String} code – the specified code for the new method

### Returns

JSMETHOD – a new JSMETHOD object for this form

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var method = form.newFormMethod('function aMethod(event){application.output("Hello world!");}');
    var button = myListViewForm.newButton('Show message!', 50, 50, 100, 30, method);
    forms['newForm1'].controller.show();
```

### newFormVariable

JSVariable **newFormVariable**(name, type)

Creates a new form JSVariable - based on the name of the variable object and the number type, uses the SolutionModel JSVariable constants.

### Parameters

{String} name – the specified name of the variable

{Number} type – the specified type of the variable (see Solution Model -> JSVariable node constants)

## Returns

[JSVariable](#) – a JSVariable object

## Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var variable = form.newFormVariable('myVar', JSVariable.TEXT);
variable.defaultValue = "'This is a default value (with tripple quotes)!'";
var field = form.newField(variable, JSField.TEXT_FIELD, 100, 100, 200, 200);
forms['newForm1'].controller.show();
```

## newHeaderPart

[JSPart](#) **newHeaderPart()**

Creates a new Header part on the form.

## Returns

[JSPart](#) – A JSPart instance corresponding to the newly created Header form part.

## Sample

```
var header = form.newHeaderPart(80);
```

## newHtmlArea

[JSField](#) **newHtmlArea**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of HTML\_AREA - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

## Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{[Number](#)} x – the horizontal "x" position of the JSField object in pixels

{[Number](#)} y – the vertical "y" position of the JSField object in pixels

{[Number](#)} width – the width of the JSField object in pixels

{[Number](#)} height – the height of the JSField object in pixels

## Returns

[JSField](#) – a JSField object on the form with the displayType of HTML\_AREA

## Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var textProvider = form.newFormVariable('myVar', JSVariable.TEXT);
textProvider.defaultValue = "'This is a tripple quotted text!'";
var htmlArea = myListViewForm.newHtmlArea(textProvider, 100, 100, 100, 100);
forms['newForm1'].controller.show();
```

## newImageMedia

[JSField](#) **newImageMedia**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of IMAGE\_MEDIA - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

## Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{[Number](#)} x – the horizontal "x" position of the JSField object in pixels

{[Number](#)} y – the vertical "y" position of the JSField object in pixels

{[Number](#)} width – the width of the JSField object in pixels

{[Number](#)} height – the height of the JSField object in pixels

## Returns

[JSField](#) – a new JSField object on the form with the displayType of IMAGE\_MEDIA

## Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var myMediaVar = form.newFormVariable("media", JSVariable.MEDIA);
var imageMedia = form.newImageMedia(myMediaVar, 100, 100, 200, 200);
forms['newForm1'].controller.show();
```

## newLabel

[JSLabel](#) **newLabel**(text, x, y, width, height, [jsmethod])

Creates a new JSLabel object on the form - including the text of the label, the "x" and "y" position of the label object in pixels, the width and height of the label object in pixels and the optional JSMETHOD action such as the method for an onAction event.

### Parameters

text – the specified text of the label object

{Number} x – the horizontal "x" position of the label object in pixels

{Number} y – the vertical "y" position of the label object in pixels

{Number} width – the width of the label object in pixels

{Number} height – the height of the label object in pixels

[jsmethod] – the event action JSMETHOD of the label object

### Returns

**JSLabel** – a JSLabel object

### Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
    var label = form.newLabel('The text on the label', 140, 140, 50, 20);
    forms['newForm1'].controller.show();
```

newLeadingGrandSummaryPart

**JSPart newLeadingGrandSummaryPart()**

Creates a new Leading Grand Summary part on the form.

### Returns

**JSPart** – A JSPart instance corresponding to the newly created Leading Grand Summary form part.

### Sample

```
var leadingGrandSummary = form.newLeadingGrandSummaryPart(120);
```

newLeadingSubSummaryPart

**JSPart newLeadingSubSummaryPart()**

Creates a new Leading Subsummary part on the form.

### Returns

**JSPart** – A JSPart instance corresponding to the newly created Leading Subsummary form part.

### Sample

```
var leadingSubsummary = form.newLeadingSubSummaryPart(160);
```

newPart

**JSPart newPart(type, height)**

Creates a new part on the form. The type of the new part (use one of the JSPart constants) and its height must be specified.

### Parameters

{Number} type – The type of the new part.

{Number} height – The height of the new part

### Returns

**JSPart** – A JSPart instance corresponding to the newly created form part.

### Sample

```
var form = solutionModel.newForm('myForm', 'example_data', 'my_table', null, false, 1200, 800);
var header = form.newPart(JSPart.HEADER, 100);
header.background = 'yellow';
var body = form.newPart(JSPart.BODY, 700);
body.background = 'green';
var footer = form.newPart(JSPart.FOOTER, 800);
footer.background = 'orange';
```

newPassword

**JSField newPassword(dataprovidername/jsvariable, x, y, width, height)**

Creates a new JSField object on the form with the displayType of PASSWORD - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{Number} x – the horizontal "x" position of the JSField object in pixels

{Number} y – the vertical "y" position of the JSField object in pixels

{Number} width – the width of the JSField object in pixels

{Number} height – the height of the JSField object in pixels

### Returns

**JSField** – a new JSField object on the form with the displayType of PASSWORD

## Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var pass = form.newPassword(globals.aVariable, 100, 100, 70, 30);
forms['newForm1'].controller.show();
```

## newPortal

**JSField newPortal**(name, relation, x, y, width, height)

Creates a new JSField object on the form - including the name of the JSField object; the relation the JSField object is based on, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

**{String}** name – the specified name of the JSField object  
**{Object}** relation – the relation of the JSField object  
**{Number}** x – the horizontal "x" position of the JSField object in pixels  
**{Number}** y – the vertical "y" position of the JSField object in pixels  
**{Number}** width – the width of the JSField object in pixels  
**{Number}** height – the height of the JSField object in pixels

### Returns

**JSField** – a JSField object

## Sample

```
var form = solutionModel.newForm('newForm1', 'server1', 'table1', null, true, 800, 600);
var relation = solutionModel.newRelation('parentToChild', 'server1', 'table1', 'server2', 'table2',
JSRelation.INNER_JOIN);
relation.newRelationItem('another_parent_table_id', '=', 'another_child_table_parent_id');
var portal = form.newPortal('portal', relation, 200, 200, 300, 300);
portal.newField('someColumn', JSField.TEXT_FIELD, 200, 200, 120);
forms['newForm1'].controller.show();
```

## newRadios

**JSField newRadios**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of RADIOS (radio buttons) - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object  
**{Number}** x – the horizontal "x" position of the JSField object in pixels  
**{Number}** y – the vertical "y" position of the JSField object in pixels  
**{Number}** width – the width of the JSField object in pixels  
**{Number}** height – the height of the JSField object in pixels

### Returns

**JSField** – a JSField object with the displayType of RADIOS (radio buttons)

## Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "value1\nvalue2\nvalue3";
var radios = form.newRadios('columnDataProvider', 100, 100, 200, 200);
radios.valuelist = vlist;
```

## newRtfArea

**JSField newRtfArea**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of RTF\_AREA (enables more than one line of text to be displayed in a field) - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object  
**{Number}** x – the horizontal "x" position of the JSField object in pixels  
**{Number}** y – the vertical "y" position of the JSField object in pixels  
**{Number}** width – the width of the JSField object in pixels  
**{Number}** height – the height of the JSField object in pixels

### Returns

**JSField** – a JSField object with the displayType of RTF\_AREA

## Sample

```
var form = solutionModel.newForm('newForm1','myServer','myTable',null,true,800,600);
    var rtf_area = form.newRtfArea('columnDataProvider',100,100,100,100);
    forms['newForm1'].controller.show();
```

newTabPanel

**JSTabPanel** **newTabPanel**(name, x, y, width, height)

Creates a new JSTabPanel object on the form - including the name of the JSTabPanel object, the "x" and "y" position of the JSTabPanel object in pixels, as well as the width and height of the JSTabPanel object in pixels.

### Parameters

{String} name – the specified name of the JSTabPanel object  
{Number} x – the horizontal "x" position of the JSTabPanel object in pixels  
{Number} y – the vertical "y" position of the JSTabPanel object in pixels  
{Number} width – the width of the JSTabPanel object in pixels  
{Number} height – the height of the JSTabPanel object in pixels

### Returns

**JSTabPanel** – a JSTabPanel object

## Sample

```
var form = solutionModel.newForm('parentForm','theServer','parent_table',null,false,640,480);
    var childOne = solutionModel.newForm('childOne','theServer','child_table',null,false,400,300);
    childOne.newField('child_table_text', JSField.TEXT_FIELD,10,10,100,20);
    var parentToChild = solutionModel.newRelation
('parentToChild','theServer','parent_table','theServer','child_table',JSRelation.INNER_JOIN);
    parentToChild.newRelationItem('parent_table_id','=', 'child_table_parent_id');
    var childTwo = solutionModel.newForm('childTwo','theServer','my_table',null,false,400,300);
    childTwo.newField('my_table_image', JSField.IMAGE_MEDIA,10,10,100,100);
    var tabPanel = form.newTabPanel('tabs',10,10,620,460);
    tabPanel.newTab('tab1','Child One',childOne,parentToChild);
    tabPanel.newTab('tab2','Child Two',childTwo);
    forms['parentForm'].controller.show();
```

newTextArea

**JSField** **newTextArea**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of TEXT\_AREA - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object  
{Number} x – the horizontal "x" position of the JSTabPanel object in pixels  
{Number} y – the vertical "y" position of the JSTabPanel object in pixels  
{Number} width – the width of the JSTabPanel object in pixels  
{Number} height – the height of the JSTabPanel object in pixels

### Returns

**JSField** – a JSField object with the displayType of TEXT\_AREA

## Sample

```
var form = solutionModel.newForm('newForm1','myServer','myTable',null,true,800,600);
    var globalVar = solutionModel.newGlobalVariable('myGlobal',JSVariable.TEXT);
    globalVar.defaultValue = "'Type your text in here'";
    var textArea = form.newTextArea(globalVar,100,100,300,150);
    forms['newForm1'].controller.show();
```

newTextField

**JSField** **newTextField**(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of TEXT\_FIELD - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

### Parameters

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object  
{Number} x – the horizontal "x" position of the JSfield object in pixels  
{Number} y – the vertical "y" position of the JSfield object in pixels  
{Number} width – the width of the JSfield object in pixels  
{Number} height – the height of the JSfield object in pixels

### Returns

**JSField** – a JSField object with the displayType of TEXT\_FIELD

### Sample

```
var form = solutionModel.newForm('newForm1','myServer','myTable',null,true,800,600);
    //choose the dataprovider or jsvariable you want for the Text Field
    var x = null;
    //global jsvariable as the dataprovider
    //x = solutionModel.newGlobalVariable('myGlobal',JSVariable.TEXT);
    //x.defaultValue = "'Text from a global variable'";
    //or a form jsvariable as the dataprovider
    //x = form.newFormVariable('myFormVar',JSVariable.TEXT);
    //x.defaultValue = "'Text from a form variable'";
    var textField = form.newTextField(x,100,100,200,50);
    //or a column data provider as the dataprovider
    //textField.dataProviderID = columnTextDataProvider;
    forms['newForm1'].controller.show();
```

#### newTitleFooterPart

##### **JSPart** newTitleFooterPart()

Creates a new Title Footer part on the form.

##### **Returns**

**JSPart** – A JSPart instance corresponding to the newly created Title Footer form part.

### Sample

```
var titleFooter = form.newTitleFooterPart(500);
```

#### newTitleHeaderPart

##### **JSPart** newTitleHeaderPart()

Creates a new Title Header part on the form.

##### **Returns**

**JSPart** – A JSPart instance corresponding to the newly created Title Header form part.

### Sample

```
var titleHeader = form.newTitleHeaderPart(40);
```

#### newTrailingGrandSummaryPart

##### **JSPart** newTrailingGrandSummaryPart()

Creates a new Trailing Grand Summary part on the form.

##### **Returns**

**JSPart** – A JSPart instance corresponding to the newly created Trailing Grand Summary form part.

### Sample

```
var trailingGrandSummary = form.newTrailingGrandSummaryPart(400);
```

#### newTrailingSubSummaryPart

##### **JSPart** newTrailingSubSummaryPart()

Creates a new Trailing Subsummary part on the form.

##### **Returns**

**JSPart** – A JSPart instance corresponding to the newly created Trailing Subsummary form part.

### Sample

```
var trailingSubsummary = form.newTrailingSubSummaryPart(360);
```

#### newTypeAhead

##### **JSField** newTypeAhead(dataprovidername/jsvariable, x, y, width, height)

Creates a new JSField object on the form with the displayType of TYPE\_AHEAD - including the dataprovider/JSVariable of the JSField object, the "x" and "y" position of the JSField object in pixels, as well as the width and height of the JSField object in pixels.

##### **Parameters**

dataprovidername/jsvariable – the specified dataprovider name/JSVariable of the JSField object

{**Number**} x – the horizontal "x" position of the JSField object in pixels

{**Number**} y – the vertical "y" position of the JSField object in pixels

{**Number**} width – the width of the JSField object in pixels

{**Number**} height – the height of the JSField object in pixels

**Returns**

[JSField](#) – a JSField object with the displayType of TYPE\_AHEAD

**Sample**

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var vlist = solutionModel.newValueList('options', JSValueList.CUSTOM_VALUES);
vlist.customValues = "value1\nvalue2\nvalue3";
var typeAhead = form.newTypeAhead(columnTextDataProvider, 100, 100, 300, 200);
typeAhead.valueList = vlist;
forms['newForm1'].controller.show();
```

**removeButton**

**Boolean** **removeButton**(name)

Removes a JSButton that has the specified name. Returns true if removal was successful, false otherwise.

**Parameters**

**{String}** name – the specified name of the JSButton to be removed

**Returns**

**Boolean** – true if the JSButton has been removed; false otherwise

**Sample**

```
var form = solutionModel.newForm('newFormX', 'myServer', 'myTable', null, true, 800, 600);
var b1 = form.newButton('This is button1', 100, 100, 200, 50, null);
b1.name = 'b1';
var jsmethod = form.newFormMethod("function removeMe(event) { var form = solutionModel.getForm('newFormX'); if
(form.removeButton('b1') == true) application.output('Button has been removed ok'); else application.output
('Button could not be deleted'); forms['newFormX'].controller.recreateUI();});");
var b2 = form.newButton('Click here to remove button1', 100, 230, 200, 50, jsmethod);
b2.name = 'b2';
forms['newFormX'].controller.show();
```

**removeComponent**

**Boolean** **removeComponent**(name)

Removes a component (JSLabel, JSButton, JSField, JSPortal, JSTabpanel) that has the given name. It is the same as calling "if(!removeLabel(name) && !removeButton(name) ....)".

Returns true if removal was successful, false otherwise.

**Parameters**

**{String}** name – the specified name of the component to be deleted

**Returns**

**Boolean** – true if component has been successfully deleted; false otherwise

### Sample

```
var form = solutionModel.newForm('newFormX', 'myServer', 'parentTable', null, true, 1000, 750);
var jsbutton = form.newButton('JSButton to delete', 100, 100, 200, 50, null);
jsbutton.name = 'jsb';
var jslabel = form.newLabel('JSLabel to delete', 100, 200, 200, 50, null);
jslabel.name = 'jsl';
jslabel.transparent = false;
jslabel.background = 'green';
var jsfield = form.newField('globals.myGlobalVariable', JSField.TEXT_FIELD, 100, 300, 200, 50);
jsfield.name = 'jsf';
var relation = solutionModel.newRelation('parentToChild', 'myServer', 'parentTable', 'myServer', 'childTable',
JSRelation.INNER_JOIN);
relation.newRelationItem('parent_table_id', '=', 'child_table_id');
var jsportal = form.newPortal('jsp', relation, 100, 400, 300, 300);
jsportal.newField('child_table_id', JSField.TEXT_FIELD, 200, 200, 120);
var childOne = solutionModel.newForm('childOne', 'myServer', 'childTable', null, false, 400, 300);
childOne.newField('child_table_id', JSField.TEXT_FIELD, 10, 10, 100, 20);
var childTwo = solutionModel.newForm('childTwo', 'myServer', 'someOtherTable', null, false, 400, 300);
childTwo.newField('some_table_id', JSField.TEXT_FIELD, 10, 10, 100, 100);
var jstabpanel = form.newTabPanel('jst', 450, 30, 620, 460);
jstabpanel.newTab('tab1', 'Child One', childOne, relation);
jstabpanel.newTab('tab2', 'Child Two', childTwo);
var jsmethod = form.newFormMethod("function removeMe(event) { var form = solutionModel.getForm('newFormX');\n
if ((form.removeComponent('jsb') == true) && (form.removeComponent('jsl') == true) && (form.removeComponent
('jsf') == true) && (form.removeComponent('jsp') == true) & (form.removeComponent('jst') == true)) application.
output('Components removed ok'); else application.output('Some component(s) could not be deleted'); forms
['newFormX'].controller.recreateUI();}");
var removerButton = form.newButton('Click here to remove form components', 450, 500, 250, 50, jsmethod);
removerButton.name = 'remover';
forms['newFormX'].controller.show();
```

### removeField

**Boolean** **removeField**(name)

Removes a JSField that has the given name. Returns true if removal was successful, false otherwise.

#### Parameters

{[String](#)} name – the specified name of the JSField to remove

#### Returns

**Boolean** – true is the JSField has been successfully removed; false otherwise

### Sample

```
var form = solutionModel.newForm('newFormX', 'myServer', 'myTable', null, true, 800, 600);
var jsfield = form.newField(globals.myGlobalVariable, JSField.TEXT_FIELD, 100, 300, 200, 50);
jsfield.name = 'jsf';
var jsmethod = form.newFormMethod("function removeMe(event) { var form = solutionModel.getForm('newFormX');\n
if (form.removeComponent('jsf') == true) application.output('Field has been removed ok'); else application.
output('Field could not be deleted'); forms['newFormX'].controller.recreateUI();}");
var removerButton = form.newButton('Click here to remove the field', 450, 500, 250, 50, jsmethod);
removerButton.name = 'remover';
forms['newFormX'].controller.show();
```

### removeLabel

**Boolean** **removeLabel**(name)

Removes a JSLabel that has the given name. Returns true if removal successful, false otherwise

#### Parameters

{[String](#)} name – the specified name of the JSLabel to be removed

#### Returns

**Boolean** – true if the JSLabel with the given name has successfully been removed; false otherwise

### Sample

```
var form = solutionModel.newForm('newFormX', 'myServer', 'myTable', null, true, 1000, 750);
var jslabel = form.newLabel('JSLabel to delete', 100, 200, 200, 50, null);
jslabel.name = 'jsl';
jslabel.transparent = false;
jslabel.background = 'green';
var jsmethod = form.newFormMethod("function removeMe(event) { var form = solutionModel.getForm('newFormX'); if
(form.removeComponent('jsl') == true) application.output('Label has been removed'); else application.output
('Label could not be deleted'); forms['newFormX'].controller.recreateUI();}");
var removerButton = form.newButton('Click here to remove the green label', 450, 500, 250, 50, jsmethod);
removerButton.name = 'remover';
forms['newFormX'].controller.show();
```

### removePart

**Boolean** **removePart**(type, [height])

Removes a JSPart of the given type. Use the height parameter if you want to remove one of multiple subsummary parts.

#### Parameters

{**Number**} type – The type of the part that should be removed.

{**Number**} [height] – The height of the part that should be removed. Use this parameter when you want to remove one of multiple Leading/Trailing Subsummary parts.

#### Returns

**Boolean** – True if the part is successfully removed, false otherwise.

### Sample

```
form.removePart(JSPart.HEADER);
form.removePart(JSPart.LEADING_SUBSUMMARY, 160);
```

### removePortal

**Boolean** **removePortal**(name)

Removes a JSPortal that has the given name. Returns true if removal was successful, false otherwise.

#### Parameters

{**String**} name – the specified name of the JSPortal to be removed

#### Returns

**Boolean** – true if the JSPortal has successfully been removed; false otherwise

### Sample

```
var form = solutionModel.newForm('newFormX', 'myServer', 'myTable', null, true, 800, 600);
var relation = solutionModel.newRelation('parentToChild', 'myServer', 'myTable', 'myServer', 'myOtherTable',
JSRelation.INNER_JOIN);
relation.newRelationItem('parent_table_id', '=', 'child_table_id');
var jsportal = form.newPortal('jsp', relation, 100, 400, 300, 300);
jsportal.newField('child_table_id', JSField.TEXT_FIELD, 200, 200, 120);
var jsmethod = form.newFormMethod("function removeMe(event) { var form = solutionModel.getForm
('newFormX');\n if (form.removeComponent('jsp') == true) application.output('Portal removed ok'); else
application.output('Portal could not be deleted'); forms['newFormX'].controller.recreateUI();}");
var removerButton = form.newButton('Click here to remove the portal', 450, 500, 250, 50, jsmethod);
removerButton.name = 'remover';
forms['newFormX'].controller.show();
```

### removeTabPanel

**Boolean** **removeTabPanel**(name)

Removes a JSTabPanel that has the given name. Returns true if removal was successful, false otherwise.

#### Parameters

{**String**} name – the specified name of the JSTabPanel to be removed

#### Returns

**Boolean** – true is the JSTabPanel has been successfully removed, false otherwise

## Sample

```
var form = solutionModel.newForm('newFormX', 'myServer', 'parentTable', null, false, 800, 600);
var childOne = solutionModel.newForm('childOne', 'myServer', 'childTable', null, false, 400, 300);
childOne.newField('child_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
var parentToChild = solutionModel.newRelation('parentToChild', 'myServer', 'parentTable', 'myServer', 'childTable',
JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id', '=', 'child_table_id');
var childTwo = solutionModel.newForm('childTwo', 'myServer', 'anotherTable', null, false, 400, 300);
childTwo.newField('columnDataProvider', JSField.TEXT_FIELD, 10, 10, 100, 100);
var tabPanel = form.newTabPanel('jst', 10, 10, 620, 460);
tabPanel.newTab('tab1', 'Child One', childOne, parentToChild);
tabPanel.newTab('tab2', 'Child Two', childTwo);
var jsmethod = form.newFormMethod("function removeMe(event) { var form = solutionModel.getForm('newFormX');\n
if (form.removeComponent('jst') == true)\n application.output('TabPanel has been removed ok');\n else\n
application.output('TabPanel could not be deleted');\n forms['newFormX'].controller.recreateUI();\n}");
var removerButton = form.newButton('Click here to remove the tab panel', 450, 500, 250, 50, jsmethod);
removerButton.name = 'remover';
forms['newFormX'].controller.show();
```