JSForm

Apr 05, 2024 07:44

Supported Clients

SmartClient WebClient NGClient MobileClient

Constants Summary

String	EMPTY_FOUNDSET	Constant used for form namedFoundset property.
Number	LIST_VIEW	The constants to set or get the view property of a JSForm.
Number	LOCKED_LIST_VIEW	The constants to set or get the view property of a JSForm.
Number	LOCKED_RECORD_VIEW	The constants to set or get the view property of a JSForm.
Number	LOCKED_TABLE_VIEW	The constants to set or get the view property of a JSForm.
Number	RECORD_VIEW	The constants to set or get the view property of a JSForm.
Number	SELECTION_MODE_DEFAULT	Constant used for form selectionMode property.
Number	SELECTION_MODE_MULTI	Constant used for form selectionMode property.
Number	SELECTION_MODE_SINGLE	Constant used for form selectionMode property.
String	SEPARATE_FOUNDSET	Constant used for form namedFoundset property.

Property Summary

String	borderType	The type, color and style of border of the component.
String	dataSource	The names of the database server and table that this form is linked to.
String	defaultPageFormat	The default page format for the form.
Number	encapsulation	Get or set the encapsulation level for the form.
JSForm	extendsForm	A JSForm instance representing the super form of this form, if this form has a super form.
String	initialSort	The default sort order only when the form loads.
String	name	The name of the form.
String	namedFoundSet	Property that tells the form to use a named foundset instead of the default foundset.
Object	navigator	The navigator (previously named "controller") that is used to control/navigate to the form.
Boolean	ngReadOnlyMode	Get or set the ngReadonlyMode for the form.
JSMethod	onDeleteAllRecordsCmd	The method that overrides the Servoy menu item Select > Delete All.
JSMethod	onDeleteRecordCmd	The method that overrides the Servoy menu item Select > Delete Record (or keyboard shortcut).
JSMethod	onDrag	The method that is triggered when (non Design Mode) dragging occurs.
JSMethod	onDragEnd	The method that is triggered when (non Design Mode) dragging end occurs.
JSMethod	onDragOver	The method that is triggered when (non Design Mode) dragging over a component occurs.
JSMethod	onDrop	The method that is triggered when (non Design Mode) dropping occurs.
JSMethod	onDuplicateRecordCmd	The method that overrides the Servoy menu item Select > Duplicate Record (or keyboard shortcut).
JSMethod	onElementDataChange	Method that is executed when the data in one of the form's component is successfully changed and the onDataChange callback from the component does not exist or exists and returned true
JSMethod	onElementFocusGained	The method that is triggered when focus is gained by a component inside the form.
JSMethod	onElementFocusLost	The method that gets triggered when focus is lost by a component inside the form.
JSMethod	onFindCmd	The method that overrides the Servoy menu item Select > Find (or keyboard shortcut) in Data (ready) mode.
JSMethod	onHide	This method is triggered when the form gets hidden.
JSMethod	onInvertRecordsCmd	The method that overrides the Servoy menu item Select > Invert Records.
JSMethod	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.
JSMethod	onNewRecordCmd	The method that overrides the Servoy menu item Select > New Record (or keyboard shortcut).
JSMethod	onNextRecordCmd	The method that overrides the Servoy menu item Select > Next Record.
JSMethod	onOmitRecordCmd	The method that overrides the Servoy menu item Select > Omit Record.
JSMethod	onPreviousRecordCmd	The method that overrides the Servoy menu item Select > Previous Record.
JSMethod	onPrintPreviewCmd	The method that overrides the Servoy menu item File > Print Preview.
JSMethod	onRecordEditStart	The method that is triggered when a user clicks into a column on the form.
JSMethod	onRecordEditStop	The method that is triggered when a record is being saved.
JSMethod	onRecordSelection	The method that is triggered each time a record is selected.
JSMethod	onRender	The method that is executed when the component is rendered.
JSMethod	onResize	The method that gets triggered when resize occurs.
JSMethod	onSearchCmd	The method that overrides the Servoy menu item Select > Search (or keyboard shortcut) in Find mode.
JSMethod	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.
JSMethod	onShowAllRecordsCmd	The method that overrides the Servoy menu item Select > Show All (or keyboard shortcut).
JSMethod	onShowOmittedRecordsCmd	The method that overrides the Servoy menu item Select > Show Omitted Records.

JSMethod	onSortCmd	The method that overrides the Servoy menu item Select > Sort.
JSMethod	onUnLoad	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.
Boolean	responsiveLayout	Returns true if this form is in responsive mode
Number	scrollbars	Scrollbar options for the vertical and horizontal scrollbars.
Number	selectionMode	Returns the value of the form's selectionMode property.
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.
Boolean	useCssPosition	Get or set the positioning (either use anchoring or use css position) for the form.
Number	view	The default form view mode.
Number	width	The width of the form in pixels.

Methods Summary

JSLayoutContainer	findLayoutContainer(name)	Returns a JSLayoutContainer that has the given name throughout the whole form hierarchy.
JSComponent	findWebComponent(name)	Returns a JSWebComponent that has the given name through the whole hierarchy of JSLayoutContainers
JSComponent	getBean(name)	Returns a JSBean that has the given name.
Array	getBeans()	Returns all JSBeans of this form.
Array	getBeans(returnInheritedElements)	Returns all JSBeans of this form.
JSPart	getBodyPart()	Retrieves the Body part of the form.
JSComponent	getButton(name)	Returns a JSButton that has the given name.
Array	getButtons()	Returns all JSButtons of this form, including the ones without a name.
Array	getButtons(returnInheritedElements)	Returns all JSButtons of this form, including the ones without a name.
String	getComment()	Returns the comment of this container.
JSComponent	getComponent(name)	Returns a JSComponent that has the given name; if found it will be a JSField, JSLabel, JSButton, JSPortal, JSBean, JSWebComponent or JSTabPanel.
Array	getComponents()	Returns a array of all the JSComponents that a form has; they are of type JSField, JSLabel,JSButton,JSPortal,JSBean, JSWebComponents or JSTabPanel.
Array	getComponents(returnInheritedElements)	Returns a array of all the JSComponents that a form has; they are of type JSField, JSLabel,JSButton,JSPortal,JSBean, JSWebComponent or JSTabPanel.
Object	getDesignTimeProperty(key)	Get a design-time property of a form.
Array	getDesignTimePropertyNames()	Get the design-time properties of a form.
JSField	getField(name)	The field with the specified name.
Array	getFields()	Returns all JSField objects of this form, including the ones without a name.
Array	getFields(returnInheritedElements)	Returns all JSField objects of this form, including the ones without a name.
JSPart	getFooterPart()	Retrieves the Footer part of the form.
JSPart	getHeaderPart()	Retrieves the Header part of the form.
JSComponent	getLabel(name)	Returns a JSLabel that has the given name.
Array	getLabels()	Returns all JSLabels of this form (not including its super form), including the ones without a name.
Array	getLabels(returnInheritedElements)	Returns all JSLabels of this form (optionally including it super forms labels), including the ones without a name.
JSLayoutContainer	getLayoutContainer(name)	Returns a JSLayoutContainer that has the given name of this container.
Array	getLayoutContainers()	Returns all JSLayoutContainers objects of this container.
Array	getLayoutContainers(returnInheritedElements)	Returns all JSLayoutContainers objects of this container
JSPart	getLeadingGrandSummaryPart()	Retrieves the Leading Grand Summary part of the form.
Array	getLeadingSubSummaryParts()	Gets an array of the Leading Subsummary parts of the form, ordered by their height from top == 0 to bottom.
JSMethod	getMethod(name)	Gets an existing form method for the given name.
Array	getMethods()	Returns all existing form methods for this form.
Array	getMethods(returnInheritedElements)	Returns all existing form methods for this form.
JSPart	getPart(type)	Gets a part of the form from the given type (see JSPart constants).
JSPart	getPart(type, height)	Gets a part of the form from the given type (see JSPart constants).
Number	getPartYOffset(type)	Returns the Y offset of a given part (see JSPart) of the form.
Number	getPartYOffset(type, height)	Returns the Y offset of a given part (see JSPart) of the form.
Array	getParts()	Gets all the parts from the form (not including the parts of the parent form), ordered by there height (lowerbound) property, from top == 0 to bottom.
Array	getParts(returnInheritedElements)	Gets all the parts from the form (optionally also from the parent form), ordered by there height (lowerbound) property, from top == 0 to bottom.
JSComponent	getPortal(name)	Returns a JSPortal that has the given name.
Array	getPortals()	Returns all JSPortal objects of this form (not including the ones from the parent form), including the ones without a name.

Array	getPortals(returnInheritedElements)	Returns all JSPortal objects of this form (optionally also the ones from the parent form), including the ones without a name.
JSComponent	getTabPanel(name)	Returns a JSTabPanel that has the given name.
Array	getTabPanels()	Returns all JSTabPanels of this form (not including the ones from the parent form), including the ones without a name.
Array	getTabPanels(returnInheritedElements)	Returns all JSTabPanels of this form (optionally the ones from the parent 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.
Array	getTrailingSubSummaryParts()	Gets an array of the Trailing Subsummary parts of the form, ordered by their height from top == 0 to bottom.
UUID	getUUID()	Returns the UUID of this form.
JSVariable	getVariable(name)	Gets an existing form variable for the given name.
Array	getVariables()	An array consisting of all form variables for this form.
Array	getVariables(returnInheritedElements)	An array consisting of all form variables for this form.
JSComponent	getWebComponent(name)	Returns a JSWebComponent that has the given name that is a child of this layout container.
Array	getWebComponents()	Returns all JSWebComponents of this form/container.
Array	getWebComponents(returnInheritedElements)	Returns all JSWebComponents of this form/container.
JSComponent	newBean(name, className, x, y, width, height)	Creates a new JSBean object on the form - including the name of the JSBean object; the classname the JSBean object is based on, the "x" and "y" position of the JSBean object in pixels, as well as the width and height of the JSBean object in pixels.
JSComponent	newButton(txt, x, y, width, height, action)	Creates a new button on the form with the given text, place, size and JSMethod as the onAction event triggered action.
JSComponent	newButton(txt, action)	Creates a new button on the form with the given text and JSMethod as the onAction event triggered action.
JSField	newCalendar(dataprovider)	Creates a new JSField object on the form with the displayType of CALENDAR - including the dataprovider/JSVariable of the JSField object.
JSField	newCalendar(dataprovider, 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(dataprovider)	Creates a new JSField object on the form with the displayType of CHECK (checkbox) - including the dataprovider/JSVariable of the JSField object.
JSField	newCheck(dataprovider, 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(dataprovider)	Creates a new JSField object on the form with the displayType of COMBOBOX - including the dataprovider/JSVariable of the JSField object.
JSField	newComboBox(dataprovider, 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(dataprovider, type)	Creates a new JSField object on the form - including the dataprovider/JSVariable of the JSField object.
JSField	newField(dataprovider, type, 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(height)	Creates a new Footer part on the form.
JSPart	newHeaderPart(height)	Creates a new Header part on the form.
JSField	newHtmlArea(dataprovider)	Creates a new JSField object on the form with the displayType of HTML_AREA - including the dataprovider/JSVariable of the JSField object.
JSField	newHtmlArea(dataprovider, 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(dataprovider)	Creates a new JSField object on the form with the displayType of IMAGE_MEDIA - including the dataprovider/JSVariable of the JSField object.
JSField	newImageMedia(dataprovider, 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.
JSComponent	newLabel(txt)	Creates a new JSLabel object on the form - including the text of the label.
JSComponent	newLabel(txt, x, y, width, height)	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.
JSComponent	newLabel(txt, x, y, width, height, action)	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 a JSMethod action such as the method for an onAction event.
JSLayoutContainer	newLayoutContainer()	Create a new layout container as the last child of its parent container.
JSLayoutContainer	newLayoutContainer(position)	Create a new layout container.
JSLayoutContainer	newLayoutContainer(position, spec)	Create a new layout container.
JSLayoutContainer	newLayoutContainer(spec)	Create a new layout container as the last child in its parent container.
JSPart	newLeadingGrandSummaryPart(height)	Creates a new Leading Grand Summary part on the form.
JSPart	newLeadingSubSummaryPart(height)	Creates a new Leading Subsummary part on the form.

JSField	newListBox(dataprovider)	Creates a new JSField object on the form with the displayType of LISTBOX - including the dataprovider/JSVariable of the JSField object.
JSField	newListBox(dataprovider, x, y, width, height)	Creates a new JSField object on the form with the displayType of LISTBOX - 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.
JSMethod	newMethod(code)	Creates a new form JSMethod - based on the specified code.
JSField	newMultiSelectListBox(dataprovider)	Creates a new JSField object on the form with the displayType of MULTISELECT_LISTBOX - including the dataprovider/JSVariable of the JSField object.
JSField	newMultiSelectListBox(dataprovider, x, y, width, height)	Creates a new JSField object on the form with the displayType of MULTISELECT_LISTBOX - 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	newPart(type, height)	Creates a new part on the form.
JSField	newPassword(dataprovider)	Creates a new JSField object on the form with the displayType of PASSWORD - including the dataprovider/JSVariable of the JSField object.
JSField	newPassword(dataprovider, 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.
JSComponent	newPortal(name, relation)	Creates a new JSPortal object on the form - including the name of the JSPortal object and the relation the JSPortal object is based on.
JSComponent	newPortal(name, relation, 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	newRadios(dataprovider)	Creates a new JSField object on the form with the displayType of RADIOS (radio buttons) - including the dataprovider/JSVariable of the JSField object.
JSField	newRadios(dataprovider, 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.
JSField	newRtfArea(dataprovider, 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.
JSField	newSpinner(dataprovider)	Creates a new JSField object on the form with the displayType of SPINNER - including the dataprovider/JSVariable of the JSField object.
JSField	newSpinner(dataprovider, x, y, width, height)	Creates a new JSField object on the form with the displayType of SPINNER - 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.
JSComponent	newTabPanel(name)	Creates a new JSTabPanel object on the form - including the name of the JSTabPanel object.
JSComponent	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.
JSField	newTextArea(dataprovider)	Creates a new JSField object on the form with the displayType of TEXT_AREA - including the dataprovider/JSVariable of the JSField object.
JSField	newTextArea(dataprovider, 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	newTextField(dataprovider)	Creates a new JSField object on the form with the displayType of TEXT_FIELD - including the dataprovider/JSVariable of the JSField object.
JSField	newTextField(dataprovider, 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.
JSPart	newTitleFooterPart(height)	Creates a new Title Footer part on the form.
JSPart	newTitleHeaderPart(height)	Creates a new Title Header part on the form.
JSPart	newTrailingGrandSummaryPart(height)	Creates a new Trailing Grand Summary part on the form.
JSPart	newTrailingSubSummaryPart(height)	Creates a new Trailing Subsummary part on the form.
JSField	newTypeAhead(dataprovider)	Creates a new JSField object on the form with the displayType of TYPE_AHEAD - including the dataprovider/JSVariable of the JSField object.
JSField	newTypeAhead(dataprovider, 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.
JSVariable	newVariable(name, type)	Creates a new form JSVariable - based on the name of the variable object and the number type, uses the SolutionModel JSVariable constants.
JSVariable	newVariable(name, type, defaultValue)	Creates a new form JSVariable - based on the name of the variable object , the type and it's default value , uses the SolutionModel JSVariable constants.
JSComponent	newWebComponent(type)	Creates a new JSWebComponent (spec based component) object on the RESPONSIVE form.
JSComponent	newWebComponent(type, position)	Creates a new JSWebComponent (spec based component) object on the RESPONSIVE form.
JSComponent	newWebComponent(name, type)	Creates a new JSWebComponent (spec based component) object on a form.
JSComponent	newWebComponent(name, type, position)	Creates a new JSWebComponent (spec based component) object on the RESPONSIVE form.
JSComponent	newWebComponent(name, type, x, y, width, height)	Creates a new JSWebComponent (spec based component) object on the form.
Object	putDesignTimeProperty(key, value)	Set a design-time property of a form.
Boolean	removeBean(name)	Removes a JSBean that has the specified name.

Boolean	removeButton(name)	Removes a JSButton that has the specified name.
Boolean	removeComponent(name)	Removes a component (JSLabel, JSButton, JSField, JSPortal, JSBean, JSTabpanel, JSWebComponent) that has the given name.
Object	removeDesignTimeProperty(key)	Clear a design-time property of a form.
Boolean	removeField(name)	Removes a JSField that has the given name.
Boolean	removeLabel(name)	Removes a JSLabel that has the given name.
Boolean	removeMethod(name)	Removes a form JSMethod - based on the specified code.
Boolean	removePart(type)	Removes a JSPart of the given type.
Boolean	removePart(type, height)	Removes a JSPart of the given type.
Boolean	removePortal(name)	Removes a JSPortal that has the given name.
Boolean	removeTabPanel(name)	Removes a JSTabPanel that has the given name.
Boolean	removeVariable(name)	Removes a form JSVariable - based on the name of the variable object.
Boolean	removeWebComponent(name)	Removes a JSWebComponent that has the specified name.

Constants Details

EMPTY_FOUNDSET

Constant used for form namedFoundset property. The form that uses empty namedFoundset will initially have an empty (cleared) foundset.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// form with empty foundset
var frmEmpty = solutionModel.newForm('products_empty', 'example_data', 'products', null, true, 640, 480);
frmEmpty.newLabel("Empty FoundSet",10,10,200,20);
frmEmpty.newField('categoryid',JSField.TEXT_FIELD,10,40,200,20);
frmEmpty.newField('productname',JSField.TEXT_FIELD,10,70,200,20);
frmEmpty.namedFoundSet = JSForm.EMPTY_FOUNDSET;
```

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

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

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

var myRecordViewForm = solutionModel.newForm('newForm5', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

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

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

var myRecordViewForm = solutionModel.newForm('newForm5', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

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

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

var myRecordViewForm = solutionModel.newForm('newForm5', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

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

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

var myRecordViewForm = solutionModel.newForm('newForm5', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

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

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

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

SELECTION_MODE_DEFAULT

Constant used for form selectionMode property. It means that the foundset's multiSelect property is used.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.getForm('my_form_name');
myForm.selectionMode = JSForm.SELECTION_MODE_DEFAULT;
```

SELECTION_MODE_MULTI

Constant used for form selectionMode property. It means that the form will force multiSelect to true on the foundset it uses.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.getForm('my_form_name');
myForm.selectionMode = JSForm.SELECTION_MODE_MULTI;
```

SELECTION_MODE_SINGLE

Constant used for form selectionMode property. It means that the form will force multiSelect to false on the foundset it uses.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.getForm('my_form_name');
myForm.selectionMode = JSForm.SELECTION_MODE_SINGLE;
```

SEPARATE_FOUNDSET

Constant used for form namedFoundset property. The form that uses a separate namedFoundset will initially have an separate (not shared with other forms) foundset.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// form with separate foundset
var frmSeparate = solutionModel.newForm('products_separate', 'example_data', 'products', null, true, 640, 480);
frmSeparate.newLabel("Separate FoundSet",10,10,200,20);
frmSeparate.newField('categoryid',JSField.TEXT_FIELD,10,40,200,20);
frmSeparate.newField('productname',JSField.TEXT_FIELD,10,70,200,20);
frmSeparate.namedFoundSet = JSForm.SEPARATE_FOUNDSET;
forms['products_separate'].controller.find();
forms['products_separate'].categoryid = '=2';
forms['products_separate'].controller.search();
```

Property Details

borderType

The type, color and style of border of the component.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//HINT: To know exactly the notation of this property set it in the designer and then read it once out through
the solution model.
var field = form.newField('my_table_text', JSField.TEXT_FIELD, 10, 10, 100, 20);
field.borderType = solutionModel.createLineBorder(1,'#ff0000');
```

dataSource

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

Returns

String

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

defaultPageFormat

The default page format for the form.

Returns

String

Supported Clients

SmartClient,WebClient

Sample

```
var form = solutionModel.getForm("someForm");
application.output(form.defaultPageFormat);
form.defaultPageFormat = solutionModel.createPageFormat(612,792,72,72,72,72,SM_ORIENTATION.PORTRAIT,SM_UNITS.
PIXELS)
```

encapsulation

Get or set the encapsulation level for the form.

Encapsulation is one of constants JSForm.DEFAULT_ENCAPSULATION, JSForm.PRIVATE_ENCAPSULATION, JSForm.
MODULE_PRIVATE_ENCAPSULATION,
JSForm.HIDE_DATAPROVIDERS_ENCAPSULATION, JSForm.HIDE_FOUNDSET_ENCAPSULATION, JSForm.
HIDE_CONTROLLER_ENCAPSULATION or JSForm.HIDE_ELEMENTS_ENCAPSULATION

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
myForm.encapsulation = JSForm.HIDE_CONTROLLER_ENCAPSULATION;
```

extendsForm

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

Returns

JSForm

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

name

The name of the form.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('myForm',myDatasource,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 JSForm.SEPARATE_FOUNDSET is specified the form will always create a copy of assigned foundset and
therefore become separated from other foundsets.
When JSForm.EMPTY_FOUNDSET, the form will have an initially empty foundset.

The namedFoundset can be based on a global relation; in this case namedFoundset is the relation's name.
You can also set the namedFoundset to a JSRelation object directly.
It will tell this form to initially load a global relation based foundset.
The global relation's foreign datasource must match the form's datasource.
Do not use relations named "empty" or "separate" to avoid confusions.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// form with separate foundset
var frmSeparate = solutionModel.newForm('products_separate', 'db:/example_data/products', null, true, 640, 480);
frmSeparate.newLabel("Separate FoundSet",10,10,200,20);
frmSeparate.newField('categoryid',JSField.TEXT_FIELD,10,40,200,20);
frmSeparate.newField('productname',JSField.TEXT_FIELD,10,70,200,20);
frmSeparate.namedFoundSet = JSForm.SEPARATE_FOUNDSET;
forms['products_separate'].controller.find();
forms['products_separate'].categoryid = '=2';
forms['products_separate'].controller.search();

// form with empty foundset
var frmEmpty = solutionModel.newForm('products_empty', 'db:/example_data/products', null, true, 640, 480);
frmEmpty.newLabel("Empty FoundSet",10,10,200,20);
frmEmpty.newField('categoryid',JSField.TEXT_FIELD,10,40,200,20);
frmEmpty.newField('productname',JSField.TEXT_FIELD,10,70,200,20);
frmEmpty.namedFoundSet = JSForm.EMPTY_FOUNDSET;

// form with an initial foundset based on a global relation
var frmGlobalRel = solutionModel.newForm("categories_related", solutionModel.getForm("categories"));
frmGlobalRel.namedFoundSet = "g2_to_category_name";

// form with an initial foundset based on a global relation
var frmGlobalRel = solutionModel.newForm("categories_related", solutionModel.getForm("categories"));
frmGlobalRel.namedFoundSet = solutionModel.getRelation("g1_to_categories");
```

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

Object

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var aForm = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
// you can also use SM_DEFAULTS.INGORE 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', myDatasource, 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);
```

ngReadOnlyMode

Get or set the ngReadonlyMode for the form. This flag is a performance optimization for tableview/listview to
show labels instead of editable fields.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
myForm.ngReadOnlyMode = true;
```

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

onDrag

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

Returns

JSMethod

Supported Clients

SmartClient,WebClient

Sample

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

onDragEnd

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

Returns

JSMethod

Supported Clients

SmartClient,WebClient

Sample

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

Supported Clients

SmartClient,WebClient

Sample

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

onDrop

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

Returns

JSMethod

Supported Clients

SmartClient,WebClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

onElementDataChange

Method that is executed when the data in one of the form's component is successfully changed and
the onDataChange callback from the component does not exist or exists and returned true

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
form.onElementDataChange = form.newMethod('function onElementDataChange(oldValue, newValue, event) { application.
output("Data changed from " + oldValue + " to " + newValue + " at " + event.getTimestamp()); }');
```

onElementFocusGained

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

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

onHide

This method is triggered when the form gets hidden.

Return value is DEPRECATED: false return value should no longer be used. In the past, if the onHide method returned false, the form hide could be prevented from happening
in some cases (for example, when using onHide with showFormInDialog, the form will not close by clicking the dialog close box (X)). But that lead to
unexpected situations when the form being hidden had visible nested children it it (tab panels, splits etc.) because only the current form would
decide if hide could be denied, and all other forms, even if they returned false in their on-hide, would not be able to block the hide if this form allowed it.
So those nested forms might think that they are still visible even though they are not.

Please use the new onBeforeHide method/handler instead if you want to prevent forms from hiding.

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
form.onOmitRecordCmd = form.newMethod('function onOmitRecordCmd(event) { application.output("onOmitRecordCmd
intercepted on " + event.getFormName()); }');
form.onShowOmittedRecordsCmd = form.newMethod('function onShowOmittedRecordsCmd(event) { application.output
("onShowOmittedRecordsCmd intercepted on " + event.getFormName()); }');
form.onInvertRecordsCmd = form.newMethod('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.
Also calls to loadRecords() should be done in the onShow method and not in the onLoad method
If you call loadRecords() in the onShow method, you may want to set the namedFoundSet property of the form to 'empty' to prevent the first default form query.
NOTE: the onLoad event bubbles down, meaning that the onLoad is first fired on the parent then on a tab in a tabpanel (and in tab of that tab panels if you are 3 deep)

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient

Sample

```
form.onPrintPreviewCmd = form.newMethod('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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
form.onRecordEditStart = form.newMethod('function onRecordEditStart(event) { application.output
("onRecordEditStart intercepted on " + event.getFormName()); }');
form.onRecordEditStop = form.newMethod('function onRecordEditStop(record, event) { application.output
("onRecordEditStop intercepted on " + event.getFormName() + ". record is: " + record); }');
form.onRecordSelection = form.newMethod('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).
When the method returns false (for example as part of a validation), the user cannot leave the record, for example in
a table view a user cannot move to another record when the callback returns false.

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
form.onRecordEditStart = form.newMethod('function onRecordEditStart(event) { application.output
("onRecordEditStart intercepted on " + event.getFormName()); }');
form.onRecordEditStop = form.newMethod('function onRecordEditStop(record, event) { application.output
("onRecordEditStop intercepted on " + event.getFormName() + ". record is: " + record); }');
form.onRecordSelection = form.newMethod('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.
NOTE: this will also fire if the selection goes to -1 because the foundset is cleared. So foundset.getSelectedRecord() can return null.

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

onRender

The method that is executed when the component is rendered.

Returns

JSMethod

Supported Clients

SmartClient,WebClient

Sample

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

onResize

The method that gets triggered when resize occurs.

Returns

JSMethod

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
form.onResize = form.newMethod('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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
form.onSortCmd = form.newMethod('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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Supported Clients

SmartClient,WebClient

Sample

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

responsiveLayout

Returns true if this form is in responsive mode

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.getForm('myform');
if (myForm.isResponsive()) {}
```

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

selectionMode

Returns the value of the form's selectionMode property.
Selection mode is applied when necessary to the foundset used by the form (through it's multiSelect property), even if the foundset changes.
If two or more forms with non-default and different selectionMode values share the same foundset, the visible one decides.
If two or more non-visible forms with non-default and different selectionMode values share the same foundset, one of them (always the same from a set of forms) decides.
If two or more visible forms with non-default and different selectionMode values share the same foundset, one of them (always the same from a set of forms) decides what the
foundset's selectionMode should be.

Can be one of SELECTION_MODE_DEFAULT, SELECTION_MODE_SINGLE or SELECTION_MODE_MULTI.

Returns

Number

Supported Clients

SmartClient,WebClient

Sample

```
var myForm = solutionModel.getForm('my_form_name');
if (myForm.selectionMode == JSForm.SELECTION_MODE_MULTI) myForm.selectionMode = JSForm.SELECTION_MODE_DEFAULT;
```

serverName

Get the server name used by this form.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('myForm',myDatasource,null,true,800,600);
form.serverName = 'anotherServerName';
var theServerName = form.serverName;
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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var aForm = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
var anotherForm= solutionModel.newForm('newForm2', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var aForm = solutionModel.newForm('newForm1', myDatasource, 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

Supported Clients

SmartClient,WebClient

Sample

```
var aForm = solutionModel.newForm('newForm1', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var aForm = solutionModel.newForm('newForm1', myDatasource, 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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.newForm('newForm', 'db:/a_server/a_table', '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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

useCssPosition

Get or set the positioning (either use anchoring or use css position) for the form.
This is only working for absolute layout forms in NGClient.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var myForm = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
myForm.useCssPosition = true;
```

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

width

The width of the form in pixels.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

Methods Details

findLayoutContainer(name)

Returns a JSLayoutContainer that has the given name throughout the whole form hierarchy.

Parameters

String name the specified name of the container

Returns

JSLayoutContainer a JSLayoutContainer object

Supported Clients

NGClient

Sample

```
var container = myForm.findLayoutContainer("row1");
application.output(container.name);
```

findWebComponent(name)

Returns a JSWebComponent that has the given name through the whole hierarchy of JSLayoutContainers

Parameters

String name the specified name of the web component

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var btn = myForm.findWebComponent("mycomponent");
application.output(mybean.typeName);
```

getBean(name)

Returns a JSBean that has the given name.

Parameters

String name the specified name of the bean

Returns

JSComponent a JSBean object

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var btn = myForm.getBean("mybean");
application.output(mybean.className);
```

getBeans()

Returns all JSBeans of this form.

Returns

Array the list of all JSBeans on this forms

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var beans = myForm.getBeans();
for (var b in beans)
{
        if (beans[b].name != null)
                application.output(beans[b].name);
}
```

getBeans(returnInheritedElements)

`Returns all JSBeans of this form.`

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array the list of all JSBeans on this forms

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var beans = myForm.getBeans();
for (var b in beans)
{
        if (beans[b].name != null)
                application.output(beans[b].name);
}
```

getBodyPart()

`Retrieves the Body part of the form.`

Returns

JSPart A JSPart instance corresponding to the Body part of the form.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getButton(name)

`Returns a JSButton that has the given name.`

Parameters

String name the specified name of the button

Returns

JSComponent a JSButton object

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getButtons()

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

Returns

Array the list of all JSButtons on this forms

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

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 ");
}
```

getButtons(returnInheritedElements)

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

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array the list of all JSButtons on this forms

Supported Clients

SmartClient,WebClient,NGClient

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 ");
}
```

getComment()

`Returns the comment of this container.`

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var comment = solutionModel.getForm("my_form").getComment();
application.output(comment);
```

getComponent(name)

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

Parameters

String name the specified name of the component

Returns

JSComponent a JSComponent object (might be a JSField, JSLabel, JSButton, JSPortal, JSBean, JSWebComponent or JSTabPanel)

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getComponents()

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

Returns

Array an array of all the JSComponents on the form.

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getComponents(returnInheritedElements)

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

Parameters

Boolean returnInheritedElements true to also return the elements from the parent form

Returns

Array an array of all the JSComponents on the form.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getDesignTimeProperty(key)

Get a design-time property of a form.

Parameters

String key the property name

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var frm = solutionModel.getForm('orders')
var prop = frm.getDesignTimeProperty('myprop')
```

getDesignTimePropertyNames()

Get the design-time properties of a form.

Returns

Array

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var frm = solutionModel.getForm('orders')
var propNames = frm.getDesignTimePropertyNames()
```

getField(name)

The field with the specified name.

Parameters

String name the specified name of the field

Returns

JSField a JSField object

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getFields()

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

Returns

Array all JSField objects of this form

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

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);
}
```

getFields(returnInheritedElements)

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

Parameters

Boolean returnInheritedElements true to also return the elements from the parent form

Returns

Array all JSField objects of this form

Supported Clients

SmartClient,WebClient,NGClient

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()

Retrieves the Footer part of the form.

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getHeaderPart()

Retrieves the Header part of the form.

Returns

JSPart A JSPart instance corresponding to the Header part of the form.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getLabel(name)

Returns a JSLabel that has the given name.

Parameters

String name the specified name of the label

Returns

JSComponent a JSLabel object (or null if the label with the specified name does not exist)

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getLabels()

Returns all JSLabels of this form (not including its super form), including the ones without a name.

Returns

Array all JSLabels on this form

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

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);
}
```

getLabels(returnInheritedElements)

Returns all JSLabels of this form (optionally including it super forms labels), including the ones without a name.

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array all JSLabels on this form

Supported Clients

SmartClient,WebClient,NGClient

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);
}
```

getLayoutContainer(name)

Returns a JSLayoutContainer that has the given name of this container.
Use findLayoutContainer() method to find a JSLayoutContainter through the hierarchy

Parameters

String name the specified name of the container

Returns

JSLayoutContainer a JSLayoutContainer object

Supported Clients

NGClient

Sample

```
var container = myForm.getLayoutContainer("row1");
application.output(container.name);
```

getLayoutContainers()

Returns all JSLayoutContainers objects of this container.
Does not return the inherited containers, use #getLayoutContainers(true) to get the inherited as well.

Returns

Array all JSLayoutContainers objects of this container

Supported Clients

NGClient

Sample

```
var frm = solutionModel.getForm("myForm");
var containers = frm.getLayoutContainers();
for (var c in containers)
{
		var fname = containers[c].name;
		application.output(fname);
}
```

getLayoutContainers(returnInheritedElements)

Returns all JSLayoutContainers objects of this container

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array all JSLayoutContainers objects of this container

Supported Clients

NGClient

Sample

```
var frm = solutionModel.getForm("myForm");
var containers = frm.getLayoutContainers();
for (var c in containers)
{
                var fname = containers[c].name;
                application.output(fname);
}
```

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.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getLeadingSubSummaryParts()

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

Returns

Array An array of JSPart instances corresponding to the Leading Subsummary parts of the form.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getMethod(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)

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getMethods()

Returns all existing form methods for this form.

Returns

Array all form methods for the form

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getMethods(returnInheritedElements)

Returns all existing form methods for this form.

Parameters

Boolean returnInheritedElements true to also return the elements from the parent form

Returns

Array all form methods for the form

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getPart(type)

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

Parameters

Number type The type of the part to retrieve.

Returns

JSPart A JSPart instance representing the retrieved form part.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getPart(type, height)

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getPartYOffset(type)

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.

Parameters

Number type The type of the part whose Y offset will be returned.

Returns

Number A number holding the Y offset of the specified form part.

Supported Clients

SmartClient,WebClient,NGClient

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('globals', 'test'));
```

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 height parameter 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.

Supported Clients

SmartClient,WebClient,NGClient

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('globals', 'test'));
```

getParts()

Gets all the parts from the form (not including the parts of the parent form), ordered by there height (lowerbound) property, from top == 0 to bottom.

Returns

Array An array of JSPart instances corresponding to the parts of the form.

Supported Clients

SmartClient,WebClient,NGClient

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());
}
```

getParts(returnInheritedElements)

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

Parameters

Boolean returnInheritedElements true to also return the parts from parent form

Returns

Array An array of JSPart instances corresponding to the parts of the form.

Supported Clients

SmartClient,WebClient,NGClient

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(name)

Returns a JSPortal that has the given name.

Parameters

String name the specified name of the portal

Returns

JSComponent a JSPortal object

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getPortals()

Returns all JSPortal objects of this form (not including the ones from the parent form), including the ones without a name.

Returns

Array an array of all JSPortal objects on this form

Supported Clients

SmartClient,WebClient,NGClient

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");
}
```

getPortals(returnInheritedElements)

Returns all JSPortal objects of this form (optionally also the ones from the parent form), including the ones without a name.

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array an array of all JSPortal objects on this form

Supported Clients

SmartClient,WebClient,NGClient

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(name)

Returns a JSTabPanel that has the given name.

Parameters

String name the specified name of the tabpanel

Returns

JSComponent a JSTabPanel object

Supported Clients

SmartClient,WebClient,NGClient

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()

Returns all JSTabPanels of this form (not including the ones from the parent form), including the ones without a name.

Returns

Array an array of all JSTabPanel objects on this form

Supported Clients

SmartClient,WebClient,NGClient

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");
}
```

getTabPanels(returnInheritedElements)

Returns all JSTabPanels of this form (optionally the ones from the parent form), including the ones without a name.

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array an array of all JSTabPanel objects on this form

Supported Clients

SmartClient,WebClient,NGClient

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()

Retrieves the Title Footer part of the form.

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getTitleHeaderPart()

Retrieves the Title Header part of the form.

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getTrailingSubSummaryParts()

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

Returns

Array An array of JSPart instances corresponding to the Trailing Subsummary parts of the form.

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getUUID()

Returns the UUID of this form.

Returns

UUID

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form_UUID = myForm.getUUID();
application.output(form_UUID.toString());
```

getVariable(name)

Gets an existing form variable for the given name.

Parameters

String name the specified name of the variable

Returns

JSVariable a JSVariable object

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getVariables()

An array consisting of all form variables for this form.

Returns

Array an array of all variables on this form

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

getVariables(returnInheritedElements)

An array consisting of all form variables for this form.

Parameters

Boolean returnInheritedElements true to also return the elements from the parent form

Returns

Array an array of all variables on this form

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

getWebComponent(name)

Returns a JSWebComponent that has the given name that is a child of this layout container.
Use findWebComponent() to find a webcomponent through the hierarchy

Parameters

String name the specified name of the web component

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var btn = myForm.getWebComponent("mycomponent");
application.output(mybean.typeName);
```

getWebComponents()

```
Returns all JSWebComponents of this form/container.
If this method is called on a form, then it will return all web components on that form.
If the form is responsive, it will return the web components from all the containers.
It does not return the inherited components, use #getWebComponents(true) to get the inherited as well.
```

Returns

Array the list of all JSWebComponent on this forms

Supported Clients

NGClient

Sample

```
var webComponents = myForm.getWebComponents();
for (var i in webComponents)
{
	if (webComponents[i].name != null)
		application.output(webComponents[i].name);
}
```

getWebComponents(returnInheritedElements)

```
Returns all JSWebComponents of this form/container.
If this method is called on a form, then it will return all web components on that form.
If the form is responsive, it will return the web components from all the containers.
```

Parameters

Boolean returnInheritedElements true to also return the elements from parent form

Returns

Array the list of all JSWebComponents on this forms

Supported Clients

NGClient

Sample

```
var webComponents = myForm.getWebComponents(false);
for (var i in webComponents)
{
	if (webComponents[i].name != null)
		application.output(webComponents[i].name);
}
```

newBean(name, className, x, y, width, height)

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

Parameters

String	name	the specified name of the JSBean object
String	className	the class name of the JSBean object
Number	x	the horizontal "x" position of the JSBean object in pixels
Number	y	the vertical "y" position of the JSBean object in pixels
Number	width	the width of the JSBean object in pixels
Number	height	the height of the JSBean object in pixels

Returns

JSComponent a JSBean object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
var bean = form.newBean('bean','com.servoy.extensions.beans.dbtreeview.DBTreeView',200,200,300,300);
forms['newForm1'].controller.show();
```

newButton(txt, x, y, width, height, action)

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

Parameters

String	txt	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
Object	action	the method assigned to handle an onAction event

Returns

JSComponent a new JSButton object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
var method = form.newMethod('function onAction(event) { application.output("onAction intercepted on " + event.
getFormName()); }');
var button = form.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());
```

newButton(txt, action)

Creates a new button on the form with the given text and JSMethod as the onAction event triggered action. You must set location/dimension or css position afterwards

Parameters

String	txt	the text on the button
Object	action	the method assigned to handle an onAction event

Returns

JSComponent a new JSButton object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var method = form.newMethod('function onAction(event) { application.output("onAction intercepted on " + event.
getFormName()); }');
var button = form.newButton('myButton', method);
button.cssPosition.l("10%").t("10%").w("80px").h("30px")
application.output("The new button: " + button.name + " has the following onAction event handling method
assigned " + button.onAction.getName());
```

newCalendar(dataprovider)

Creates a new JSField object on the form with the displayType of CALENDAR - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object	dataprovider	the specified dataprovider name/JSVariable of the JSField object

Returns

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

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var calendar = form.newCalendar(myDataProvider);
calendar.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newCalendar(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newCheck(dataprovider)

Creates a new JSField object on the form with the displayType of CHECK (checkbox) - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object	dataprovider	the specified dataprovider name/JSVariable of the JSField object

Returns

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

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var check = form.newCheck(myDataProvider);
check.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newCheck(dataprovider, 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

Object	dataprovider	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)

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newComboBox(dataprovider)

Creates a new JSField object on the form with the displayType of COMBOBOX - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

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

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var combo = form.newComboBox(myDataProvider);
combo.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newComboBox(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newField(dataprovider, type)

Creates a new JSField object on the form - including the dataprovider/JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object	dataprovider	the specified dataprovider name/JSVariable of the JSField object
Number	type	the display type of the JSField object (see the Solution Model -> JSField node for display types)

Returns

JSField

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var variable = form.newVariable('myVar', JSVariable.TEXT);
variable.defaultValue = "'This is a default value (with triple quotes)!'";
var field = form.newField(variable, JSField.TEXT_FIELD);
field.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newField(dataprovider, type, 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

Object	dataprovider	the specified dataprovider name/JSVariable of the JSField object
Number	type	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)

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newFooterPart(height)

Creates a new Footer part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newHeaderPart(height)

Creates a new Header part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newHtmlArea(dataprovider)

Creates a new JSField object on the form with the displayType of HTML_AREA - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a JSField object on the form with the displayType of HTML_AREA

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var textProvider = form.newVariable('myVar',JSVariable.TEXT);
textProvider.defaultValue = "'This is a triple quoted text!'";
var htmlArea = myListViewForm.newHtmlArea(textProvider);
htmlArea.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newHtmlArea(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newImageMedia(dataprovider)

Creates a new JSField object on the form with the displayType of IMAGE_MEDIA - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a new JSField object on the form with the displayType of IMAGE_MEDIA

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var myMediaVar = form.newVariable("media", JSVariable.MEDIA);
var imageMedia = form.newImageMedia(myMediaVar)
imageMedia.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newImageMedia(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newLabel(txt)

Creates a new JSLabel object on the form - including the text of the label. You must set location/dimension or css position afterwards

Parameters

String txt the specified text of the label object

Returns

JSComponent a JSLabel object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var label = form.newLabel('The text on the label');
label.cssPosition.l("10%").t("10%").w("80px").h("30px")
forms['newForm1'].controller.show();
```

newLabel(txt, x, y, width, height)

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.

Parameters

String	txt	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

Returns

JSComponent a JSLabel object

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newLabel(txt, x, y, width, height, action)

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 a JSMethod action such as the method for an onAction event.

Parameters

String	txt	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
Object	action	the event action JSMethod of the label object

Returns

JSComponent a JSLabel object

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newLayoutContainer()

Create a new layout container as the last child of its parent container.
This method can only be used in responsive forms.

If you want to use default values and so on from a layout package (like 12grid) or if you use the solution model to create a form that is saved back into the workspace (servoyDeveloper.save(form)) then you have to set the packageName and specName properties. So that it works later on in the designer.

If the packageName and specName are not provided, then:
   the packageName is the same as for the parent container
   the specName is the first allowed child defined in the specification of the parent container

If the specification of the parent container does not defined allowed children, then if it is not empty
   the packageName and the specName are copied from the first child layout container.

Returns

JSLayoutContainer the new layout container

Supported Clients

NGClient

Sample

```
var container = form.newLayoutContainer();
container.packageName = "12grid";
container.specName = "row";
```

newLayoutContainer(position)

Create a new layout container. The position is used to determine the generated order in html markup.
This method can only be used in responsive forms.

If you want to use default values and so on from a layout package (like 12grid) or if you use the solution model to create a form that is saved back into the workspace (servoyDeveloper.save(form)) then you have to set the packageName and specName properties. So that it works later on in the designer.

If the packageName and specName are not provided, then:
   the packageName is the same as for the parent container
   the specName is the first allowed child defined in the specification of the parent container

If the specification of the parent container does not defined allowed children, then if it is not empty
   the packageName and the specName are copied from the first child layout container.

Parameters

Number position the position of JSWebComponent object in its parent container

Returns

JSLayoutContainer the new layout container

Supported Clients

NGClient

Sample

```
var container = form.newLayoutContainer(1);
container.packageName = "12grid";
container.specName = "row";
```

newLayoutContainer(position, spec)

Create a new layout container. The position is used to determine the generated order in html markup.
This method can only be used in responsive forms.

Parameters

Number position the position of JSWebComponent object in its parent container
String spec a string of the form 'packageName-layoutName', or 'layoutName'

Returns

JSLayoutContainer the new layout container

Supported Clients

NGClient

Sample

```
var container = form.newLayoutContainer(1, "12grid-row");
container.newLayoutContainer(1, "column");
```

newLayoutContainer(spec)

Create a new layout container as the last child in its parent container.
This method can only be used in responsive forms.

Parameters

String spec a string of the form 'packageName-layoutName', or 'layoutName'

Returns

JSLayoutContainer the new layout container

Supported Clients

NGClient

Sample

```
var container = form.newLayoutContainer(1, "12grid-row");
container.newLayoutContainer(1, "column");
```

newLeadingGrandSummaryPart(height)

Creates a new Leading Grand Summary part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newLeadingSubSummaryPart(height)

Creates a new Leading Subsummary part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newListBox(dataprovider)

Creates a new JSField object on the form with the displayType of LISTBOX - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a new JSField object on the form with the displayType of LISTBOX

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
form.useCssPosition = true
var list = form.newListBox(myDataProvider);
list.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newListBox(dataprovider, x, y, width, height)

Creates a new JSField object on the form with the displayType of LISTBOX - 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

Object	dataprovider	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 LISTBOX

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newMethod(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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

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

newMultiSelectListBox(dataprovider)

Creates a new JSField object on the form with the displayType of MULTISELECT_LISTBOX - including the dataprovider/JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a new JSField object on the form with the displayType of MULTISELECT_LISTBOX

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
form.useCssPosition = true
var list = form.newMultiSelectListBox(myDataProvider);
list.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newMultiSelectListBox(dataprovider, x, y, width, height)

Creates a new JSField object on the form with the displayType of MULTISELECT_LISTBOX - 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

Object	dataprovider	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 MULTISELECT_LISTBOX

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('myForm', 'db:/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(dataprovider)

Creates a new JSField object on the form with the displayType of PASSWORD - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

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

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var pass = form.newPassword(scopes.globals.aVariable);
pass.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newPassword(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newPortal(name, relation)

Creates a new JSPortal object on the form - including the name of the JSPortal object and the relation the JSPortal object is based on. You must set location/dimension or css position afterwards

Parameters

String name the specified name of the JSPortal object

Object relation the relation of the JSPortal object

Returns

JSComponent a JSPortal object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
form.useCssPosition = true
var relation = solutionModel.newRelation('parentToChild','db:/server1/table1','db:/server2/table2',JSRelation.
INNER_JOIN);
relation.newRelationItem('another_parent_table_id', '=', 'another_child_table_parent_id');
var portal = form.newPortal('portal',relation);
portal.cssPosition.l("10%").t("10%").b("10%").r("10%")
portal.newField('someColumn',JSField.TEXT_FIELD,200,200,120);
forms['newForm1'].controller.show();
```

newPortal(name, relation, 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.

Parameters

String	name	the specified name of the JSPortal object
Object	relation	the relation of the JSPortal object
Number	x	the horizontal "x" position of the JSPortal object in pixels
Number	y	the vertical "y" position of the JSPortal object in pixels
Number	width	the width of the JSPortal object in pixels
Number	height	the height of the JSPortal object in pixels

Returns

JSComponent a JSPortal object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
var relation = solutionModel.newRelation('parentToChild','db:/server1/table1','db:/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(dataprovider)

Creates a new JSField object on the form with the displayType of RADIOS (radio buttons) - including the dataprovider/JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

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

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
form.useCssPosition = true
var vlist = solutionModel.newValueList('options',JSValueList.CUSTOM_VALUES);
vlist.customValues = "value1\nvalue2\nvalue3";
var radios = form.newRadios('columnDataProvider');
radios.cssPosition.l("10").t("10").w("20%").h("30px")
radios.valuelist = vlist;
```

newRadios(dataprovider, 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

Object	dataprovider	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)

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, 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(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newSpinner(dataprovider)

Creates a new JSField object on the form with the displayType of SPINNER - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object	dataprovider	the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a new JSField object on the form with the displayType of SPINNER

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
form.useCssPosition = true
var spinner = form.newSpinner(myDataProvider, 10, 460, 100, 20);
spinner.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newSpinner(dataprovider, x, y, width, height)

Creates a new JSField object on the form with the displayType of SPINNER - 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

Object	dataprovider	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 SPINNER

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'myServer', 'myTable', null, true, 800, 600);
var spinner = form.newSpinner(myDataProvider, 10, 460, 100, 20);
forms['newForm1'].controller.show();
```

newTabPanel(name)

Creates a new JSTabPanel object on the form - including the name of the JSTabPanel object. You must set location /dimension or css position afterwards

Parameters

String	name	the specified name of the JSTabPanel object

Returns

JSComponent a JSTabPanel object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('parentForm','db:/server1/parent_table',null,false,640,480);
form.useCssPosition = true
var childOne = solutionModel.newForm('childOne','db:/server1/child_table',null,false,400,300);
childOne.newField('child_table_text', JSField.TEXT_FIELD,10,10,100,20);
var parentToChild = solutionModel.newRelation('parentToChild','db:/server1/parent_table','db:/server1
/child_table',JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id','=','child_table_parent_id');
var childTwo = solutionModel.newForm('childTwo','db:/server1/my_table',null,false,400,300);
childTwo.newField('my_table_image', JSField.IMAGE_MEDIA,10,10,100,100);
var tabPanel = form.newTabPanel('tabs');
tabPanel.cssPosition.l("10%").t("10%").b("10%").r("10%")
tabPanel.newTab('tab1','Child One',childOne,parentToChild);
tabPanel.newTab('tab2','Child Two',childTwo);
forms['parentForm'].controller.show();
```

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

JSComponent a JSTabPanel object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('parentForm','db:/server1/parent_table',null,false,640,480);
var childOne = solutionModel.newForm('childOne','db:/server1/child_table',null,false,400,300);
childOne.newField('child_table_text', JSField.TEXT_FIELD,10,10,100,20);
var parentToChild = solutionModel.newRelation('parentToChild','db:/server1/parent_table','db:/server1
/child_table',JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id','=','child_table_parent_id');
var childTwo = solutionModel.newForm('childTwo','db:/server1/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(dataprovider)

Creates a new JSField object on the form with the displayType of TEXT_AREA - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a JSField object with the displayType of TEXT_AREA

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1',myDatasource,null,true,800,600);
form.useCssPosition = true
var globalVar = solutionModel.newGlobalVariable('globals', 'myGlobal',JSVariable.TEXT);
globalVar.defaultValue = "'Type your text in here'";
var textArea = form.newTextArea(globalVar);
textArea.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newTextArea(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newTextField(dataprovider)

Creates a new JSField object on the form with the displayType of TEXT_FIELD - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a JSField object with the displayType of TEXT_FIELD

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1',myDatasource,null,true,800,600);
form.useCssPosition = true
var x = solutionModel.newGlobalVariable('globals', 'myGlobal',JSVariable.TEXT);
x.defaultValue = "'Text from a global variable'"
var textField = form.newTextField(x);
textField.cssPosition.l("10").t("10").w("20%").h("30px")
forms['newForm1'].controller.show();
```

newTextField(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1',myDatasource,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('globals', 'myGlobal',JSVariable.TEXT);
//x.defaultValue = "'Text from a global variable'";
//or a form jsvariable as the dataprovider
//x = form.newVariable('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(height)

Creates a new Title Footer part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newTitleHeaderPart(height)

Creates a new Title Header part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newTrailingGrandSummaryPart(height)

Creates a new Trailing Grand Summary part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newTrailingSubSummaryPart(height)

Creates a new Trailing Subsummary part on the form.

Parameters

Number height The height of the new part

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

newTypeAhead(dataprovider)

Creates a new JSField object on the form with the displayType of TYPE_AHEAD - including the dataprovider /JSVariable of the JSField object. You must set location/dimension or css position afterwards

Parameters

Object dataprovider the specified dataprovider name/JSVariable of the JSField object

Returns

JSField a JSField object with the displayType of TYPE_AHEAD

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1',myDatasource,null,true,800,600);
form.useCssPosition = true
var vlist = solutionModel.newValueList('options',JSValueList.CUSTOM_VALUES);
vlist.customValues = "value1\nvalue2\nvalue3";
var typeAhead = form.newTypeAhead(columnTextDataProvider);
typeAhead.cssPosition.l("10").t("10").w("20%").h("30px")
typeAhead.valuelist = vlist;
forms['newForm1'].controller.show();
```

newTypeAhead(dataprovider, 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

Object	dataprovider	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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newForm1',myDatasource,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();
```

newVariable(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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
var variable = form.newVariable('myVar', JSVariable.TEXT , "'This is a default value (with triple quotes)!'");
//or variable = form.newVariable('myVar', JSVariable.TEXT)
//variable.defaultValue = "'This is a default value (with triple quotes)!'" // setting the default value after
the variable is created requires form recreation
//variable.defaultValue = "{a:'First letter',b:'Second letter'}"
var field = form.newField(variable, JSField.TEXT_FIELD, 100, 100, 200, 200);
forms['newForm1'].controller.show();
```

newVariable(name, type, defaultValue)

Creates a new form JSVariable - based on the name of the variable object , the type and it's default value , uses the SolutionModel JSVariable constants.

This method does not require the form to be destroyed and recreated. Use this method if you want to change the form's model without destroying the runtime form

Parameters

String name the specified name of the variable
Number type the specified type of the variable (see Solution Model -> JSVariable node constants)
String defaultValue the default value as a javascript expression string

Returns

JSVariable a JSVariable object

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newForm1', myDatasource, null, true, 800, 600);
var variable = form.newVariable('myVar', JSVariable.TEXT , "'This is a default value (with triple quotes)!'");
//or variable = form.newVariable('myVar', JSVariable.TEXT)
//variable.defaultValue = "'This is a default value (with triple quotes)!'" // setting the default value after
the variable is created requires form recreation
//variable.defaultValue = "{a:'First letter',b:'Second letter'}"
var field = form.newField(variable, JSField.TEXT_FIELD, 100, 100, 200, 200);
forms['newForm1'].controller.show();
```

newWebComponent(type)

Creates a new JSWebComponent (spec based component) object on the RESPONSIVE form.
Will receive a generated name. Will be added as last position in container.

Parameters

String type the webcomponent name as it appears in the spec

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
var container = myForm.getLayoutContainer("row1")
var bean = container.newWebComponent('mypackage-testcomponent');
```

newWebComponent(type, position)

Creates a new JSWebComponent (spec based component) object on the RESPONSIVE form.
Will receive a generated name.

Parameters

String type the webcomponent name as it appears in the spec
Number position the position of JSWebComponent object in its parent container

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
var container = myForm.getLayoutContainer("row1")
var bean = container.newWebComponent('mypackage-testcomponent',1);
```

newWebComponent(name, type)

Creates a new JSWebComponent (spec based component) object on a form. You must set location/dimension or css
position afterwards

Parameters

String name the specified name of the JSWebComponent object
String type the webcomponent name as it appears in the spec

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
form.useCssPosition = true
var webcomponent = form.newWebComponent('mywebcomponent','mypackage-testcomponent');
```

newWebComponent(name, type, position)

Creates a new JSWebComponent (spec based component) object on the RESPONSIVE form.

Parameters

String	name	the specified name of the JSWebComponent object
String	type	the webcomponent name as it appears in the spec
Number	position	the position of JSWebComponent object in its parent container

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
var container = myForm.getLayoutContainer("row1")
var bean = container.newWebComponent('bean','mypackage-testcomponent',1);
```

newWebComponent(name, type, x, y, width, height)

Creates a new JSWebComponent (spec based component) object on the form.

Parameters

String	name	the specified name of the JSWebComponent object
String	type	the webcomponent name as it appears in the spec
Number	x	the horizontal "x" position of the JSWebComponent object in pixels
Number	y	the vertical "y" position of the JSWebComponent object in pixels
Number	width	the width of the JSWebComponent object in pixels
Number	height	the height of the JSWebComponent object in pixels

Returns

JSComponent a JSWebComponent object

Supported Clients

NGClient

Sample

```
var form = solutionModel.newForm('newForm1', 'db:/server1/table1', null, true, 800, 600);
var bean = form.newWebComponent('bean','mypackage-testcomponent',200,200,300,300);
forms['newForm1'].controller.show();
```

putDesignTimeProperty(key, value)

Set a design-time property of a form.

Parameters

String	key	the property name
Object	value	the value to set

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var frm = solutionModel.getForm('orders')
frm.putDesignTimeProperty('myprop', 'lemon')
```

removeBean(name)

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

Parameters

String name the specified name of the JSBean to be removed

Returns

Boolean true if the JSBean has been removed; false otherwise

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.getForm('myform');
form.removeBean('mybean')
```

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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newFormX',myDatasource,null,true,800,600);
var b1 = form.newButton('This is button1',100,100,200,50,null);
b1.name = 'b1';
var jsmethod = form.newMethod("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(name)

Removes a component (JSLabel, JSButton, JSField, JSPortal, JSBean, JSTabpanel, JSWebComponent) 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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newFormX','db:/server1/parent_table',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('scopes.globals.myGlobalVariable',JSField.TEXT_FIELD,100,300,200,50);
jsfield.name = 'jsf';
var relation = solutionModel.newRelation('parentToChild','db:/server1/parent_table','db:/server1/child_table',
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','db:/server1/child_table',null,false,400,300);
childOne.newField('child_table_id', JSField.TEXT_FIELD,10,10,100,20);
var childTwo = solutionModel.newForm('childTwo','server1','other_table',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.newMethod("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();
```

removeDesignTimeProperty(key)

Clear a design-time property of a form.

Parameters

String key the property name

Returns

Object

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var frm = solutionModel.getForm('orders')
frm.removeDesignTimeProperty('myprop')
```

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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newFormX',myDatasource,null,true,800,600);
var jsfield = form.newField(scopes.globals.myGlobalVariable,JSField.TEXT_FIELD,100,300,200,50);
jsfield.name = 'jsf';
var jsmethod = form.newMethod("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(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

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newFormX',myDatasource,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.newMethod("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();
```

removeMethod(name)

Removes a form JSMethod - based on the specified code.

Parameters

String name the specified name of the method

Returns

Boolean true if method was removed successfully , false otherwise

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newForm1', null, null, true, 800, 600);
var hello = form.newMethod('function aMethod(event){application.output("Hello world!");}');
var removeMethod = form.newMethod('function removeMethod(event){ \
                                                                solutionModel.getForm(event.
getFormName()).removeMethod("aMethod"); \
                                                                forms[event.getFormName()].controller.
recreateUI();\
                                                                }');
var button1 = form.newButton('Call method!',50,50,120,30,hello);
var button2 = form.newButton('Remove Mehtod!',200,50,120,30,removeMethod);
forms['newForm1'].controller.show();
```

removePart(type)

Removes a JSPart of the given type.

Parameters

Number type The type of the part that should be removed.

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

removePart(type, height)

Removes a JSPart of the given type. The height parameter is for removing 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. This parameter is for removing one of multiple Leading/Trailing Subsummary parts.

Returns

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

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

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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newFormX',myDatasource,null,true,800,600);
var relation = solutionModel.newRelation('parentToChild','db:/server1/myTable','db:/server1/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.newMethod("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(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

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var form = solutionModel.newForm('newFormX','db:/server1/parent_table',null,false,800,600);
var childOne = solutionModel.newForm('childOne','db:/server1/child_table',null,false,400,300);
childOne.newField('child_table_text', JSField.TEXT_FIELD,10,10,100,20);
var parentToChild = solutionModel.newRelation('parentToChild','db:/server1/parent_table','db:/server1
/child_table',JSRelation.INNER_JOIN);
parentToChild.newRelationItem('parent_table_id','=','child_table_id');
var childTwo = solutionModel.newForm('childTwo','db:/server1/another_table',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.newMethod("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();
```

removeVariable(name)

Removes a form JSVariable - based on the name of the variable object.

Parameters

String name the specified name of the variable

Returns

Boolean true if removed, false otherwise (ex: no var with that name)

Supported Clients

SmartClient,WebClient,NGClient,MobileClient

Sample

```
var form = solutionModel.newForm('newForm1', null, null, true, 800, 600);
var variable = form.newVariable('myVar', JSVariable.TEXT);
variable.defaultValue = "'This is a default value (with triple quotes)!'";
//variable.defaultValue = "{a:'First letter',b:'Second letter'}"
var field = form.newField(variable, JSField.TEXT_FIELD, 100, 100, 200, 200);
forms['newForm1'].controller.show();

variable = form.removeVariable('myVar');
application.sleep(4000);
forms['newForm1'].controller.recreateUI();
```

removeWebComponent(name)

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

Parameters

String name the specified name of the JSWebComponent to be removed

Returns

Boolean true if the JSWebComponent has been removed; false otherwise

Supported Clients

NGClient

Sample

```
var form = solutionModel.getForm('myform');
form.removeWebComponent('mybean')
```