JSWindow

Feb 26, 2024 08:58

Supported Clients

SmartClient WebClient NGClient

Constants Summary

Number	DEFAULT	Value used for x, y, width, height of initial bounds when you want the window to auto-determine bounds when shown for the first time.
Number	DIALOG	Window type constant that identifies a non-modal dialog type.
Number	FULL_SCREEN	Value that can be used for bounds in order to specify that a dialog/window should completely fill the screen.
Number	MODAL_DIALOG	c Window type constant that identifies a modal dialog type.
Number	WINDOW	Window type constant that identifies a window type.

Property Summary

controller	controller	Get the current controller from the window/dialog.
Number	opacity	Gets/Sets the opacity property.
Boolean	resizable	Gets/Sets whether or not this window can be resized by the user (default true).
Boolean	storeBounds	Tells whether or not the bounds of this window should be stored/persisted (default false).
String	title	Gets/Sets the title text.
Boolean	transparent	Gets/Sets the transparency property.
Boolean	undecorated	Gets/Sets the undecorated property.

Methods Summary

void	destroy()	Frees the resources allocated by this window.
Number	getHeight()	Returns the height.
String	getName()	Returns the window name.
JSWindow	getParent()	Returns the parent JSWindow, if available.
Number	getType()	Returns the window type.
Number	getWidth()	Returns the width.
Number	getX()	Returns the x coordinate.
Number	getY()	Returns the y coordinate.
Boolean	hide()	Hides the window.
Boolean	isVisible()	Returns true if the window is visible, false otherwise.
void	resetBounds()	Deletes the window's currently stored bounds.
void	setCSSClass(cssClassName)	Sets the dialog CSS class, can not be used to alter it when already showing, this should be set before the dialog is used.
void	setInitialBounds(x, y, width, height)	Sets the initial window bounds.
void	setLocation(x, y)	Set the window location.
void	setSize(width, height)	Set the window size.
void	show(form)	Shows the given form(form name, form object or JSForm) in this window.
void	showTextToolbar(showTextToolbar)	Sets whether or not this window should have a text tool bar.
void	toBack()	Shows this window behind other windows, if possible.
void	toFront()	Bring this window in front of other windows, if possible.

Constants Details

DEFAULT

```
Value used for x, y, width, height of initial bounds when you want the window to auto-determine bounds when shown for the first time.
```

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// show a dialog that self-determines bounds the first time it it open, then remembers last bounds for future
show operations
var win = application.createWindow("myName", JSWindow.DIALOG);
win.setInitialBounds(JSWindow.DEFAULT, JSWindow.DEFAULT, JSWindow.DEFAULT, JSWindow.DEFAULT); // will be shown
initially centred and with preferred size
forms.myForm.show(win);
```

DIALOG

Window type constant that identifies a non-modal dialog type.
Non-modal dialogs will allow the user to interact with parent windows, but are less independent then windows with WINDOW type.
Dialogs will stay on top of parent windows and are less accessible through the OS window manager. In web-client dialogs will not
open in a separate browser window.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create a non-modal dialog on top of current active form's window and show a form inside it
var myWindow = application.createWindow("myName", JSWindow.DIALOG);
myWindow.show(forms.myForm);
```

FULL_SCREEN

Value that can be used for bounds in order to specify that a dialog/window should completely fill the screen.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create and show a window, with specified title, full screen
var win = application.createWindow("windowName", JSWindow.WINDOW);
win.setInitialBounds(JSWindow.FULL_SCREEN, JSWindow.FULL_SCREEN, JSWindow.FULL_SCREEN, JSWindow.FULL_SCREEN);
win.setTitle("This is a window");
controller.show(win);
```

MODAL_DIALOG

c
Window type constant that identifies a modal dialog type. Modal dialogs will not allow the user to interact with the parent window(s) until closed.
Dialogs will stay on top of parent windows and are less accessible through the OS window manager. In web-client dialogs will not
open in a separate browser window. NOTE: no code is executed in Smart Client after a modal dialog is shown (the show operation blocks) until this dialog closes.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create a modal dialog on top of current active form's window and show a form inside it
var myWindow = application.createWindow("myName", JSWindow.MODAL_DIALOG);
myWindow.show(forms.myForm);
```

WINDOW

Window type constant that identifies a window type. WINDOW type is the most independent type of window. It will be more accessible through the OS window
manager, it can appear both in front of and under other windows and it doesn't block user interaction for other windows. In web-client windows will
open in a separate browser window.

Returns

Number

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// create a window and show a form inside it
var myWindow = application.createWindow("myName", JSWindow.WINDOW);
myWindow.show(forms.myForm);
```

Property Details

controller

Get the current controller from the window/dialog.

Returns

controller

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var formName = application.getWindow('test').controller.getName();
```

opacity

Gets/Sets the opacity property. By default will have value 1 (completely opaque), and can be assigned to values between 0 and 1.
If set then window will also be undecorated. This should be set before the dialog/window is shown, otherwise it has no effect.

Returns

Number the opacity of this window

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

resizable

Gets/Sets whether or not this window can be resized by the user (default true).

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.getWindow("someWindowName");
if (someWindow.isVisible() == false) {
        controller.show(someWindow);
        someWindow.resizable = false;
}
```

storeBounds

Tells whether or not the bounds of this window should be stored/persisted (default false).
If true, the window's bounds will be stored when the window is closed. Stored bounds will be used when the window is shown again instead of initialBounds.
For non resizable windows, only location is stored/persisted.

Returns

Boolean

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win1 = application.createWindow("Window 1", JSWindow.DIALOG, null);
win1.setInitialBounds(200, 200, 450, 350);
win1.resizable = false;
win1.storeBounds = true;
win1.title = "Window 1";
controller.show(win1);
```

title

Gets/Sets the title text.

Returns

String

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win1 = application.createWindow("Window 1", JSWindow.WINDOW, null);
win1.setInitialBounds(200, 200, 450, 350);
win1.title = "Window 1";
controller.show(win1);
```

transparent

Gets/Sets the transparency property.
NOTE: For smart clients, the window must be undecorated or the servoy.smartclient.allowLAFWindowDecoration property set to true

Returns

Boolean transparency state of the window

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

undecorated

Gets/Sets the undecorated property.
If set then this window will not have any decoration and can't be moved/resized or closed. This should be set before dialog/window is shown, otherwise has no effect.

Returns

Boolean if this window will be undecorated

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

Methods Details

destroy()

Frees the resources allocated by this window. If window is visible, it will close it first.
The window will no longer be available with application.getWindow('windowName') and will no longer be usable.

The main application window cannot be destroyed.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var getWindow = application.getWindow("someWindowName");
getWindow.destroy();
getWindow = application.getWindow("someWindowName");
if (getWindow == null) {
        application.output("Window has been destroyed");
} else {
        application.output("Window could not be destroyed");
}
```

getHeight()

Returns the height.

Returns

Number the height.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

getName()

Returns the window name. It will be null in case of main application frame.

Returns

String the window name.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

getParent()

Returns the parent JSWindow, if available.

Returns

JSWindow the parent JSWindow, if available. If there is no parent JSWindow, it will return null.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

getType()

Returns the window type.

Returns

Number the window type. Can be one of JSWindow.DIALOG, JSWindow.MODAL_DIALOG, JSWindow.WINDOW.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

getWidth()

Returns the width.

Returns

Number the width.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

getX()

Returns the x coordinate.

Returns

Number the x coordinate.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

getY()

Returns the y coordinate.

Returns

Number the y coordinate.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.createWindow("someWindowName", JSWindow.WINDOW, null);
someWindow.setInitialBounds(200, 200, 450, 350);
controller.show(someWindow);

var name = "Name: " + someWindow.getName() + "\n"
var parent = "Parent: " + (someWindow.getParent() == null ? "none" : someWindow.getParent()) + "\n"
var type = "TypeNumber: " + someWindow.getType() + "\n"
var height = "Height: " + someWindow.getHeight() + "\n"
var width = "Width: " + someWindow.getWidth() + "\n"
var undecorated = "Undecorated: " + someWindow.isUndecorated() + "\n"
var opacity = "Opacity: " + someWindow.opacity + "\n"
var transparent = "Transparent: " + someWindow.transparent + "\n"
var locationX = "Location-X-coordinate: " + someWindow.getX() + "\n"
var locationY = "Location-Y-coordinate: " + someWindow.getY() + "\n"
var info = name + parent + type + height + width + locationX + locationY + undecorated + "\n"
var closeMsg = "Press 'Ok' to close this dialog."

var infoDialog = plugins.dialogs.showInfoDialog("Window Info", info + closeMsg, "Ok");
if (infoDialog == "Ok") someWindow.close()
```

hide()

Hides the window. It can be shown again using window.show(), controller.show() or controller.showRecords().
The main application window cannot be hidden.

Returns

Boolean Boolean true if the window was successfully closed and false otherwise.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
//creates and shows a window for 3 seconds before closing it
var win = application.createWindow("someWindowName", JSWindow.WINDOW, null);
win.setInitialBounds(200, 200, 450, 350);
controller.show(win);
application.sleep(3000);
win.hide();
```

isVisible()

Returns true if the window is visible, false otherwise.

Returns

Boolean true if the window is visible, false otherwise.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var someWindow = application.getWindow("someWindowName");
if (someWindow.isVisible() == false) {
        controller.show(someWindow);
        someWindow.resizable = false;
}
```

resetBounds()

Deletes the window's currently stored bounds. It will only affect the next show of the window.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win1 = application.createWindow("Window 1", JSWindow.DIALOG, null);
win1.title = "Window 1";
win1.setInitialBounds(200, 200, 400, 600);
win1.storeBounds = true;
if (newSolutionVersion) win1.resetBounds();
win1.show(forms.myform);
```

setCSSClass(cssClassName)

Sets the dialog CSS class, can not be used to alter it when already showing, this should be set before the dialog is used.
See sample code for examples of CSS classes for display customizations

Parameters

String cssClassName CSS class name

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
// Here are some examples of a classes for customizing the display of the window

//        .myDlgCSS {
//                border-radius: initial; // show edged dialog corners
//                -webkit-box-shadow: 0 5px 15px rgba(0,0,0,.0); // hide dialog box shadow
//                box-shadow: 0 5px 15px rgba(0,0,0,.0); // hide dialog box shadow
//        }
//
//        // dialog header styling
//        .myDlgCSS .window-header {
//                background: green;
//        }
//
//        // style/hide dialog close button
//        .myDlgCSS .window-header .svy-dialog-close {
//                display: none;
//        }
//
//        // dialog body styling
//        .myDlgCSS .window-body {
//                background: yellow;
//        }
//
//        // dialog footer styling/hiding
//        .myDlgCSS .window-footer {
//                background: blue;
//                display: none !important;
//        }

var win = application.createWindow("myName", JSWindow.DIALOG);
win.setCSSClass('myDlgCSS');
```

setInitialBounds(x, y, width, height)

Sets the initial window bounds.
The initial bounds are only used the first time this window is shown (what first show means depends on storeBounds property).

Parameters

- Number x — the initial x coordinate of the window. Can be JSWindow.DEFAULT, JSWindow.FULL_SCREEN.
- Number y — the initial y coordinate of the window. Can be JSWindow.DEFAULT, JSWindow.FULL_SCREEN.
- Number width — the initial width of the window. Can be JSWindow.DEFAULT, JSWindow.FULL_SCREEN.
- Number height — the initial height of the window. Can be JSWindow.DEFAULT, JSWindow.FULL_SCREEN.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win = application.createWindow("myName", JSWindow.DIALOG);
win.setInitialBounds(20, 10, 300, 200);
forms.myForm.show(win);
```

setLocation(x, y)

Set the window location.
If the coordinates are not valid they might be corrected. (for example out of screen locations)

Parameters

- Number x — x coordinate.
- Number y — y coordinate.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var window = application.createWindow('test',JSWindow.DIALOG);
window.setLocation(0,0);
window.setSize(400,600);
window.show(forms.child1);
```

setSize(width, height)

Set the window size.

Parameters

Number width the width.
Number height the height.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var window = application.createWindow('test',JSWindow.DIALOG);
window.setLocation(0,0);
window.setSize(400,600);
window.show(forms.child1);
```

show(form)

Shows the given form(form name, form object or JSForm) in this window.

Parameters

Object form the form that will be shown inside this window. It can be a form name or a form object (actual form or JSForm).

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
win.show(forms.myForm);
// win.show("myForm");
```

showTextToolbar(showTextToolbar)

Sets whether or not this window should have a text tool bar. Has no effect on web client or smart client main application frame.

Parameters

Boolean showTextToolbar true if you want a text tool bar to be added to this window, false otherwise.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win1 = application.createWindow("Window 1", JSWindow.WINDOW, null);
win1.setInitialBounds(200, 200, 450, 350);
win1.setTitle("Window 1");
win1.showTextToolbar(false);
controller.show(win1);

var win2 = application.createWindow("Window 2", JSWindow.WINDOW, null);
win2.setInitialBounds(500, 500, 450, 350);
win2.setTitle("Window 2");
win2.showTextToolbar(false);
controller.show(win2);

var win3 = application.createWindow("Window 3", JSWindow.WINDOW, null);
win3.setInitialBounds(650, 700, 450, 350);
win3.setTitle("Window 3");
win3.showTextToolbar(true);
controller.show(win3);

application.sleep(2000);
win3.toBack();
application.sleep(2000);
win1.toFront();
```

toBack()

Shows this window behind other windows, if possible.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win1 = application.createWindow("Window 1", JSWindow.WINDOW, null);
win1.setInitialBounds(200, 200, 450, 350);
win1.setTitle("Window 1");
win1.showTextToolbar(false);
controller.show(win1);

var win2 = application.createWindow("Window 2", JSWindow.WINDOW, null);
win2.setInitialBounds(500, 500, 450, 350);
win2.setTitle("Window 2");
win2.showTextToolbar(false);
controller.show(win2);

var win3 = application.createWindow("Window 3", JSWindow.WINDOW, null);
win3.setInitialBounds(650, 700, 450, 350);
win3.setTitle("Window 3");
win3.showTextToolbar(true);
controller.show(win3);

application.sleep(2000);
win3.toBack();
application.sleep(2000);
win1.toFront();
```

toFront()

Bring this window in front of other windows, if possible.

Supported Clients

SmartClient,WebClient,NGClient

Sample

```
var win1 = application.createWindow("Window 1", JSWindow.WINDOW, null);
win1.setInitialBounds(200, 200, 450, 350);
win1.setTitle("Window 1");
win1.showTextToolbar(false);
controller.show(win1);

var win2 = application.createWindow("Window 2", JSWindow.WINDOW, null);
win2.setInitialBounds(500, 500, 450, 350);
win2.setTitle("Window 2");
win2.showTextToolbar(false);
controller.show(win2);

var win3 = application.createWindow("Window 3", JSWindow.WINDOW, null);
win3.setInitialBounds(650, 700, 450, 350);
win3.setTitle("Window 3");
win3.showTextToolbar(true);
controller.show(win3);

application.sleep(2000);
win3.toBack();
application.sleep(2000);
win1.toFront();
```