jsoFlow Version 1.0

by Brainy Data Limited

About jsoFlow

Introduction

The jsoFlow control was designed and developed by Perry and is a new JSON control which we first presented at EurOmnis 2025. In simple terms, this control enables a flow-chart style navigation of the functions of your software. It provides a visual aid for end-users in understanding the various stages of procedures that your software implements and how these stages relate to each other. The image below shows an example of how this type of interface has been and can be used in accounts software involving purchasing, selling, PAYE and, banking procedures. In our example the control displays a number of panels, Suppliers, Customers, Employees, etc., each with interactive buttons performing various actions related to its panel. For example, the suppliers panel has buttons for creating purchase orders, receiving stock, entering and paying bills. The stages of the purchasing procedure are shown by the relation arrows. Panel headings, i.e. SUPPLIERS, CUSTOMERS, etc, are also buttons that can generate click events.

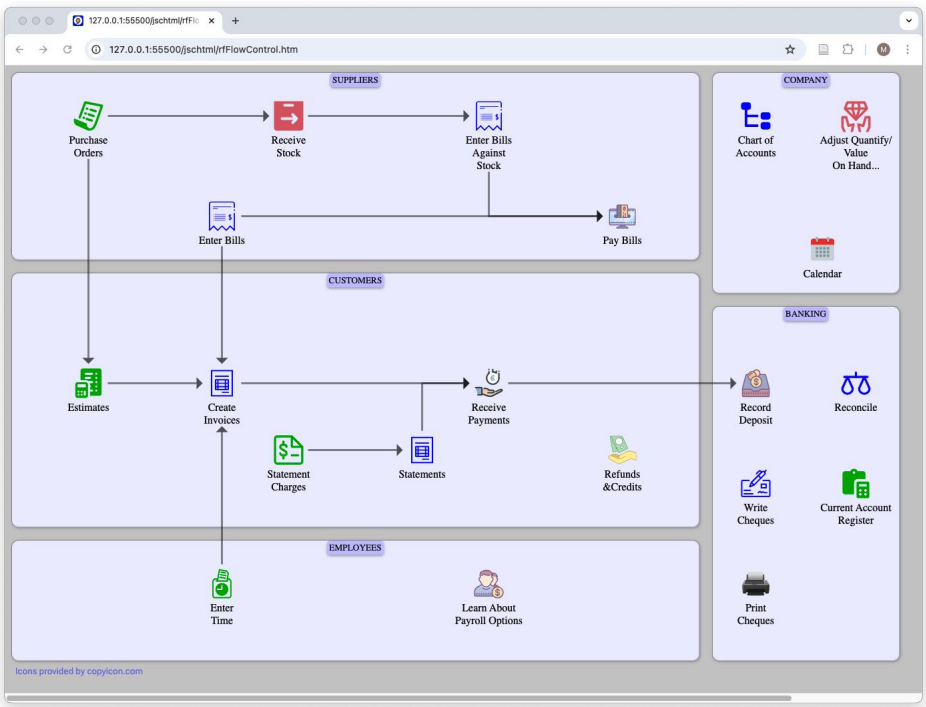

Figure 1.1: flow control example of accounts software
Note: the provided example library merely presents a visual interface with no actual accounts functionality

If the above example looks familiar, that is because it is modelled on the popular Quick-books accounts software which has incorporated a similar flow-chart style interface.

Another example use for this control is a learning tool that guides the user through the various stages of a particular subject. The image below shows a work-in-progress by Perry who is designing this interactive tool to cover the basics about HTML, CSS and JavaScript development.

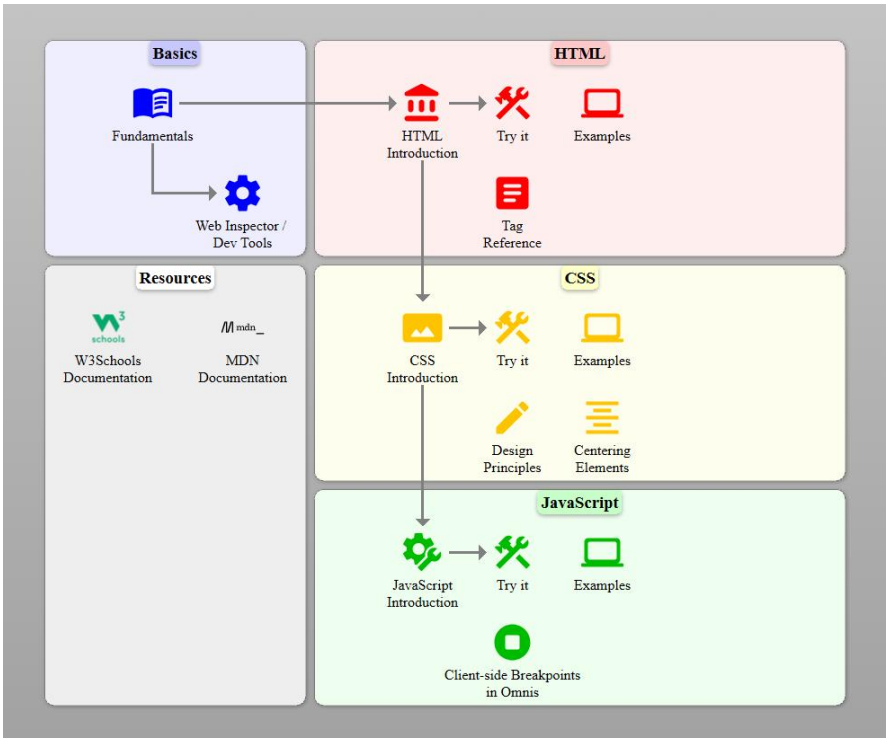

Figure 1.2: flow control example of a teaching tool

This type of interface can be highly intuitive and helps to reduce the need for complex menu systems and toolbars for the most common tasks.

Downloading the Software and Examples

If you have an active Developer Maintenance Subscription (DMS) for jsoFlow, you can download the current release software from https://support.brainydata.com/jsoflow.htm. The download page will list two downloads:

jsoflow_<version>_js.zip and
jsoflow_<version>_master_js.zip.

The former is the standard release build for that version, whereas the latter includes the non-minified JavaScript source files and requires a full Developer Support Subscription (DSS) to

download. If you download the master build, make sure you read the included ReadMe.txt file regarding how to distribute the software, as you are not allowed to distribute non-minified versions of our JavaScript source.

Installing the Software

The jsoFlow software and examples were primarily tested with Studio 11.1 build 37255, but should also work with earlier versions.

Copy files to Studio tree

Installation requires copying a number of files or folders to the Omnis Studio HTML support folder and the editing of the Studio "jsctempl.htm' file and either Launch Omnis Studio, or reload the control via the JSON control editor if you do not wish to restart Studio.

First copy the control's interface source, CSS, JavaScript and example Icon set.

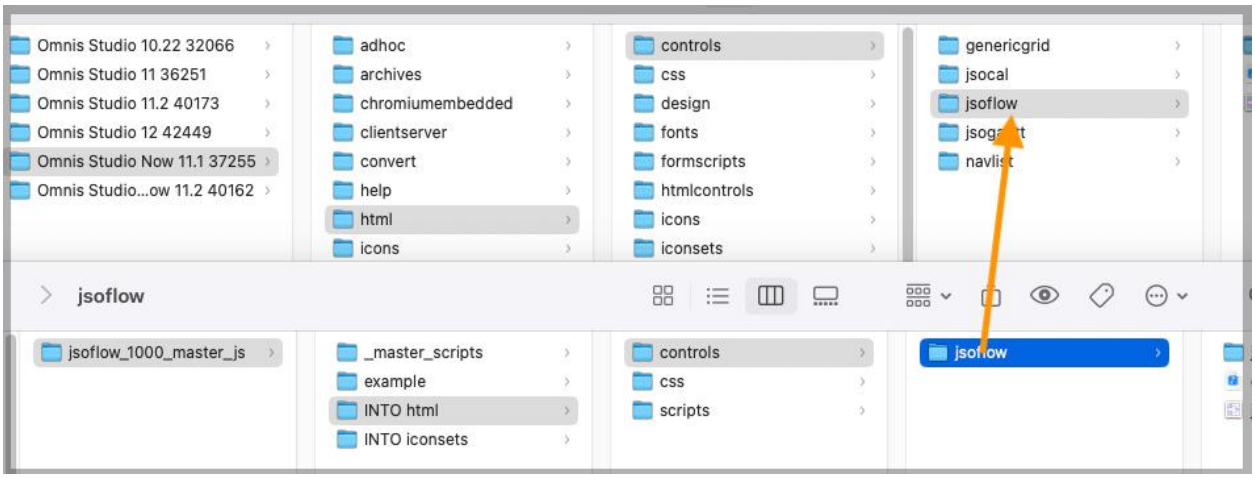

Figure 1.3: where to copy the controls interface source to the html/controls folder

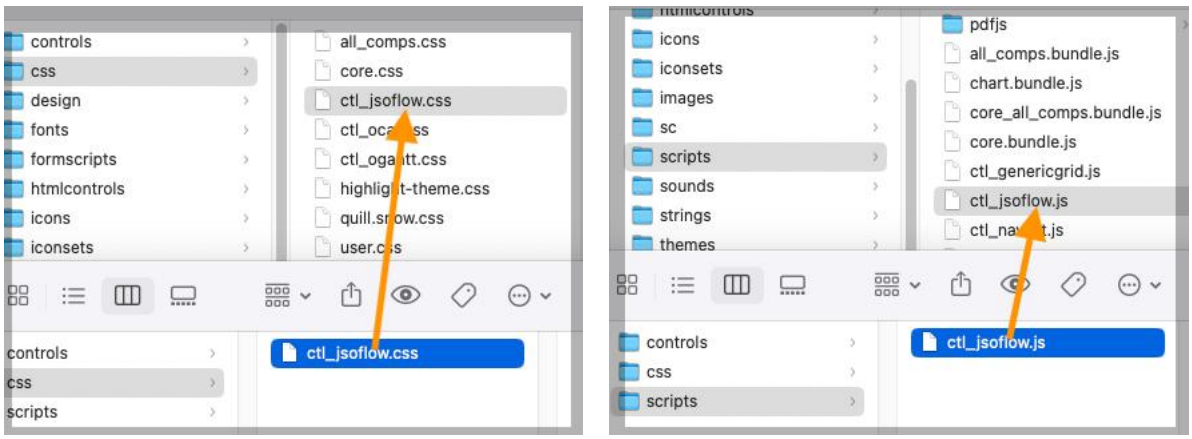

Figure 1.4: where to copy the control's css and js files to the css and scripts folders respectively

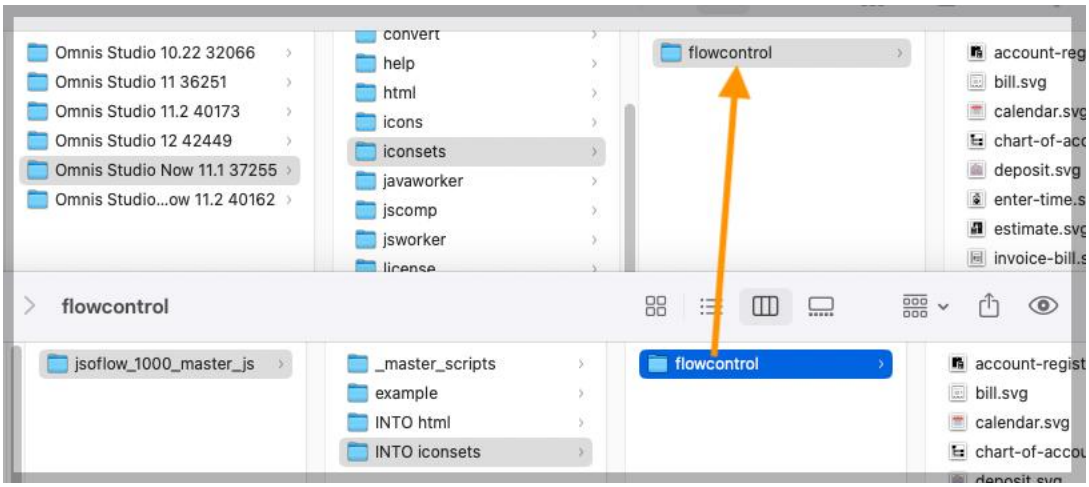

Figure 1.5: copy the example Icon set to the Studio ‘iconsets’ folder

If you have also installed the JSCBridge for use with jsoflow, you must copy the *ctl_jsoflow.css* and *ctl_jsoflow.js* files and the *flowcontrol* icons folder to the equivalent folders in the *jsclient_bridge* folder.

Note: If you do not use the JS Client in your deployment, you do not need to distribute the HTML folder with your runtime. This installation is only required in order to design the remote forms within the Studio IDE. You will need to distribute the JSBridge folder, either as part of your ‘firstruninstall’ folder or as a separate installation to the Omnis Studio support folder once Studio is installed.

Edit the Studio HTML template

Finally, edit the studio “jsctemplate.htm” and insert the lines that will load the jsoFlow Cascading Style Sheet and JavaScript.

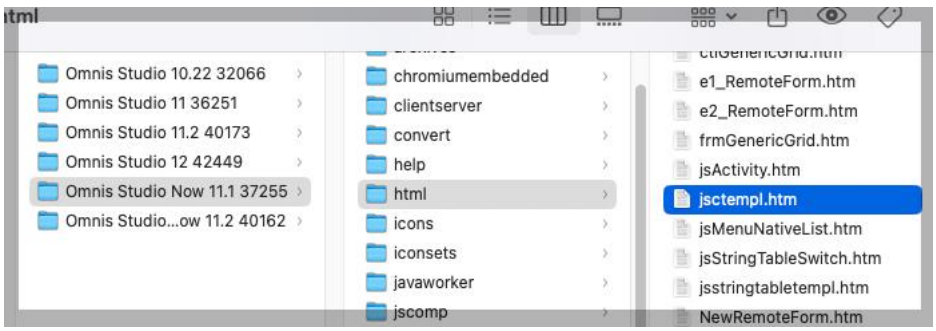

Figure 1.6: the location of the template file

Figure 1.7 showing the lines to be inserted in the template file

Deploying your software

Please refer to the license agreement for rules on deployment.

Documentation

The remainder of the documentation covers the following topics

- Designing a jsoFlow interface
- Deploying to both JavaScript client and Desktop client installations
- Control Reference listing the controls features
- Example Reference describing the example classes in more detail

Designing jsoFlow

Introduction

The jsoFlow control is implemented using the Omnis Studio JSON control editor and, although designed to work in browsers, can also be deployed in fat-clients using the JSCBridge SDK. Both these implementations were showcased at the EurOmnis 2025 conference and this chapter discusses the JS Client implementation.

The Design Interface

As jsoFlow is essentially a JavaScript control with a JSON design interface that merely describe its properties, developing a fully interactive design interface is not possible within an Omnis design window. The reason is that when opening a JSON control on a remote form in a design window, Omnis blocks all mouse based interaction. This is not ideal for a complex control such as jsoFlow which requires some kind of intuitive drag & drop interaction for designing its appearance.

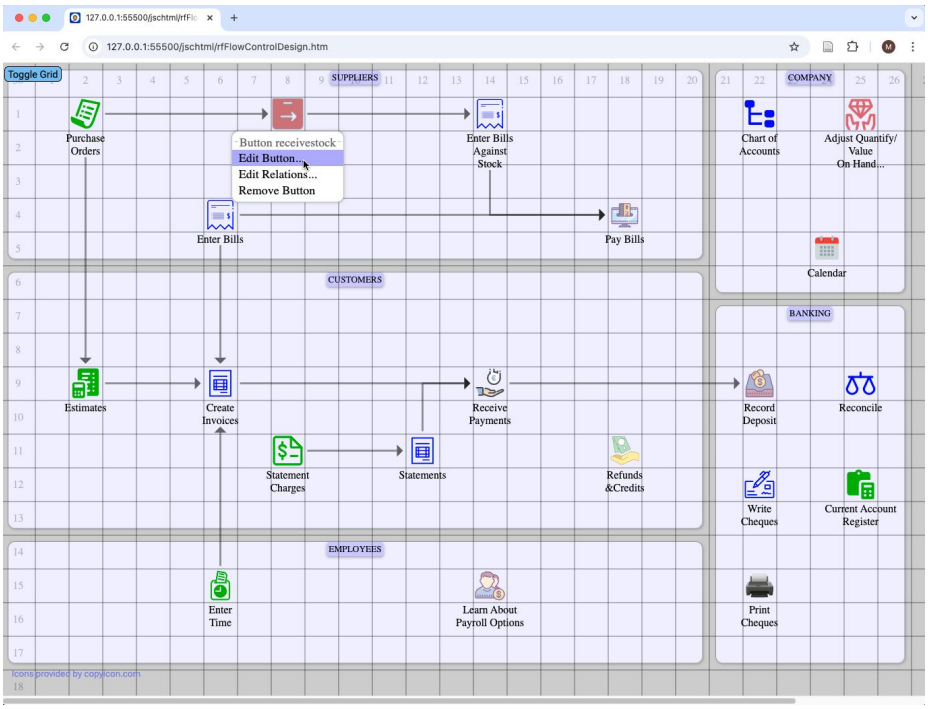

Figure 2.1 showing flow control in design mode

Consequently, the design interface has been developed to run in an external browser. To simplify this process, the examples implement a special design remote form called

rfFlowControlDesign, that is used in conjunction with rfFlowControl, the main form that implements a flow control named ‘jsoflow’.

Behind the Scenes

When the design remote form is opened in a browser, it activates the control’s design features within the super-class by executing

```
Do $cinst.$objs.jsoflow.$editmode.$assign(kTrue)
```

in its **$construct** method.

The design interface allows creating, editing and removing panels, buttons and their relations. Drag and drop is also supported, making it easy to arrange the panels and buttons. The control will automatically re-position any relations during dragging and dropping. When changes to the flow interface occur, the **evFlowChanged** is generated and is handled by rfFlowControl.$objs.jsoflow.$eventclient) which calls the custom server method **$updateclass** which is handled by both classes. The rfFlowControlDesign form calls the super-class method which saves the changes to its class data, whereas the design form is merely concerned with visually updating the design window in Omnis. Thus, when changes are made in the browser they are instantly reflected in the Omnis design window.

The interface changes are stored as JSON based text data in a single property called $flowdata.

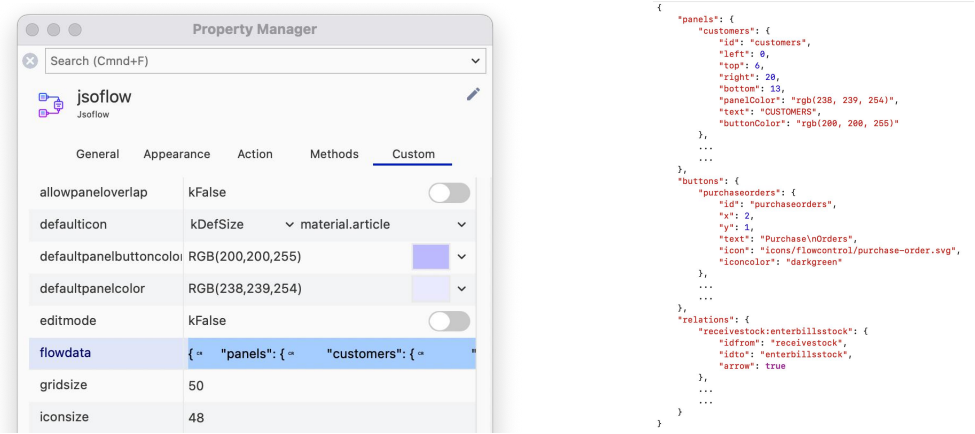

Figure 2.2 showing flow control property and sample property value

Making Changes

Right-clicking the icon or text of a button, the panel background, or the background canvas, will present a design menu with appropriate options. When editing a button or panel you are presented with a JSON editor for editing the item's properties.

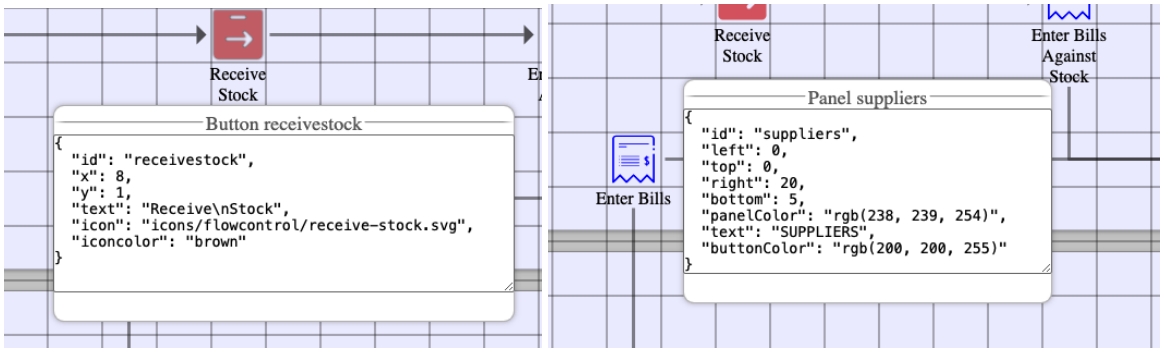

Figure 2.3 showing example JSON editors when editing a button and a panel properties

As you edit the properties for the item, live feedback is provided in the interface and if there are any JSON errors, error messages are displayed in the on-screen editor.

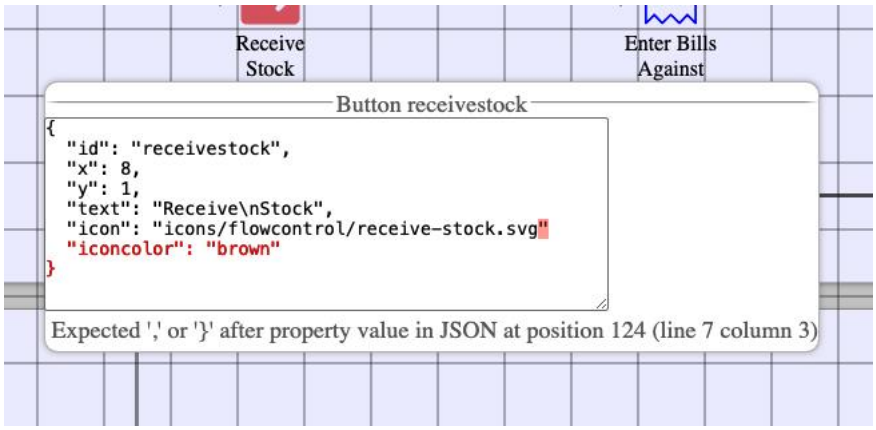

Figure 2.4 showing JSON editor with error message

Storing Multiple Interfaces

Because the entire flow interface is stored as a single JSON based property, it becomes possible to design various interfaces for different purposes and store them in a database. Changing the interface is then merely a matter of using a single remote form with a flow control and repeatedly assigning the $flowdata property.

Quick Start

We recommend you simply copy the two remote forms to your own library and clear the $flowdata property to get started designing your own interface. The method rfFlowControl.$objs.jsoflow.$eventclient handles, evClick, evContextClick and evFlowChanged (already discussed above) and $eventclient calls the methods $doclick, $docontextclick and $updateclass (also already discussed above) respectively. The $doclick and $docontextclick are the places where you need to plug in your runtime interface interaction.

$doclick

This method handles the click that is to execute something depending on what has been clicked. When kJSOFlowTargetButton or kJSOFlowTargetPanelButton are the reported targets of the click, the example retrieves additional information and calls $execute with the target ID and the display text.

$execute

The $execute method then uses the ID to search for an appropriate method to call by executing

```
$cinst.$exec_[pID].$cando
```

which it then calls if it exists.

This is probably where you would want to plug in your own method execution handling. If you are intending to store multiple JSON interfaces in your database, I recommend you use a separate class instance that implements the appropriate methods that go with a particular JSON interface.

The examples implement two methods called $exec_createinvoices and $exec_receivepayments. These are dummy interfaces for the ‘createinvoices’ and ‘receivepayments’ buttons. You would want to remove these if you use the example classes to design your own interfaces.

$docontextclick

This is where you would plug-in your context menu handling.

Desktop-Client

The jsoFlow control is of course designed to work with the Omnis Studio JS-CLient. However, it is possible to use JavaScript controls in your desktop client interface. If you want to use jsoFlow with the desktop client, you must read the chapter Designing Desktop Client which explains how you can use the JSC Bridge and oBrowser to achieve this.

Designing Desktop Client

Introduction

The jsoFlow control is implemented using the Omnis Studio JSON control editor and, although designed to work in browsers, can also be deployed in fat-clients using the JSCBridge SDK. Both these implementations were showcased at the EurOmnis 2025 conference and this chapter discusses the JSCBridge implementation.

Note: If you installed the JSCBridge previously, you will need to update the jsoFlow css, icons and script files inside the relevant jsclient_bridge folders from the latest jsoFlow release.

Note: The following instructions are intended for use with Studio 11.0 or later and are not compatible with Studio 10.x.

Installing the JSCBridge

The JSCBridge SDK can be downloaded from
https://github.com/OmnisStudio/Omnis-JSCBridge

Once downloaded, follow the instructions in the README file to install the SDK. As an additional aid we included our EurOmnis 2025 slides below which describe this process more visually.

Figure 3.1: Download JSCBridge

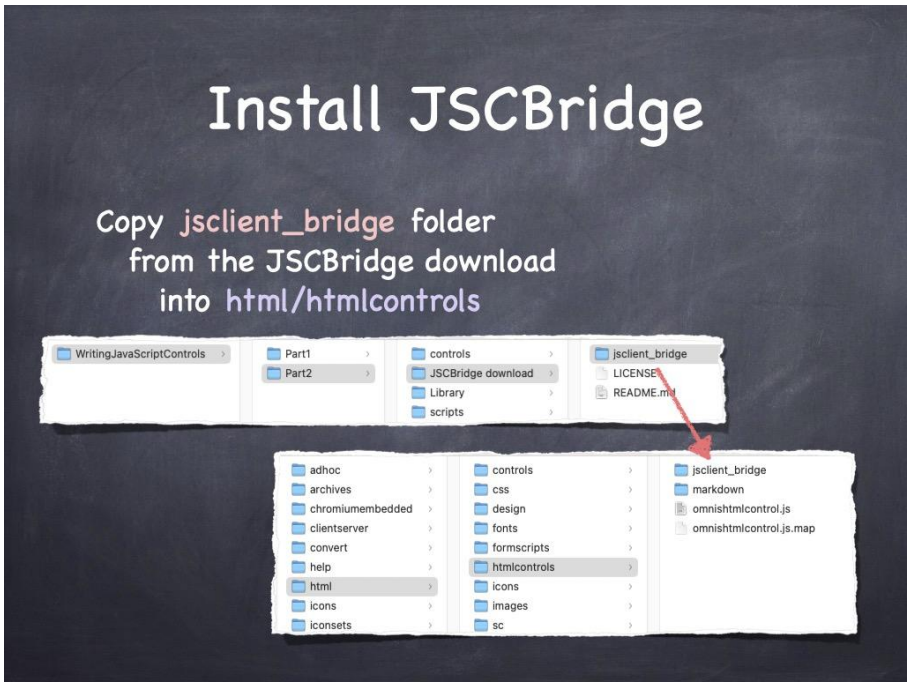

Figure 3.2: Copy jsclient_bridge folder to html/htmlcontrols

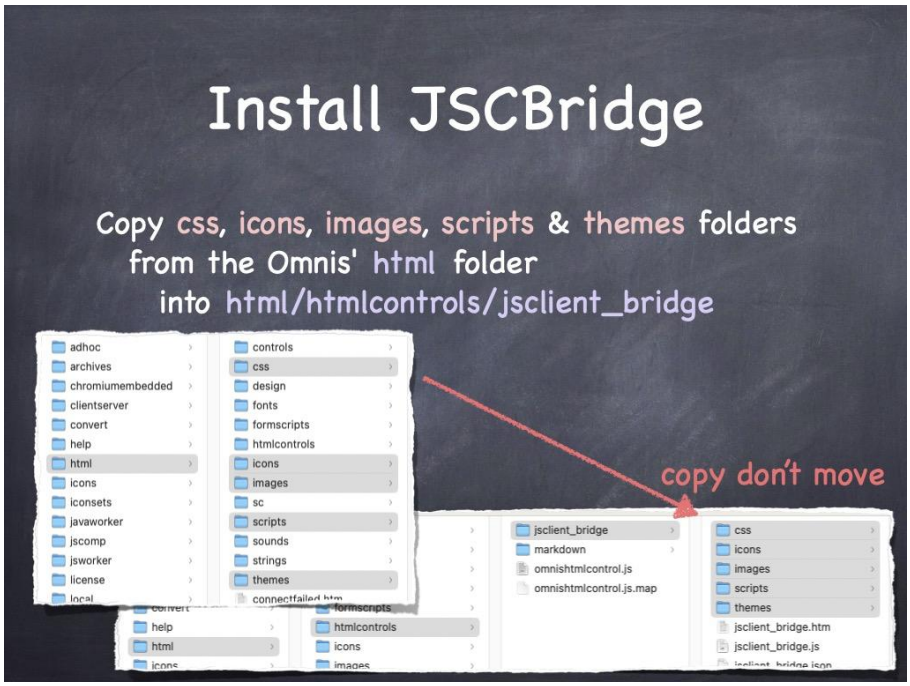

Figure 3.3: Duplicate the Studio css, icon, images, scripts & themes folders to jsclient_bridge

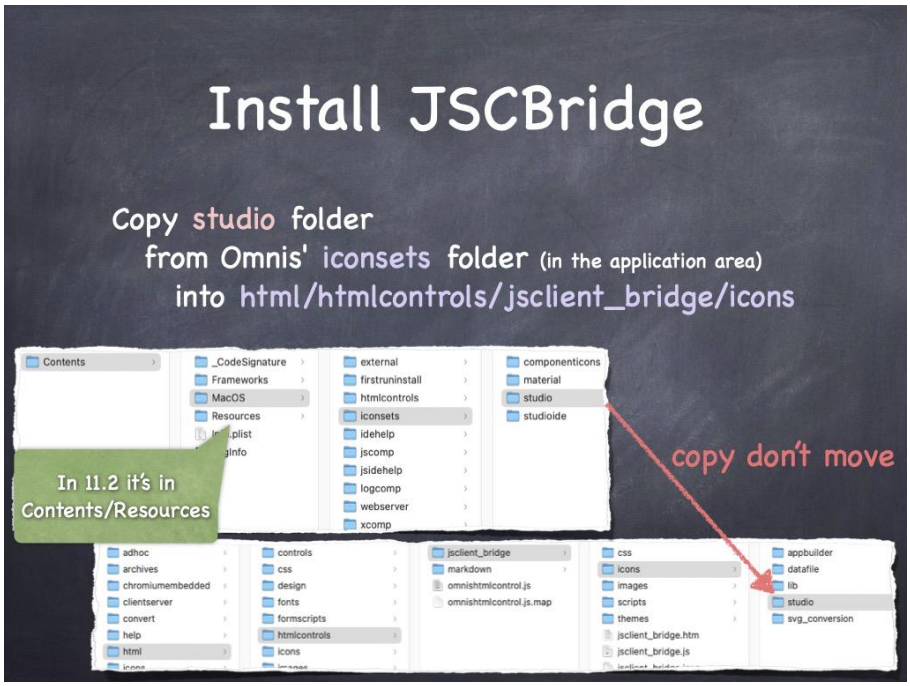

Figure 3.4: Duplicate the Studio iconsets folder to jsclient_bridge/icons

Note: any icons or icon groups that are not required by remote forms displayed by oBrowser through the JSCBridge SDK, can be omited.

Designing oBrowser

Once the JSCBridge is installed, we have to prepare a Desktop Application window with the oBroser control which will display our jsoFlow remote form. The examples already include a fully prepared window called winFlowControl. However, this window was designed to work with our local installation of Studio 11.1 37255 and you may need to tweak the odd parameter to get it to work with your installation of Studio. Consequently, we will work through what it would take to setup the oBrowser control from the beginning.

Initialize oBrowser Properties

After placing the oBrowser control on a desktop window, update the $urlorcontrolname, $htmlcontrolsusehttp and $htmlcontroloptions properties.

Figure 3.5: Set urlorcontrolname to jsclient_bridge

Figure 3.6: Set htmlcontrolsusehttp to true

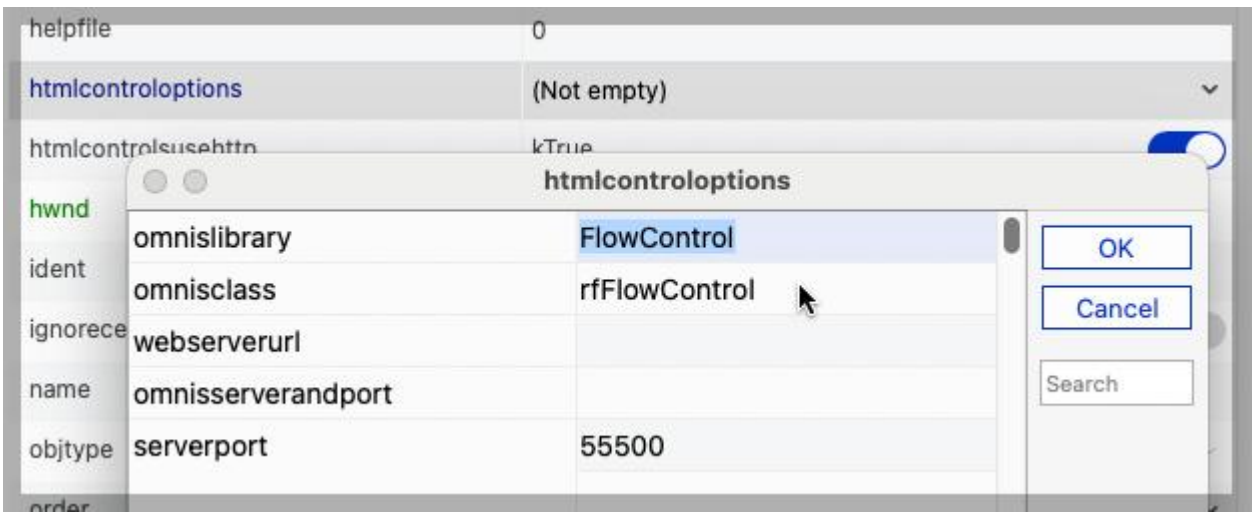

Figure 3.7: Set the htmlcontroloptions omnislibrary, omnisclass and serverport to the appropriate settings for your implementation

Desktop Client jsoFlow Communications

The remote form rfFlowControl already includes code that allows it to interact with and send messages to the oBrowser's evControlEvent. The rfFlowControl's $init method detects whether we are running inside a standard browser or within jsclient_bridge SDK.

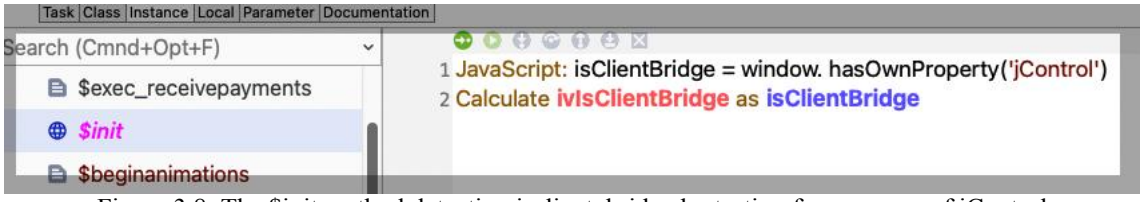

Figure 3.8: The $init method detecting jsclient_bridge by testing for presence of jControl

Once the remote form knows that it is running inside the JSClient Bridge SDK, The $doclick method, instead of calling $execute which would call the JSClient remote form server method, it sends information about the click via jControl to the oBrowser's evControlEvent.

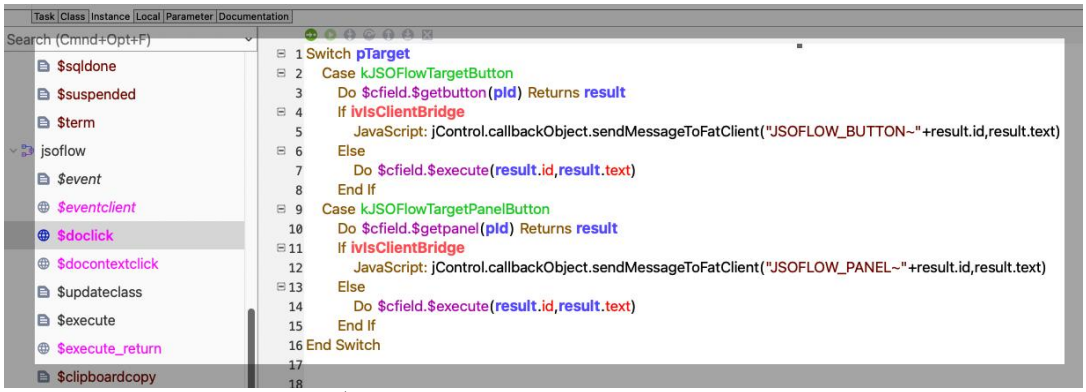

Figure 3.9: The $doclick method with both JSClient and jControl calls

Figure 3.10: oBrowser event method handling jsoFlow messages

Calling the JS Client

For the purpose of demonstrating the JSCLient Bridge callback mechanism, the examples also implement a simple callback to the client when the oBrowser event handles a jsoFlow click event. The callback simply changes the icon color of the button that was clicked.

For this purpose the rfFlowControl remote form implements the method htmlcontrolMessage for

receiving JSClient Bridge callbacks. In the examples, this method is set to be executed on the client as it wants to update the client interface, but it could also be server executed.

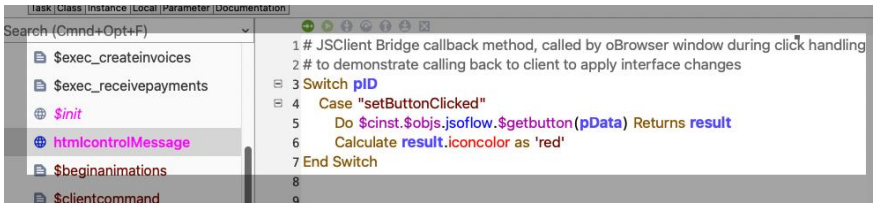

Figure 3.11: The callback method htmlcontrolMessage updating the specified button's icon color

The Desktop Client oBrowser window calls this method by calling

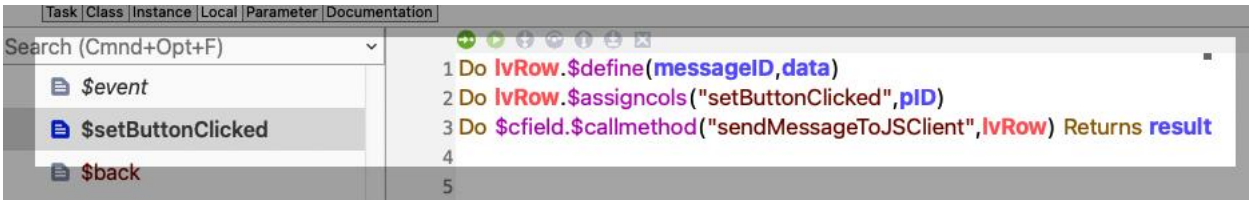

Figure 3.12: The oBrowser calling back to the JS Client with the 'setButtonClicked' message

Examples Reference

Introduction

This reference section lists the classes in the FlowControl.lbs example library. The classes are divided into two types. Those that are required for pure JS Client Implementations, and those classes that are required for Desktop Client Implementations.

Contents

JavaScript Client Classes - Consisting of the main example interface

Desktop Client Classes

JavaScript Client Classes

This section lists all the example classes required for the JS Client implementation.

rfFlowControl

This remote form implements the main example runtime interface.

Important methods are:

$init - this method tests for the presence of the jControl in the JS Client. If present the form is instantiated in oBrowser on a desktop window which changes how the jsoFlow event handling deals with clicks.

htmlcontrolMessage - this method receives callback messages from the desktop oBrowser window. Together with the methods in the desktop window winFlowControl it demonstrates how to manipulate the jsoFlow control from Omnis code within a desktop window.

jsoflow.*$eventclient* - implements jsoflow client event handling and calls other methods based on the received event

jsoflow.*$doclick* - called by $eventclient it implements client click handling for both JS Client and JSClient Bridge implementations.

jsoflow.*$docontextclick* - called by $eventclient it acts as place holder for implementing client context clicks.

jsoflow.$updateclass - server method called by $eventclient during design mode when changes have occurred to the jsoflow interface data which requires writing back to the objects class data.

jsoflow.$execute - server method called by $eventclient when end user clicks occur in JS Client implementations.

rfFlowControlDesign

This remote form subclasses rfFlowControl and activates the jsoflow control's design features.

Important methods are:

$construct - activates the design interface features of the jsoflow control in the super-class.

jsoflow.$updateclass - overridden server method which is responsible for updating the remote form design window when changes occur in the jsoflow class data. It calls the super-class method which writes the actual changes to its own class data.

Desktop Client Classes

This section lists all example classes for the JS Client Bridge implementation for use in desktop applications.

winFlowControl

This window contains the oBrowser control that displays and interacts with the jsoFlow remote form.

Important methods are:

oBrowser.$event - handles jsoFlow control events including end user clicks on icon buttons and panel buttons.

Control Reference

Introduction

This reference section lists the properties, constants, events and methods of the jsoFlow control.

Contents

Properties - The remote form object properties control the behaviour and appearance of the flow chart objects and background.

Constants - jsoFlow provides a small number of constants that are used with jsoFlow properties or methods. The constants are organised into functional groups and can be accessed from the Omnis Catalogue.

Events - The events listed here concern themselves with end user click events and changes to the flow chart data when designing a chart. Events are usually generated as a result of user actions.

Methods - The control's methods listed in this chapter apply to the remote form control that is instantiated on the client. Consequently, these methods should be called from within client executed methods in order to guarantee their behaviour.

jsoFlow Properties

Property	Description
$allowpaneloverlap	(boolean) when moving panels, allows them to overlap
$defaulticon	(icon) the default icon for new buttons (default = "material.article")
$defaultpanelcolor	(color) the default back color for new panels (default = “16707566”)
$defaultpanelbuttoncolor	(color) the default button color for new panels (default = "16763080")
$editmode	(boolean) enables runtime editor of the component, allowing you to add and edit panels, buttons and relations
$flowdata	(character) the JSON data containing all the panels, buttons and relations
$gridsize	(integer) size of grid for panel & button alignment
$iconsize	(integer) size of the icons for buttons in pixels (32-128, default = 48)
$menulabels	(multiline) localised labels for in-built menu buttons (default = "Panel,panel\rAdd Panel,add-panel\rRemove Panel,remove-panel\rEdit Panel...,edit-panel\rButton,button\rAdd Button,add-button\rRemove Button,remove-button\rEdit Button...,edit-button\rRelations,relations\rEdit Relations...,edit-relations")
$panelmargin	(integer) margin around the panels as percentage of $gridsize (0-80, default = 20)
$relationmargin	(integer) margin in pixels between relation lines and buttons (0-20, default = 5)
$texticonmargin	(integer) margin in pixels between button icons and button text (0-20, default = 3)

jsoFlow Constants

kJSOFlowTarget...

This group specifies the target during mouse events in the pTarget parameter.

See also: evClick, evContextClick

Name	Value	Description
kJSOFlowTargetNone	0	The control's background was clicked.
kJSOFlowTargetPanel	1	A panel background was clicked.
kJSOFlowTargetButton	2	A button was clicked.
kJSOFlowTargetPanelButton	3	A panel button was clicked.

kJSOFlowArrow...

This group specifies the style of the flow lines that connect the buttons.

See also: $setrelation()

Name	Value	Description
kJSOFlowArrowNone	0	A simple line is used without arrow heads.
kJSOFlowArrowArrow	1	The relation lines have arrow shaped tips.

jsoFlow Events

evClick

This event is generated when a flow chart item or background is clicked.

See also: kJSOFlowTarget...

Event Parameter	Description
pPosx	The horizontal position in pixel coordinates.
pPosy	The vertical position in pixel coordinates.
pTarget	What was clicked. One of the kJSOFlowTarget... constants
pId	The items ID or zero.

evContextClick

This event is generated when a flow chart item or background received a right-click/context click.

Event Parameter	Description
pTarget	What was clicked. One of the kJSOFlowTarget... constants
pId	The items ID or zero.

evFlowChanged

Sent when the user adds, removes or modifies a button, panel or relation.

Event Parameter	Description
pFlowdata	The updated JSON data of the flow control.

jsoFlow Methods

$deletebutton()

Syntax: $cfield.$deletebutton(cId)

Removes the specified button from the flowchart data.

Parameter	Description
cId	The id of the button to be removed
returns	<none>

$deletepanel()

Syntax: $cfield.$deletepanel(cId)

Removes the specified panel from the flowchart data.

Parameter	Description
cId	The id of the panel to be removed
returns	<none>

$deleterelation()

Syntax: $cfield.$deleterelation(cIdFrom,cIdTo)

Removes the specified relation line from the flowchart data.

Parameter	Description
cIdFrom	The id of the starting point button
cIdTo	The id of the ending point button
returns	<none>

$getbutton()

Syntax: $cfield.$getbutton(cId)

Limitations: Can only be used in a client executed method.

Returns the JavaScript object for the specified button. The returned JavaScript object can be modified by assigning the button properties using the Omnis Studio *Calculate* command. As this is a proper JavaScript object, the values that are assigned can be CSS constants as in the example below.

Example:

```
Do $cinst.$objs.jsoflow.$getbutton(buttonId) Returns result
Calculate result.iconcolor as 'red'
```

Parameter	Description
cId	The id of the button to be returned
Returns (Object Reference)	The returned JavaScript button object with the following properties: id, x, y, text, icon and iconcolor.

$getpanel()

Syntax: $cfield.$getpanel(cId)

Limitations: Can only be used in a client executed method.

Returns the JavaScript object for the specified panel. The returned JavaScript object can be modified by assigning the panel properties using the Omnis Studio *Calculate* command. As this is a proper JavaScript object, the values that are assigned can be CSS constants as in the example below.

Example:

```
Do $cinst.$objs.jsoflow.$getpanel(panelId) Returns result
Calculate result.panelColor as 'aqua'
```

Parameter	Description
cId	The id of the panel to be returned
Returns (Object Reference)	The returned JavaScript panel object with the following properties: id, left, top, right, bottom, panelColor, text, buttonColor.

$getrelation()

Syntax: $cfield.$getrelation(cIdFrom,cIdTo)

Limitations: Can only be used in a client executed method.

Returns the JavaScript object for the specified relation line. The returned JavaScript object can be modified by assigning the relation properties using the Omnis Studio *Calculate* command.

Example:

```
Do $cinst.$objs.jsoflow.$getrelation(fromId,toId) Returns result
Calculate result.arrow as kJSOFlowArrowNone
```

Parameter	Description
cIdFrom	The id of the starting point button
cIdTo	The id of the ending point button
Returns (Object Reference)	The returned JavaScript relation object with the following properties: idfrom, idto, arrow.

$setbutton()

Syntax: $cfield.$setbutton(cId, iX, iY, cText, cIcon, cIconColor)

This method will create a new button with the given ID or update the specified button if a button with the ID already exists. The button will be centred within the specified grid position.

Limitations: Can only be used in a client executed method.

Example:

```
Do $cfield.$setbutton("mybutton",5,5,"My Button","icons/flowcontrol/purchase-order.svg","darkgreen")
```

Parameter	Description
cId	The id of the button to be created or updated
iX	The horizontal grid position
iY	The vertical grid position
cText	The button text
cIcon	The partial URL to the Omnis icon
cIconColor	The color for the icon (only works with non-color SVG icons)

$setpanel()

Syntax: $cfield.$setbutton(cId, iLeft, iTop, iRight, iBottom, cPanelBackColor, cText, cButtonColor)

This method will create a new panel with the given ID or update the specified panel if a panel with the ID already exists. The panel will be sized to encompass the specified grid positions minus the $panelmargin.

Limitations: Can only be used in a client executed method.

Example:

```
Do $cfield.$setpanel("panel1",1,1,12,6,"lightslategray","Panel 1",
"darkslategray")
```

Parameter	Description
cId	The id of the panel to be created or updated
iLeft	The panel's left grid position
iTop	The panel's top grid position
iRight	The panel's right grid position
iBottom	The panel's bottom grid position
cPanelBackColor	The panel background's fill color
cText	The text for the panel button
cButtonColor	The panel button's fill color

$setrelation()

Syntax: $cfield.$setrelation(cIdFrom, cIdTo, iToArrow)

This method will create a new relation line with the given ID or update the specified relation line if a relation with the cIdFrom and cIdTo already exists.

Limitations: Can only be used in a client executed method.

Example:

```
Do $cfield.$setrelation("receivepayments","recorddeposit",kJSOFlowArrowArrow)
```

Parameter	Description
cIdFrom	The id of the starting point button
cIdTo	The id of the ending point button
iToArrow	Specifies the arrow style. One of the kJSOFlowArrow... Constants.

$togglegrid()

Syntax: $cfield.$togglegrid([bShowGrid])

This method will show or hide the flow chart grid.

Limitations: Can only be used in a client executed method.

See also: $gridsize

Parameter	Description
bShowGrid (optional)	Specifies wether to hide or show the grid. If this parameter is omitted, the grid state is toggled.