Client Side (JavaScript)

- ApplinX Server Actions
- Navigating between Input Fields
- Tables
- Design
- Keyboard Mapping
- ApplinX Web Application Event
- Browser Related Functions
- JavaScript Logging
- Page Validation
- ApplinX Web Application Windows
- HTML Controls
- Web Application Configurations
- Functionality
- Screen Locker
- User Exits

ApplinX Server Actions

Refer to the following tasks for further details on implementing these functions:

- Creating a Button / Hyperlink for Submitting a Host Key
- Enabling Sending Dup and FieldMark Characters to the Host

gx_SubmitKey(key)

Submits a PF key to the host.

parameters

key

The key to submit.

Example:

```
"[ENTER]", "[PF3]"
```

gx_SubmitKeyInPos(pos, keyName)

Sets the position of the cursor and then submits a PF key to the host.

Parameters

pos

The host screen position from which to send the host key. In a standard 24X80 host screen, the position can be any number between 1 and 1920.

keyName

The host key to send to the host.

For example

```
"[ENTER]"
```

SubmitCustomKey()

Prompts the user to specify a host key (for example "[PF3]") and submits it to the host.

gx_SetCursorPos(pos)

Sets the position from which the next PF key will be sent. In a standard 24X80 host screen, the position can be any number between 1 and 1920.

gx_ExecPath(pathName)

Executes an ApplinX Path Procedure for navigation purposes. When a folder is not specified, ApplinX server looks for the path in the directory of the current host screen. If the path cannot be found in the directory of the current host screen, ApplinX server looks for the path in the root directory. When a folder is specified, it needs to be relative to the root directory.

Parameters

pathName

The path to execute (case-sensitive).

For example:

```
ExecPath("gotoMainMenu")
ExecPath("Common/gotoMainMenu")
```

gx_systemRequest()

Prompts the client to type in the system request and then submits it to the host. Equivalent to gx_SubmitKey("[sysreq]").

gx_fieldmark()

Places the Field Mark symbol in the currently selected input field, and moves the cursor to the next unprotected position. Handling of the symbol is dependent on the application program. Equivalent to gx_SubmitKey("[fieldmark]").

gx_dup()

Simulates pressing the MainFrame duplicate (Dup) key. A duplicate symbol ("*") will appear in the currently selected input. Upon submitting the form, APX server will send the dup command. This only works on host fields that allow duplicating (in the host application). The host should support this functionality as well.

Navigating between Input Fields

Refer to the following task for further details on implementing these functions:

- Navigating between Input Fields

gx_home()

Moves to the first input element on the page. Equivalent to gx_SubmitKey("[home]").

gx_end()

Moves to last input element on the page. Equivalent to gx_SubmitKey("[end]").

gx_newLine()

Moves to the first input field you find in the rows below. Equivalent to gx_SubmitKey("[newline]").

gx_jumpToNextInput(currTextBox)

Sets the cursor on the next input field on the screen, relative to currTextBox.

For example:

```
<input type="button"
      onclick="gx_jumpToNextInput(this)"…
```

gx_jumpToPrevInput(currTextBox)

Sets the cursor on the previous input field on the screen, relative to currTextBox

For example:

```
<input type="button"
onclick="gx_jumpToPrevInput(this)"…
```

Tables

Refer to the following tasks for further details on implementing these functions:

- Retrieving Values from a Selected Row within a Table

gx_selectKey(elem)

Each table in a generated page keeps track of the selected row in the HTML table by a hidden field, which is added automatically.

This JavaScript function, when called inside a table row, updates this hidden field with the row number (by default) or row key to keep track of the selected row on the server-side.

Can be inserted by an on click event on a table row (tr) or a link/button placed inside a table cell.

Parameters

elem

Any element inside a table row (including the TR element itself).

gx_getSelectedKey(tableName)

Returns the selected key after `gx_selectKey` was called.

Parameters

tableName

The name/ID of the HTML table whose selected value we want to retrieve.

gx_isTableKeySelected(tableName)

Query if a row was selected. Used for validation to force a row selection.

Possible values: true/false

parameters

tableName

The name of the HTML table.

gx_markRow(obj,selectedRowCss)

Can be called in a table row (tr) event. This function marks the clicked row with the selected row css class name.

Can be used by ApplinX css: gx_markRow(this,"gx_tbl_selected")

Parameters

obj

The table row object selected.

RowCss

The css class to use to change the selected row.

Design

Refer to the following tasks for further details on implementing these functions:

- Enabling the User to Control the Font Size

gx_changeCss()

Toggles between the display styles specified in the web application configuration (In Emulation node>Color Set).

gx_changeCssExact(cssName)

Sets the display style to the specified style sheet.

Parameters

cssName

Style sheet URL.

gx_increaseFontSize()

Increases the font size used in the Web application. This function is called by the plus link in the page footer. The maximum font size is 20px. Refer to Enabling the User to Control the Font Size.

gx_decreaseFontSize()

Decreases the font size used in the Web application. This function is called by the minus link in the page footer. The minimum font size is 7px. Refer to Enabling the User Control the Font Size.

gx_changeFontSize(size)

Changes the font size used in the Web application to the specified size.

Parameters

size

An integer. The new font size in pixels.

Keyboard Mapping

Refer to the following tasks for further details on implementing these functions:

- Mapping Keyboard Keys to User Actions in Individual Pages

gx_AddKeyboardMapping(additionalKey,keyCode,functionElement,overrideExisting,cancelMapFunction)

This function allows developers to attach JavaScript functions to specific keyboard keystrokes.

Parameters

additionalKey

Possible values : 0-none, 1-CTRL, 2-ALT, 3-SHIFT

keyCode

An integer. The ASCII code of the pressed key. For example: Enter = 13

functionElement

The JavaScript element to execute.

overrideExisting

A boolean parameter indicating whether to override the existing XML keyboard mapping definition (gx_keyboardmapping.xml) .

cancelMapFunction

Optional. A function element that returns either true or false, determining whether to execute the keyboard mapping function or not. True, cancels the mapped function and False executes the function.

Examples

When CTRL+ESC are pressed, executes a confirm `function(myConfirmFunc)`. When confirmed, performs the `myLogoff` function.

```
gx_AddKeyboardMapping(1,27,myLogoff,true,myConfirmFunc)
```

When CTRL+ESC are pressed, executes the `myLogoff` function.

```
gx_AddKeyboardMapping(1,27,doLogoff,true)
```

Overrides the default PF3 action with a do-nothing action (`void(0)`).

```
gx_AddKeyboardMapping(0,27,void(0),true)
```

ApplinX Web Application Event

What is GXEvent?

GXEvent is a Wrapper for a browser event. This wrapper provides a cross browser event object.

Refer to the following task for further details on implementing these functions:

- Implementing & Controlling JavaScript Events using the gx_event Object

Properties

GXEvent.keyCode

Sets or retrieves the Unicode key code associated with the key that caused the event.

GXEvent.additionalKey

An integer representing the key that was pressed in addition to the key that triggered the event. Possible values: 0-none, 1-CTRL, 2-ALT, 3-SHIFT

GXEvent.element

Retrieves the element that fired the event.

Methods

GXEvent.cancel()

Cancels the event and stops it from bubbling further up the hierarchy of event handlers.

ApplinX Web Application Event Example

The following example will cancel the `OnKeyDown` whenever the [ENTER] key is pressed in a specific text area ("myTextArea"), prevent the page from being submitted and manually add a newline to the text area value:

Add the following to `globalOnKeyDown` function in the userExits.js file:

```
function globalOnKeyDown(gx_event){
        .........
        var win = gx_event.window;

        if (gx_event.keyCode==13 && gx_event.element.id=="myTextArea"){
                gx_event.cancel();
                GXBrowserUtil.getElement("myTextArea").value += "\r\n"
        }
.........
}
```

Assume your JSP/ASPX page has the following input: `<textarea row="5" id="myTextArea" ></textarea>`

Browser Related Functions

Refer to the following task for further details on implementing these functions:

- Retrieving Browser Information

GXBrowserUtil.isIE()

Boolean. Retrieves whether or not the browser is Microsoft Internet Explorer.

GXBrowserUtil.isIE7()

Boolean. Retrieves whether or not the browser is Microsoft Internet Explorer 7 or higher.

GXBrowserUtil.isMozilla()

Boolean. Retrieves whether or not the browser is Mozilla FireFox.

JavaScript Logging

GXLog is used to handle client side JavaScript logging. While mostly used by the ApplinX web application JavaScript engine, users can also add their own log messages. Refer to JavaScript Logger Engine for further details.

GXLog.debug(moduleName, message)

Adds a log message only when in debug mode.

Parameters

moduleName

This parameter can be an empty string. However, in order to make log messages more distinctive in the log, it is recommended to add a unique value to the `engineConfig debugModules` property in *config/gx_clientConfig.xml* and then use that value as the `moduleName` for debugging the JavaScript.

message

The log message.

GXLog.warning(moduleName, message)

Adds a log message only when in Warning mode.

GXLog.error(moduleName, message)

Adds a log message only when in Error mode.

Page Validation

GXValidator represents an array of validators, used to validate the form's data before submitting it to the host.

Refer to the following task for further details on implementing these functions:

- Validating your Data

GXValidator.registerValidator(Validator)

Adds a validator to the array.

GXValidator.clearValidators()

Clears the validator array.

Page Validation Example

The following example is automatically added to every JSP/ASPX page generated by ApplinX:

```
var pageValidator = new function(){
        this.validateField = function(inputField){
                if (inputField.name == "FIELD_A"){
                        if (inputField.value == ""){
                                return "Field cannot be empty";
                        }
                }
                // ...
        }
}
function pageOnLoad(){  // register validator function
        GXValidator.registerValidator(pageValidator);
}
```

This validator checks that "Field_A" has a value. Otherwise, it returns an error message.

ApplinX Web Application Windows

What is gx_window JavaScript class?

GXWindow interacts with ApplinX framework's server side gx_window(.NET)/getGXWindow() (JSP). In addition, gx_window can perform actions on the browser side using the following methods:

Refer to the following task for further details on implementing these functions:

- Handling Web Application Windows using the gx_windows Object

gx_window.open(page,width,height,left,top)

Locks the current main window, opens a new pop-up window and loads the document specified by a given URL (page).

Parameters

page

A page name (including relative folders) to load into the pop-up window.

width

The window's width.

height

The window's height.

left

An integer. The position of the window's left edge, in relation to the desktop.

top

An integer. The position of the window's top edge, in relation to the desktop.

gx_window.resizeTo(width,height)

Sets the size of the window to the specified width and height values.

gx_window.moveTo(posX,posY)

Moves the screen position of the upper-left corner of the window to the specified `posX` and `posY` position.

gx_window.setField(fldName,fldVal)

Sets the value or InnerHTML property of field `fldName`, to the specified `fldVal.`

gx_window.close()

Closes the window. If the main window is closed the host session will be disconnected.

gx_window.getOpener()

In pop-up windows, retrieves the parent window element.

gx_window.loadPageFull(PageName)

Loads a URL. The PageName represents the URL of the document that is to be loaded.

gx_window.loadPage(PageName)

Loads a URL. The PageName represents the URL of the document that is to be loaded.

ApplinX Web Application Windows Example

Main Window HTML and Script

```
<script>
     function getData(){
                  gx_window.open("myData.jsp",200,300,0,0) ;

                //  .NET
                // gx_window.open("myData.ASPX",200,300,0,0);
              }
      </script>
<input id="myInput"  value=" "/>
<a href="#" onclick=" getData ();">open Win</a>
```

Pop-Up Window

```
function setMainWindowInput(){
            // Set the value of an Input on the main window
                  gx_window.getOpener().
                       gx_getElement("myInput").value="someValue";
                  // Unlock the main window before closing the pop up
                        gx_window.getOpener().gx_unlockScreen();
                  // Close the pop up window
                        gx_window.close();
   }

<a href="#" onclick=" setMainWindowInput ();">close Win</a>
```

HTML Controls

gx_updatePagePart(panelId)

This functionality can only be used once this feature has been selected in the Framework Configuration Editor. This feature is used for updating a specific part of the page (can only be used once within a single page). This is useful when sorting/scrolling within host tables or working with tabs. Call this function just before a submitting a key to the host.

Parameters

panelId

The server side control ID, whose content we wish to update.

gx_isValidInputElement (inputElement)

Boolean. Returns "true" if `inputElement` is either a combo box, text area or an input tag of the following type: radio, check box, password or text. Otherwise returns "false".

Parameters

inputElement

The input field to validate.

gx_getElement(elemId)

Similar to `document.getElementById("id")`, this function retrieves the required HTML object from the active frame according to its identifier (`elemtId`).

gx_eraseEOF()

Removes text from an input field, leaving only the text before the cursor.

gx_showCalendar(Title,dateFieldName1,Format1,dateFieldName2,Format2, dateFieldName3,Format3)

Displays the calendar window, attaches it to a specific input or inputs (up to three inputs: day, month and year) and sets the format the selected date will be displayed with.

Parameters

Title

Calendar window title.

dateFieldName1/dateFieldName2/dateFieldName3

An Input field used to display the date (or part of it).

Format1/Format2/Format3

The format in which the date will be displayed.

The fields dateFieldName2/dateFieldName3 and Format2/Format3 are optional and required only when there is more than one date field and date format.

Examples

```
<img onclick="gx_showCalendar('Select a date','BirthDate','MM-dd-yy')"…
```

```
<img onclick="gx_showCalendar('Select a date','day','dd' ,month,'MM', 'year','yyyy')…
```

Possible date format examples:

MMM, d, yyyy - March 22, 2009

dd/MM/yy - 22/03/09

MM-yyyy - 03-2009

Web Application Configurations

gx_getCookie(name)

Retrieves the value of the cookie <name>.

Parameters

name

The name of the cookie.

GXLAFHandler.COLOR_CSS_COOKIE

Saves the user preferred Style sheet.

GXLAFHandler.FONT_SIZE_COOKIE

Saves the user preferred font-size for Instant pages

GXLAFHandler.RESOLUTION_COOKIE

Saves the user's screen width (this is useful when developers set the Instant configuration's font-size node to "Dynamic by screen resolution").

gx_setCookie(name, value)

Sets the value in a specific (<name>) cookie.

Parameters

name

The name of the cookie.

GXLAFHandler.COLOR_CSS_COOKIE

Saves the user preferred Style sheet.

GXLAFHandler.FONT_SIZE_COOKIE

Saves the user preferred font-size for Instant pages

GXLAFHandler.RESOLUTION_COOKIE

Saves the user's screen width (this is useful when developers set the Instant configuration's font-size node to "Dynamic by screen resolution").

value

The value to be set to the cookie.

Functionality

Refer to the following tasks for further details on implementing these functions:

- Printing a Capture of the Host Screen
- Opening the File Transfer Dialog Box
- Deploying the Printlet

gx_printScreen()

Opens a customized print screen Web page that contains a text representation of the current host screen. This is used in order to print a snapshot of the current host screen.

gx_openFtpDialog

Opens the file transfer dialog box.

gx_openNewBrowser (url)

Opens a new browser window with its own process. Useful when you need to open multiple sessions from the same Web application. The ApplinX emulation component ActiveX Object in the current page needs to be uncommented (see *index.jsp/aspx* page in the HTML emulation demo).

Parameters

url

The URL destination of the new window.

gx_hidePrinter/gx_showPrinter()

Hides/shows the Printlet window/frame. Will only work if the ApplinX application is printer enabled. Refer to Deploying the Printletfor more details about ApplinX printers.

gx_postBack(func)

Use this function to execute server side functionality.

Parameters

func

A string. The server side function name to be executed.

Example

Create a button in your JSP page that when clicked runs a server side function called `myServersideFunc`:

```
<gx:input type="button" value="GO!" onserverclick=" myServersideFunc" />
```

The server side code must have a method like this:

```
public void myServersideFunc(){
    ……
}
```

Screen Locker

The ApplinX framework contains a built-in feature of a screen locker. The purpose of a screen locker is to indicate to the user by means of a message, that the application is processing your request, and that you are blocked from interfering with the current process by repressing a button/link or keyboard PF/ENTER.

Refer to the following tasks for further details on implementing these functions:

- Activating the Screen Locker
- Handling the Screen Locker on the Page Level

gx_lockScreen(); gx_unlockScreen()

Locks/unlocks the screen locker. Can be used from client-side code to activate/cancel the screen locker. When the screen locker is locked, no additional host keys/links/buttons can be pressed until the screen locker is unlocked.

gx_disableScreenLocker(); gx_enableScreenLocker()

Disables/Enables the screen locker ("Please wait" message) before/after redirecting to a new page.

Usage: When the target page downloads a file, screen locking is not needed. Without using the following functionality the current page will get "stuck" with the "Please wait" message, which causes a lock on the page.

```
// Disable the screen locker
      gx_disableScreenLocker();
      // redirect to the excel download page, which opens a save/open/cancel window dialog
      location.href = "excel.jsp"; // or "excel.aspx"
      // Enable the screen locker after 2 seconds for the next page actions
      window.setTimeout(gx_enableScreenLocker,2000);
```

User Exits

The user exits file is located in the js directory of your web application. It allows you to add functionality to your pages without disrupting the framework's events.

Registering Events

gx_engine.registerEvent

Registering an event to a page means that the function is executed when the event bubbles up to the document level.

```
gx_engine.registerEvent(GXEventType.FOCUS,globalOnBlur);
```

gx_engine.attachInputTagsEvent

Registering an event to an input means that the function is executed when the input fires the event:

```
gx_engine.attachInputTagsEvent (GXEventType.FOCUS,  globalInputOnFocus);
```

gx_engine.setLabelFocusFunction

This function sets the LabelFocus function:

- labelFocus: This function is executed when the label is clicked.
- labelBlur: This function is executed once the label has been clicked once and then another object became in focus.

```
gx_engine.setLabelFocusFunction(labelFocus,labelBlur);
```

Global Functions

Code placed in these global functions affects the whole application and therefore must be used carefully.

For example (taken from userExits.js):

```
function globalOnLoad(gx_event){
             // use win.<SOMETHING> to access the page tags
             // for example: win.document.GX_form
             var win = gx_event.window;

             // activate page scope function if exits
             activateIfExists(gx_event,gx_event.window.pageOnLoad);

             .........
}
```

Notice that every global function contains the line:
`activateIfExists(gx_event,gx_event.window.functionName)`which executes the user functions: <functionName> (pageOnLoad in this case), if indeed the function exists on the page. This enables running the mentioned function (in the example above: pageOnLoad) in a specific page.

For Example: Adding the following function to a generated JSP/ASPX in the ApplinX web application will cause this function to automatically execute whenever the page is loaded and pop-up the alert message:

```
function pageOnLoad(){
        alert("Hello World!!");
}
```

Available User Exits

Global function	Page level function	Description
globalOnLoad	pageOnLoad	Occurs immediately after a page is loaded.
globalOnKeyDown	pageOnKeyDown	Occurs when a keyboard key is pressed.
globalOnKeyPress	pageOnKeyPress	Occurs when the user presses an alphanumeric key.
globalOnKeyUp	pageOnKeyUp	Occurs when a keyboard key is released.
globalOnFocus	pageOnFocus	Occurs when an object gets focus.
globalOnBlur	pageOnBlur	Occurs when an object loses focus.