Javascript and HTML in ABAP report

Using JavaScript and HTML in an ABAP report.

What is Javascript?

JavaScript is the most popular scripting language on the internet, and works in all major browsers, such as Internet Explorer, Firefox, Chrome, Opera, and Safari.

- JavaScript was designed to add interactivity to HTML pages
- JavaScript is a scripting language
- A scripting language is a lightweight programming language
- JavaScript is usually embedded directly into HTML pages
- JavaScript is an interpreted language (means that scripts execute without preliminary compilation)

Everyone can use JavaScript without purchasing a license

What You Should Already Know

Before you continue you should have a basic understanding of the following:

HTML (HyperText Markup Language)

With HTML you can create your own Web site.

Are Java and JavaScript the same?

NO!

Java and JavaScript are two completely different languages in both concept and design!

Java (developed by Sun Microsystems) is a powerful and much more complex programming language - in the same category as C and C++.

What can a JavaScript do?

JavaScript gives HTML designers a programming tool - HTML authors are normally not programmers, but JavaScript is a scripting language with a very simple syntax! Almost anyone can put small "snippets" of code into their HTML pages

JavaScript can put dynamic text into an HTML page - A JavaScript statement like this: `document.write("<h1>" + name + "</h1>")` can write a variable text into an HTML page

JavaScript can react to events - A JavaScript can be set to execute when something happens, like when a page has finished loading or when a user clicks on an HTML element

JavaScript can read and write HTML elements - A JavaScript can read and change the content of an HTML element

JavaScript can be used to validate data - A JavaScript can be used to validate form data before it is submitted to a server. This saves the server from extra processing

JavaScript can be used to detect the visitor's browser - A JavaScript can be used to detect the visitor's browser, and - depending on the browser - load another page specifically designed for that browser

JavaScript can be used to create cookies - A JavaScript can be used to store and retrieve information on the visitor's computer

We will be using the Class CL_GUI_HTML_VIEWER and some its methods for embedding Javascript and HTML in ABAP Report.

Let us understand the important methods of the CL_GUI_HTML_VIEWER "class."

1.) **set_registered_events**: Use this method to register the events of the control.

   CALL METHOD my_control->set_registered_events
   EXPORTING events = events
   EXCEPTIONS cntl_error = 1
   cntl_system_error = 2
   illegal_event_combination = 3.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>events</td>
<td>Table of events that you want to register for the custom control <code>my_control</code></td>
</tr>
</tbody>
</table>
The table `events` is a list of the events that you want to register. It is defined with reference to table type `CNTL_SIMPLE_EVENTS`. The table type is based on the structure `CNTL_SIMPLE_EVENT`, which consists of the following fields:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVENTID</td>
<td>Event name</td>
</tr>
<tr>
<td>APPL_EVENT</td>
<td>Indicates whether the event is a system event (initial) or an application event (X).</td>
</tr>
</tbody>
</table>

The values that you assign to the field `EVENTID` are control-specific and therefore described in the documentation of the individual controls.

2.) `show_url`: You use this method to display data in the SAP HTML Viewer. There is no longer a distinction between this method and the `show_data` method. From Release 4.6C onwards, you should **always use this method** to display data in the HTML Viewer, regardless of its source.

```plaintext
CALL METHOD html_viewer->show_url
EXPORTING url = url
frame = frame
in_place = in_place
EXCEPTIONS cntl_error = 1.
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Opt.</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td></td>
<td>URL address of the page that you want to display.</td>
</tr>
<tr>
<td>frame</td>
<td>X</td>
<td>Name of the HTML frame in which you want to display the page.</td>
</tr>
<tr>
<td>in_place</td>
<td></td>
<td>Indicates where the page should be displayed. Possible values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>'X' (default): Page is displayed in the SAPGUI window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>' ': A separate browser window is opened to display the page.</td>
</tr>
</tbody>
</table>

3.) `load_data`: Use this method to send data from the ABAP program to the presentation server. On the presentation server, the data is available under the URL that you specify.

```plaintext
CALL METHOD html_viewer->load_data
EXPORTING url = url
type = type
subtype = subtype
size = size
IMPORTING assigned_url = assigned_url
CHANGING data_table = data_table
EXCEPTIONS dp_invalid_parameter = 1
dp_error_general = 2.
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>url</td>
<td>URL with which you want to access the data.</td>
</tr>
<tr>
<td>type</td>
<td>Data type as a MIME type (for example, text).</td>
</tr>
<tr>
<td>subtype</td>
<td>Data subtype as a MIME type (for example, html).</td>
</tr>
<tr>
<td>size</td>
<td>Data size in bytes.</td>
</tr>
<tr>
<td>assigned_url</td>
<td>If you did not assign a URL in the url parameter, the system places the URL that it assigned into this parameter. If you did assign the URL yourself, the parameter contains the URL from the url.</td>
</tr>
<tr>
<td>data_table</td>
<td>Table containing data.</td>
</tr>
</tbody>
</table>

Now as we are clear about these methods, we can use Javascript and HTML in our ABAP report. It is a just a four step process:

1.) **Step 1**: Assigning of the control events. This is done with the help of the method "set_registered_events". For eg(given below): We will have to assign the events of our control to the object reference(ob_ref) created of class CL_GUI_HTML_VIEWER.
2.) Step 2: Now we need to prepare our table which contain our Javascript/HTML code.(Table "html" in below example).

3.) Step 3: Next we have generate a HTML page with the contents of our table(html). We will use the method "load_data". This method will create an HTML page an return the url of the page in the "assigned_url" attribute.

4.) Step 4: Finally the url received from the "load_data" method is displayed with help of the method "show_url".

Example:
REPORT z2qcr_test_javascript.

CLASS tcl_class DEFINITION

PUBLIC SECTION.
  METHODS: main, on_html_event FOR EVENT sapevent OF cl_gui_html_viewer
    IMPORTING action frame getdata.
ENDCLASS  "tcl_class DEFINITION"

CLASS tcl_class IMPLEMENTATION

METHOD main.
  DATA cb1cl TYPE REF TO tcl_class.
  DATA html TYPE v網站tab.
  DATA url TYPE C LENGTH 255.
  CREATE OBJECT cb1cl.
  EXPORTING
    parent = cl_gui_container->screen0.
  DATA events TYPEcntl_simple_events.
  DATA event TYPE cntl_simple_event.
  DATA eventid = cb1cl->m_id_saperent.
  DATA appl_event = 'Y'
  APPEND event TO events.
  CALL METHOD cb1cl->set_registered_events.
  EXPORTING
    events = events.
  SET HANDLER ne->on_html_event FOR cb1cl.
  APPEND '<html>' TO html.
  APPEND '<SCRIPT type="text/javascript">' TO html.
  APPEND 'function fn_msg() {' TO html.
  APPEND 'var v_msg = document.getElementById("msg"), value,' TO html.
  APPEND 'if(v_msg) {v_msg = prompt("Enter Message:");}' TO html.
  APPEND 'alert(v_msg), window.open("http://www.google.com","NewWindow");}' TO html.
  APPEND 'function fn_valid() {' TO html.
  APPEND 'var v_check = document.getElementById("msg"), value,' TO html.
  APPEND 'if(v_check) {v_result = confirm("Message is empty! Do you want to continue?");}' TO html.
  APPEND 'if(v_result) {document.getElementById("msg"), focus();}' TO html.
  APPEND 'else {fn_msg();}' TO html.
  APPEND '</SCRIPT>' TO html.
  APPEND '<body bgcolor="#999999">' TO html.
  APPEND '<font face="arial" size="2">' TO html.
  APPEND '<p>Click button to display Javascript message-box.</p>' TO html.
  APPEND '<br>' TO html.
  APPEND '<font>' TO html.
  APPEND '<form name="form1" action="SAPEVENT save">' TO html.
  APPEND 'Enter Message:' TO html.
  APPEND '<input type="text" id="msg" name="msg" onblur="fn_valid();">' TO html.
  APPEND '<br>' TO html.
  APPEND '<input type="button" onclick="fn_msg();", value="Display">' TO html.
  APPEND '</form>' TO html.
  APPEND '</body>' TO html.
  cb1cl->load_data( IMPORTING assigned_url = url
                  CHANGING data_table = html ).
  cb1cl->show_url( url = url ).
ENDMETHOD  "main"

METHOD on_html_event.
  DATA l_string TYPE string.
  l_string = getdata.
  MESSAGE 100 (400) HTML STRING.

Output:

**Display a javascript popup through a report**

Click button to display Javascript message-box.

Enter Message: 
Display

When focus come out of the textbox - Confirm box is displayed.

**Display a javascript popup through a report**

Click button to display Javascript message-box.

Enter Message: 
Display

When OK button is clicked - Prompt box is displayed

**Display a javascript popup through a report**

Click button to display Javascript message-box.

Enter Message: 
Display

The message entered in the prompt box is then displayed in the alert box.
This is just a simple example of using Javascript/HTML. Similarly for more functionality and complex requirements we can also embed AJAX, call urls/web address, or use stylesheets etc as per our requirements.