

Dynamic Web Pages - Integrating JavaScript into a SAS® Web Application

Caroline Bahler, ASG, Inc.

Abstract

The use of dynamic web pages in either Internet or Intranet applications is rapidly becoming the hottest SAS® application development topic. One component of a dynamic web page is the use of the JavaScript language. JavaScript is a language that has tremendous potential for enhancing web applications, especially client-side data validation. The objective of this paper is to introduce the JavaScript language, illustrate how web page developers are using it, and discuss how the JavaScript language could be incorporated into a web application powered by SAS.

Introduction

Development of a dynamic web application involves the integration of the client-side "front-end" HTML document and the "guts" or server-side where the actual data handling and analysis by SAS occurs. Web publishing requires the following components a browser, a web server and a HTML document. The HTML document can contain both text and images to navigate from one web page to another. Hypertext markup language (HTML) is an evolving language that uses tags to signal internet browsers of different types on different hardware environments to present a document identically over all of those environments². A tag is a command that will be interpreted by the browser to display a portion of a document in a specific manner.

JavaScript allows the developer to modify and expand the functionality of a HTML tag. This paper assumes the reader has a working knowledge of HTML and is familiar with the layout of HTML documents. The objective of this paper is to introduce the JavaScript language, illustrate how web page developers are using it, and discuss how the JavaScript language could be incorporated into a web application powered by SAS.

Dynamic HTML and JavaScript

"Dynamic" HTML is an evolving term that changes as the "technology" of the browser changes. What encompasses "dynamic" HTML is function of the person defining it. Originally, a "dynamic" HTML page would display the current date or post the number of hits at a given web site. Currently, "dynamic" HTML can be defined as the combination of HTML and the JavaScript language to allow documents to change as the user interacts with the document. In other words, "dynamic" HTML in this instance refers to the information displayed by a document changing as the user interacts with the document. Another typical use of "dynamic" HTML is data validation of form data, on the client side instead of the server. In this paper, a "dynamic" web application is defined as:

1. an application that interacts with the server (utilizing CGI or SAS/INTRNET®) to request information from a SAS data set or other data source
2. manipulates or analyzes that data
3. publishes a report or graphic analysis to the client⁴.

The JavaScript language was developed as an interpreted, object oriented language that is integrated into most web browsers³. JavaScript language usage can occur on both the client and server sides of a web application, i.e. it is considered to be "cross platform" language. Both the client and server sides of JavaScript language consist of a core language and a specific set of objects that are specific for the platform. Client-side objects are specific for the browser (Netscape or Internet Explorer) and the browser itself will determine which objects are available. Server-side JavaScripts are compiled into executable files. These executable files allow for faster access and can affect the final HTML displayed by the browser. Client-side usage of JavaScript can be through either embedded JavaScript statements in a HTML document or calls to external JavaScript files and is compiled when the HTML is

files and is compiled when the HTML is displayed. This paper will only be concerned with client-side JavaScript use.

One main function of JavaScript is to allow a document developer to greatly extend the functionality of a document. Cascading style sheets and layering both affect what and way text or images are displayed in an HTML document. This functionality is similar to the use of the SCL within a SAS/AF application to hide or unhide an object on the screen. For example, the current date can be displayed in a document by creating a short JavaScript AND then accessing it within the HTML portion of a document. Some caveats when using JavaScript⁵ -

1. Case is **CRITICAL** unlike HTML which is not case sensitive
2. Internet Explorer (IE) and Netscape **DO NOT** support all of the same JavaScript functionality
3. Older versions of browsers either do not support or support "earlier" versions only of JavaScript. Therefore, the script block within a HTML document needs to be written so that it will be hidden from non-supporting browser versions.
4. The script should be placed in or just after the Head section of the HTML document to ensure the script compiles before the Body section is interpreted. Another option is to place the script in a separate file that is called at runtime.
5. JavaScript prefers that all object placement on the page be denoted including image height and width.
6. How the JavaScript is formatted within a text editor is important. Avoid using the "wrap around" option since it could corrupt the meaning of your script unintentionally.

JavaScript can be divided into sections - event handlers and language.¹ Event handlers (Table 1) allow the developer to define a set of actions to occur when a specific type of user interaction (event) occurs. User interactions or events include clicking of a mouse key, typing characters into a text box or the loading or unloading of a HTML document. Event handlers can be defined as part of the object call (HTML tags) they are modifying or within a JavaScript.

Figure 1 is an example of a page that contains onClick, onMouseOver and onMouseOut event handlers.

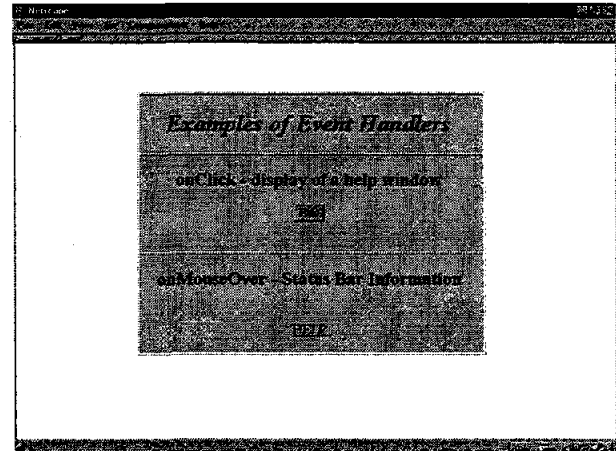


Fig 1. Event handler example.

```
<HTML>
<BODY>
<CENTER>
<br> <br> <br>

<TABLE BORDER WIDTH="60%"
  BGCOLOR="#FFCC33" >
  <TR><TD><BR><CENTER><B><I>
  <FONT COLOR="#990000" SIZE=+3>
  Examples of Event Handlers</FONT>
  </I></B></CENTER><BR>
  </TD></TR>

  <TR><TD>
  <CENTER><BR><B><FONT SIZE=+2>
  onClick - display of a help window
  </FONT></B><BR>
  <FORM>
  <INPUT type="button" name="help" value="Help"
  onClick="msgWindow=window.open('help.html',
  window2,'resizable=no,width=300,height=400')"
  >
  </FORM></CENTER><BR>
  </TD></TR>
```

The onClick action defined within the <INPUT> tag above is to open a new window (note width and height are specified) that will contain the page help.html. This new window will be opened in a new browser session and is completely separate from the original page (designated by the window2 option). Note that the size of the window and whether or not the new window is resizable where also specified. Figure 2 illustrates the new window.

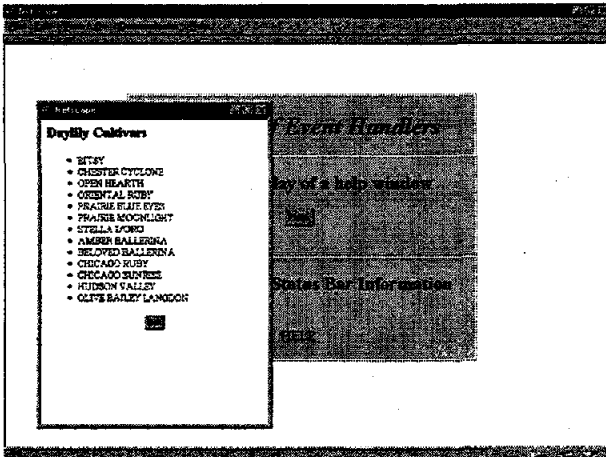


Fig 2. Help window defined within the **onClick** event handler.

```
<TR><TD>
<CENTER>
<BR><H4><FONT SIZE=+2>
onMouseOver - Status Bar Information
</FONT></H4><BR>

<CENTER><FONT COLOR="#3333FF" SIZE=+1>
<A HREF="help.html"
onClick="msgWindow=window.open('help.html','window2',
'resizable=no,width=300,height=400')
onMouseOver="window.status='Daylily Cultivar
List';return true"
onMouseOut="window.status='';return true"> HELP
</A>
</FONT></CENTER>
<BR>
</TD></TR>
</TABLE>

</CENTER>
</BODY>
</HTML>
```

In the second example on this page an action is defined for when the mouse cursor passes over that anchor. The use of **onMouseOver** in the above example is to put a message in the status line at the bottom of the window, i.e. this example is similar to the "mouse tip" in SAS/AF®. In addition, to **onMouseOver** two other, **onMouseOut** and **onClick**, event handlers are present within the **<A>** (anchor) tag. The **onMouseOut** defines the action to take place when the mouse cursor does not pass over the anchor and **onClick** defines what happens when the anchor is selected. Figure 3 illustrates text displayed in the status line of the browser.

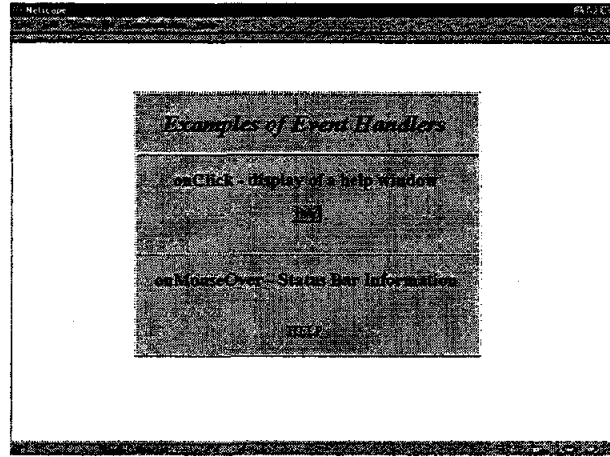


Fig 3. Message displayed in status line.

JavaScript language consists of two separate entities - statements and functions. A function can be defined as a sets of statements bundled into a single call. JavaScript language also contains a set of predefined functions also known as methods and can be grouped into element associated and character or number associated. JavaScript language statements can be classified into 3 main groups - number or text manipulation, display (browser, document and graphics) or action modifiers and input/output modifiers. In addition, there is a set of specialized JavaScript statements that deal with the browser environment, Java applets, e-mail capabilities and user privileges and security.

A JavaScript function can be thought as being analogous to a SAS macro. A function contains arguments passed to it (optional) and a set of JavaScript statements or function calls. Typically a function takes the following form:

```
Function fn (a1,...,an) {
  statements
}
```

where all bolded items are required and
 fn = name of function
 a1,...,an = arguments these are optional

The main difference between a SAS macro and a JavaScript function is how a function operates. A function will only process one set of conditional operations, for example an if ... else or do ... while. In comparison, a macro can be a complex set of conditional actions with one

action depending upon another action. For example, a survey form is created using text boxes, select box, radio button and check box HTML form elements and requires many of these fields to contain entered information or selections. To validate that each of the required fields a function needs to be created for each form element used to determine if the correct action was taken.

An example of this type of client-side validation is found in Example 1. Figure 4 illustrates the data entry screen created by the HTML document in example 1.

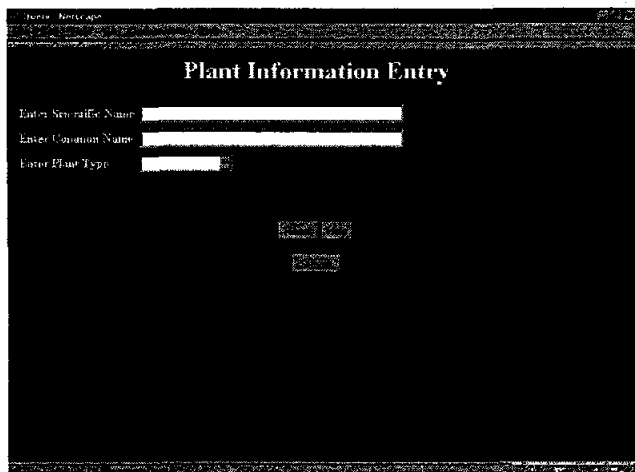


Fig 4. Data entry screen.

The application requirements defined that each field on the screen had to be valid before the user was allowed to submit the information to the server to be processed. A valid value was defined as information entered into the text boxes and a plant type selected for the select element.

To accomplish this -

1. Form and each element within the form was named.
2. A function was created for each element.
3. Each function contained a single conditional statement specifying that the length of the text entry was greater than 0 or first selection within the select box was not made. Note that the options within the select box are number 0 ... n.
 - If a valid value was present then a variable was set to 1
 - If no text or option selected an alert message is displayed.

4. A final function was created that calls each element related function and test for data entry or selection within each of the form elements.
 - If all of the validation conditions are met then the link to the next page is defined and the information entered is passed to the web server.
 - If all of the validation conditions are not met then an alert message is displayed.

Important - JavaScript does **NOT** recognize multiple conditional statements in a single function.

For example -

```
function selectFile () {  
  if (document.f1.f.options.selectedIndex == 0)  
    return document.f1.action="f0.html"  
    else return false  
  if (document.f1.f.options.selectedIndex == 1)  
    return document.f1.action="f1.html"  
    else return false  
  if (document.f1.f.options.selectedIndex == 2)  
    return document.f1.action="f2.html"  
    else return false  
}
```

In the function above only the first if ... else statement will resolve. The other if ... else statements will be ignored.

In contrast the set of functions below will resolve and move the user to the correct page.

```
<SCRIPT>  
function f0 () {  
  if (document.f1.f.options.selectedIndex == 0)  
    return document.f1.action="f0.html"  
    else return false  
}  
function f1 () {  
  if (document.f1.f.options.selectedIndex == 1)  
    return document.f1.action="f1.html"  
    else return false  
}  
function f2 () {  
  if (document.f1.f.options.selectedIndex == 2)  
    return document.f1.action="f2.html"  
    else return false  
}  
function selectFile () {  
  f0()  
  f1()  
  f2()  
}
```

Integration

The appropriate use of JavaScript in a web application is determined by the functionality required by that application. In an application

that requires interaction with SAS and uses HTML and a browser as a front-end, JavaScript can be utilized. It's primary functionality will be to validate that selections were made before the request is transferred to SAS. In addition, if the information required is a number or date with a specific range that range can be validated.

References

1. Netscape Communications Corp. (1997) JavaScript Guide, Version 3.0. Mountain View, CA, Netscape Communications Corp.
2. Graham, Ian S. (1996), HTML Sourcebook. A Complete Guide to HTML 3.0, Second Edition. New York, New York, John Wiley and Sons, Inc.
3. Kent, Peter and Muller, Kent. (1997). Official Netscape JavaScript 1.2 Programmer's Reference. Netscape Press. Mountain View, CA, Netscape Communications Corp.
4. Hoyle, Larry (1997), Interfacing the SAS System with the World Wide Web. Proceedings of the Twenty-second SAS Users Group International Conference, 22:723-732.
5. Java Goodies: Java Requirements. http://www.javagoodies.com/js_req.html

SAS, SAS/AF and SAS/INTRNET are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. Netscape is a registered trademarks or trademarks of Netscape, Inc. in the USA and other countries. Microsoft Internet Explorer is a registered trademarks or trademarks of Microsoft, Inc. in the USA and other countries. ® indicates USA registration.

Author can be contacted at
Caroline Bahler
ASG Inc
2000 Regency Parkway
Suite 355
Cary, NC 27511
919-467-0505
E-Mail: ccb@asg-inc.com

Table 1. JavaScript Event Handlers^{1,3}

Event Handler	Usage
onAbort	Used to stop loading of an image.
onBlur	Used when input focus on windows and all form elements is removed.
onChange	Used with text areas, text fields and select boxes to dictate an action when the value of a field is changed.
onClick	Used to define what action takes place when a user clicks on a form element or link.
onDbClick	Used to define an action when a form item or link is double clicked.
onDragDrop	Used to define an action when a file or shortcut is dragged from the operating system desktop onto a HTML document in the browser.
onError	Used to define an action when an error takes place during the loading of an image or HTML document.
onFocus	Used to define an action when a user places the cursor onto an input, select or submit element within a window or frame or the window itself.
onKeyDown	Used to define an action when a key is pressed within a document, image, link or text box.
onKeyPress	Used to specify multiple actions when a key is held down within a document, image, link or text box.
onKeyUp	Used to define an action when a key is let up within a document, image, link or text box.
onLoad	Used to define an action within a frame, image, layer, or window when a document is completely loaded.
onMouseDown	Used to define an action when a mouse button is pressed within a document, link or button.
onMouseMove	Used to define an action when the mouse is moved. These actions are enabled through event capturing.
onMouseOut	Used to define what action takes place when the mouse pointer is moved off of a link.
onMouseOver	Used to define what action takes place when a user moves the mouse pointer over a link.
onMouseUp	Used to define an action when a mouse button is released within a document, link or button.
onMove	Used to define an action when a window or frame is moved.
onReset	Used to define an action in addition to resetting form objects when the RESET form object is pressed.
onResize	Used to define an action by a window or frame when a window is resized.
onSelect	Used to define an action when the text within a text or textarea object is highlighted (selected).
OnSubmit	Used in conjunction with onClick to define what action takes place when a user submits a form.
OnUnload	Used to define an action by a window or frame when a page is exited.

*Event handlers in bold are used in examples.

Example 1 - Use of JavaScript language for client-side data validation.

```

<html>

<head>
<title>Query</title>
</head>

<script>
①function cS () {
    if (document.form1.sname.value.length > 0) return s=1
    else {
        alert("Enter Scientific Name")
        return false}
    }
②function cC () {
    if (document.form1.cname.value.length > 0) return c=1
    else {
        alert("Enter Common Name")
        return false}
    }
③function cT () {
    if (document.form1.plantType.options.selectedIndex > 0) return t=1
    else {
        alert("Select Plant Type")
        return false}
    }

function checkNames () {
    t=0
    s=0
    c=0
    cS()
    cC()
    cT()

    if (s+t+c < 3) return alert("Please Enter Missing Information")
    else {document.form1.action="frame.html"
        document.form1.submit()
        return false}
}
</script>

<body TEXT="#FFFFFF" BGCOLOR="#3366FF" LINK="#0000EE" VLINK="#551A8B"
ALINK="#FF0000">

<b><font SIZE="+3">

<p align="center">Plant Information Entry</font></b> </p>

<form name="form1" method="GET">
<table>
    <tr>
        <th align="left">Enter Scientific Name</th>
        <td align="left"><input type="text" name="sname"① size="40"></td>
    </tr>
    <tr>
        <th align="left">Enter Common Name</th>
        <td align="left"><input type="text" name="cname"② size="40"></td>
    </tr>

```

```

<tr>
  <th align="left">Enter Plant Type</th>
  <td align="left"><select NAME="plantType"Ⓢ size="1">
    <option NAME="plantType" value="none" selected> </option>
    <option NAME="plantType" value="perennial">perennial </option>
    <option NAME="plantType" value="annual">annual </option>
    <option NAME="plantType" value="grass">grass </option>
    <option NAME="plantType" value="shrub">shrub </option>
    <option NAME="plantType" value="conifer">tree - conifer </option>
    <option NAME="plantType" value="deciduous">tree - deciduous </option>
  </select></td>
</tr>
</table>

<div align="center"><center><p>&nbsp;</p></center></div>
<div align="center"><center><p>
<input type="reset">
<input type="button" name="help" value="Help"
  onClick="msgWindow=window.open('help.html','window2',
    'resizable=no,width=300,height=400')">
<br>
<br>
<input type="button" value="Check" onClick="checkNames()"Ⓣ>Ⓣ
</p></center></div><div align="center"></div>
</form>
</body>
</html>

```

Web Resources

Below is a listing of web sites with information on Java Script Language and other general information about publishing on the web. All of these references have some interesting and useful material and point to other web sites of interest.

JavaScript

1. FAQs, Technical Resources - NetscapeWorld
<http://www.netscapeworld.com/netscapeworld/common/nw.jumps.html#javascript>
2. The WDWL: Introduction to Authoring JavaScript
<http://www.stars.com/Authoring/JavaScript/Intro>
3. Cut-N-Paste JavaScript
<http://www.infohiway.com/javascript/indexf.htm>
4. JavaScript Reference
<http://developer.netscape.com/docs/manuals/communicator/jsref/index.htm>
5. JavaScript Guide
<http://developer.netscape.com/docs/manuals/communicator/jsguide4/index.htm>
6. Netscape JavaScript Documentation
<http://developer.netscape.com/docs/manuals/index.html?content=javascript.html>
7. Java Goodies JavaScript Repository
<http://www.javagoodies.com>
8. DevEdge Online - JavaScript Developer Central
<http://developer.netscape.com/tech/javascript/javascript.html>

General Information

9. Welcome to developer.com
<http://www.developer.com>
10. Coolnerds - Home Page
<http://www.coolnerds.com>
11. The Web Developer's Virtual Library
<http://www.stars.com>
12. Web Review -- The Legion of DOM: Exploring the Document Object Model
<http://webreview.com>
13. Webreference.com (sm) - The Webmaster's Reference Library
<http://www.webreference.com>
14. DevEdge Online - Home Page
<http://developer.netscape.com>
15. DevEdge Online - Documentation
<http://developer.netscape.com/docs/index.html>
16. Dynamic HTML in Netscape Communicator
<http://developer.netscape.com/docs/manuals/communicator/dynhtml/index.htm>
17. The HTML Goodies Home Page
<http://www.htmlgoodies.com>