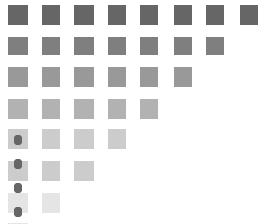
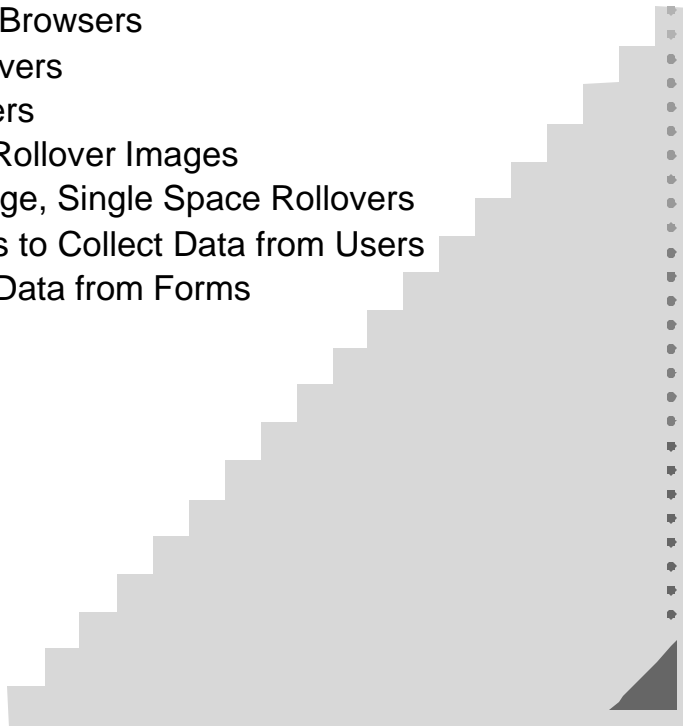


Information Resources



JavaScript Workshop 2

- ?? Event Handlers
- ?? Redirecting Browsers
- ?? Image Rollovers
- ?? Text Rollovers
- ?? Preloading Rollover Images
- ?? Multiple Image, Single Space Rollovers
- ?? Using Forms to Collect Data from Users
- ?? Processing Data from Forms



Marshall School of Business
University of Southern California

Table of Contents

Table of Contents	2
EVENT HANDLERS	3
Event Handlers & Embedding JS in HTML	4
Event Handlers Example 1: Redirecting a Link	4
Event Handler Example 2: Image Rollovers	5
Event Handlers Example 3: Triggering a Rollover from a Text Link	6
Event Handlers Example 4: Preloading Rollover Images	8
Event Handler Example 5: Multiple Images Change A Single Image Space	10
Event Handler Example 6: Using Forms to Accept User Input	11
Event Handler Example 6: Using Forms to Accept User Input	12
Variations on Example 6	14
Event Handler Example 7: Using Forms to Accept User Input	15
Variation on Example 7: Using Properties (Variables)	17
Variation on Example 7: Using the “onChange” to Look for TAB or ENTER	17

EVENT HANDLERS

"Event Handler" are built-in JavaScript commands that wait for the user to do something (like clicking the mouse). Once the user does the anticipated event, then the event-handler is triggered. Most event handlers are used to call functions. If it helps, think of event handlers at the little bells that ring when you open the door to a country store. The bells tell the shopkeeper to come to the counter. The table below lists and explains the event handlers available.

Event	Triggering Event
onAbort	Occurs when the user aborts loading an image by clicking on a link or by clicking the STOP button. Available on images.
onBlur	(The object loses the keyboard focus. (Usually tabs out of a field.) Available on: Text fields, Text areas, Select Lists
onChange	The user changed the object and then tabs out. (Used for field validation) Available on: Text Fields, Text Areas, and Select Lists
onClick	The user clicked a: button, checkbox, radio, or link.
onError	The script encountered an error with JavaScript or within an tag.
onFocus	The object obtains keyboard focus through a tab or mouse click. Available on: Text Fields, Text Areas, and Select Lists
onLoad	Occurs after a window or frame has loaded. Is an attribute of the <BODY> or <FRAMESET> tag Available on: Frames & Windows.
onMouseover	The mouse is hovering over a hyperlink.
onMouseout	The mouse moved away from a hyperlink.
onReset	Occurs when the user clicks a form's RESET button. Associated with the <FORM> tag.
onSelect	The user selected the contents of an object
onSubmit	The user submitted a form
onUnload	The user left the window or frame. Is an attribute of the <BODY> or <FRAMESET> tag Available on: Frames & Windows.

Event Handlers & Embedding JS in HTML

Event Handlers Example 1: Redirecting a Link

The script below places the JavaScript redirection directly in an HTML link. If the user's browser supports JavaScript, when they click the link "Welcome to Our Site!", they will be taken to the "JSPage.htm". If the user's browser does not support JavaScript, when they click the link "Welcome to Our Site!", the user's browser will take them to the "NoJSPage.htm".

Syntax:

?? The semicolon (;) allows you to follow one JavaScript command for another.

?? 'JSPage.htm' is in single quotes ' ' because they are nested within double quotes and the browser may not interpret the code correctly if there would double quotes within double quotes.

```

<HTML>
<HEAD>
<SCRIPT LANGUAGE=JAVASCRIPT
TYPE="TEXT/JAVASCRIPT">
</SCRIPT>
</HEAD>
<BODY>

<A HREF="NoJSPage.htm"
onClick="window.location='JSPage.htm';
return false">
Welcome to Our Site!</A>

</BODY>
</HTML>

```

Without JavaScript, the standard HTML link would look like this:

```
<A HREF="NoJSPage.htm">Welcome to our site!</A>
```

The JavaScript code consists of the lines below. They will only be read if the user's browser does support JavaScript.

```
onClick="window.location='JSPage.htm';
return false">
```

?? The "onClick" line tells the browser to load a page called "JSPage.htm".

?? The "return false" line tells the browser not to process the user's click and therefore, not load the "NoJSPage.htm".



Event Handler Example 2: Image Rollovers

A “rollover” is when you hover your mouse over a picture and the picture changes to another picture but changes back when you move the mouse away.

In the example below, an image named “FROG” is used as a hyperlink to the Los Angeles Zoo. The file “**frogblue.gif**” displays initially on the page but if the user moves the mouse over the image, “**frog-red.gif**” appears and when the user moves the mouse away, “**frogblue.gif**” is reloaded.

```
<HTML><HEAD><TITLE>Example of a Rollover</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT"></SCRIPT>
</HEAD>
<BODY>
<A HREF="http://www.lazoo.org"
  onMouseover="document.FROG.src='frog-red.gif'"
  onMouseout="document.FROG.src='frogblue.gif'">
<IMG SRC="frogblue.gif" NAME="FROG" BORDER=0></A>
</BODY>
</HTML>
```

Without the JAVASCRIPT, the HTML code alone would be the code below where the first section indicates the page to load and the second section displays a picture as the hyperlink.

```
<A HREF="http://www.lazoo.org"><IMG SRC="frogblue.gif" NAME="Frog" BORDER=0></A>
```

onMouseover	The event is triggered when the mouse is over the picture. In other words, “When the mouse is over the image area, the event will be...”
onMouseout	The event is triggered when the mouse moves away from the picture. In other words, “When the mouse moves away from the image area, the event will be...”
document.src	“document” is the initial code used to do something to the document window. In this case, changing the image source called FROG to the gif file listed.
src (used in HTML and JavaScript)	The SRC attribute allows you to reference other files and insert them into your HTML pages. Here it is used to load an object into the image area. On this page, it is used in two ways: Initially, it is used with the tag to load the <i>frogblue.gif</i> image the user sees when the page is first opened. It is used again in the JavaScript code whenever the user places the mouse on or moves away from the image area <i>named</i> “FROG”. The JavaScript code aborts the HTML code and changes what is loading.

NAME	The NAME attribute of the tag allows you to name an image.
FROG	This is the name we gave to our image area. It is like a placeholder. (You can name it anything you like). "onMouseover" and "onMouseout" then specifies what FROG will be equal to.

Event Handlers Example 3: Triggering a Rollover from a Text Link

In the proceeding example, when the user places the mouse over an image, the image turns into another image. In this example, the user will place the mouse over a text link causing the images to roll over.

Phase 1: This code simply creates a hyperlink to cnn and places an image on the page.

```
<HTML>
<HEAD>
<TITLE>Example of a text linked Rollover</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT"></SCRIPT>
</HEAD>
<BODY>

<IMG SRC="frogblue.gif" NAME=FROG border=0>
<A HREF="http://www.cnn.com">See what's new</A>

</BODY>
</HTML>
```

Phase 2: Creating the Text Rollover

This code modifies the hyperlink code to include the rollover by adding the two "onmouse" lines.

```
<HTML>
<HEAD>
<TITLE>Example of a text linked Rollover</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT"></SCRIPT>
</HEAD>
<BODY>
<IMG SRC="frogblue.gif" NAME=FROG border=0>
<A HREF="http://www.cnn.com"
  onMouseover="document.FROG.src='frog-red.gif'"
  onMouseout="document.FROG.src='frogblue.gif'"
>See what's new</A>

</BODY>
</HTML>
```

FROG is a name given to the image created by the tag. It is a place holder.

scr is a attribute what allows you to load items into a place holder.

In common terms, this might read: *"When the mouse is over the hyperlink, load the file called "frog-red.gif" into the image area called FROG and when the mouse is moved away from the hyperlink, load a file called "Frogblue.gif" into the image area called FROG."*

Event Handlers Example 4: Preloading Rollover Images

The rollover examples covered earlier work but there is a short delay loading the rollover image the first time it is activated because the image must be downloaded to the user's computer. The example below shows how to load all rollover images into the user's computer cache when the page is initially accessed.

```
<HTML>
<HEAD>
<TITLE>Example of a text linked Rollover</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
```

```
RED = new Image
BLUE = new Image
```

```
RED.src="frog-red.gif"
BLUE.src="frogblue.gif"
```

These four lines merely load the gif files into the users hard drive (cache), they do not display on the screen at this point.

Using the "new Image" operator, the JavaScript first creates two separate image objects that we have named **RED** and **BLUE** that act as placeholders.

```
</SCRIPT>
</HEAD>
<BODY>
<IMG SRC="frogblue.gif" NAME=FROG border=0>
<A HREF="http://www.cnn.com"
```

The **src** property is used to fill the two objects (RED & BLUE) with the gif files being used.

```
    onMouseover="document.FROG.src=RED.src"
    onMouseout="document.FROG.src=BLUE.src"
```

```
>See what's new</A>
</BODY>
</HTML>
```

```
<IMG SRC="frogblue.gif" NAME=FROG border=0>
```

This line displays the image "frogblue.gif" on the screen and names its area FROG.

`onmouseover="document.FROG.src=RED.src"`

"onmouseover" states that when the mouse is over the image area named "FROG", load the object called "RED". (Which we defined above as "frog-red.gif". "onmouseout" works similarly except that when the mouse is moved away from the image area called "FROG", an object named "BLUE" is loaded.

new	A JavaScript operator that lets you create an instance of a user-defined object type or of one of the built-in object types that has a constructor function.
Image	<p>The JavaScript runtime engine creates an Image object corresponding to each IMG tag in your document. It puts these objects in an array in the document.images property. You access an Image object by indexing this array.</p> <p>To define an image with the IMG tag, use standard HTML syntax with the addition of JavaScript event handlers. If specify a value for the NAME attribute, you can use that name when indexing the images array. To define an image with its constructor, use the following syntax: new Image (width, height)</p>

Event Handler Example 5: Multiple Images Change A Single Image Space

In this example, when the mouse is placed over one of the images, a description about the person pictured is displayed below where the text currently being displayed is. When the mouse is moved away from an image, the default image text returns.

This example is also different from the prior examples in that the rollovers are inserted in the image code and not a hyperlink.

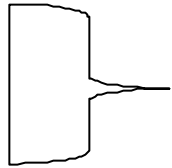


Place your cursor on a picture to read a short description of your new instructor.

Note that this can also be achieved by writing the text to a text box in a form. The text used in this example is really a picture of text.

```
<TITLE>Example multiple images and a single rollover</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
```

```
PeeweeBio = new Image
HomerBio = new Image
LisaBio = new Image
OutText = new Image
```



These four lines create four new image objects which act as placeholders.

```
PeeweeBio.src = "PEEWEE-BIO.jpg"
HomerBio.src = "HOMER-BIO.jpg"
LisaBio.src = "LISA-BIO.jpg"
OutText.src = "OutText.jpg"
```



These four lines fill the placeholders with files and pre load the images on the user's hard drive.

```
</SCRIPT></HEAD><BODY>
<P align="Center">
```

This line loads the image of Lisa (gabor2.jpg).

```
<IMG SRC="gabor2.jpg"
  onmouseover="document.BIOTEXT.src=LisaBio.src"
  onmouseout="document.BIOTEXT.src=OutText.src">
```

The "onMouseout" line loads an object named "OutText" (the instructions for the user saved as an jpg file). into the area named "BIOTEXT" when the mouse moves away from the image of Lisa.

The "onmouseover" line loads an object named "ListBio" (Lisa's description) into an area named "BioText" (the text below the images) when the user places the mouse on the image. (The "BioText" area was created below)

```
<IMG SRC="peewee.gif" height=150
  onmouseover="document.BIOTEXT.src=PeeweeBio.src"
  onmouseout="document.BIOTEXT.src=OutText.src">
```

```
<IMG SRC="homer.gif" height=150
  onmouseover="document.BIOTEXT.src=HomerBio.src"
  onmouseout="document.BIOTEXT.src=OutText.src">
```

```
<P align="Center">
<IMG SRC="Instructions.jpg" NAME=BIOTEXT >
```

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

This line displays the instructions for the user (saved as jpg file) and names the image area "BIOTEXT" which is used throughout this code.

These two sections function essentially the same as the code which gave us Lisa's description except they give us Peewee's and Homer's info.

Event Handler Example 6: Using Forms to Accept User Input

In this example, the user supplies the length and width of a rectangle and the rectangle's area is then returned to the screen.

The user will enter the length and width into form fields and then click the "Compute" button. Clicking the "Compute" button invokes our function through the use of an "event handler" called "onClick". "onClick" passes the user's input to the function, which in turn processes the input and returns the result to another form field for the user to see.

This example also demonstrates another method of placing JavaScript into HTML documents. If you recall from the beginning of this handout, we spoke of four different ways of placing JavaScript into an HTML document. Certain HTML tags can accept certain JavaScript Event Handlers as an attribute. As you will see, HTML's `<INPUT>` tag which is used in HTML forms, can accept the JavaScript "onClick" event handler as one of its attributes.

```

<HTML><HEAD><TITLE>Find the Area</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">
function Area()
{
  L=document.MyForm.Length.value
  W=document.MyForm.Width.value
  document.MyForm.Answer.value=L*W
}
</SCRIPT></HEAD><BODY>
<FORM NAME="MyForm">
Enter the length: <INPUT TYPE="text" NAME="Length" ><BR>
Enter the width: <INPUT TYPE="text" NAME="Width" ><BR>
<INPUT TYPE="button" VALUE="Compute" onClick="Area()">
<INPUT TYPE="reset" value="Clear"><BR>
The area is: <INPUT TYPE="text" NAME="Answer" >
</FORM></BODY></HTML>

```

```
function Area()
```

_____ This line creates a new function called "Area". New functions are defined with the "function" keyword and the name of the new function is followed with parenthesis ().

```
{
L=document.MyForm.Length.value
W=document.MyForm.Width.value
```

L and W are both properties (variables) of the Area function. Here we are setting them equal to the values found in the form fields "Length" and "Width" from the form called "MyForm".

Where the form is. Name of the Form Name of the Field Where the value is.

The two lines above are following the coding hierarchy of JavaScript. The syntax is:

document.NameOfTheForm.NameOfTheField.value

"value" is understood by HTML as whatever the user types in the field . You could initially set a field's value by using HTML's value code: <Input Type="Text" Name="Length" value=5>

```
document.MyForm.Answer.value=L*W
}
```

_____ This line of code returns the result of the user's input to the "Answer" field in the form.

The script above basically says: set the value of the Answer field in the form called "MyForm" which is found in this document to L*W.

L and W have already been defined above as what the user typed into the Length and Width fields.

```
<INPUT TYPE="button" VALUE="Compute" onClick="Area()">
```

This is the only line of code left that contains JavaScript. The other lines are simple HTML. This line shows how to call a function using an Event Handler (onClick). It is the name of the event handler set equal to the name of the function. The syntax is:

NameOfEventHandler="NameOfFunction()

To sum it all up, the script flows as follows: The user enters values into the "Length" and "Width" fields. When they click the "Compute" button, the event handler invokes the function which in turn grabs the values out of "Length" and "Width" and assigns them to the variables "L" and "W". The value of the "Answer" field is then defined to be "L*W".

Variations on Example 6

There are usually several different ways to define a function. We did not have to use variables at all when we defined the function, we could have also defined our function as:

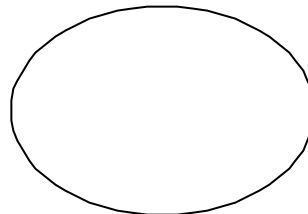
```
function Area()  
{  
document.MyForm.Answer.value=document.MyForm.Length.value*document.MyForm.Width.value  
}
```

Or, you can also set "answer" to a variable and the function could be defined like this:

```
function Area()  
{  
L=document.MyForm.Length.value  
W=document.MyForm.Width.value  
A=L*W  
document.MyForm.Answer.value=A  
}
```

Note that the code below would not work. This is because you are setting the value of the "Answer" field equal to "A" before you have defined what "A" is. You must define what a variable is before you can use it.

```
function Area()  
{  
L=document.MyForm.Length.value  
W=document.MyForm.Width.value  
document.MyForm.Answer.value=A  
A=L*W  
}
```



Event Handler Example 7: Using Forms to Accept User Input

When the user types the degrees Fahrenheit in the upper box and then clicks the "Compute" button, the temperature in Celsius degrees is displayed in the lower box. Note that a form is used because any time you wish to gather information from the user you must use a form.

Enter Fahrenheit Temperature:

- ?? The formula to convert Fahrenheit to Celsius is: $C = (5 + F - 160) / 9$
- ?? **Celsius** is the name we have given our function.
- ?? **MyForm** is the name we have given our form.
- ?? **F** is the variable name we have given to the number of degrees Fahrenheit the user types in.
- ?? **Answer** is the name we have given to the field where the result (degrees Celsius) is displayed.

```

<HTML>
<HEAD>
<TITLE>Fahrenheit to Celsius</TITLE>
<SCRIPT LANGUAGE=JAVASCRIPT TYPE="TEXT/JAVASCRIPT">

function Celsius()
    {
        document.MyForm.answer.value = (5 * document.MyForm.F.value - 160) / 9
    }
</SCRIPT>
</HEAD>
<BODY>

<FORM NAME="MyForm">

Enter Fahrenheit Temperature:
<INPUT TYPE="text" NAME="F"><BR>
<INPUT TYPE="button" VALUE="Compute" onClick="Celsius()><BR>
<INPUT TYPE="text" NAME="answer">

</FORM>
</BODY>
</HTML>

```

Analyzing the Function

All custom functions must begin with the word "function".

"Celsius" is the name we have given our function.

The function statements must be within the French braces { }

```
function Celsius()
{
document.MyForm.answer.value = (5 * document.MyForm.F.value - 160) / 9
}
```

This determines the value that should be displayed in the "answer" field which is located within the form called "MyForm", all of which is within the document window.

This section takes the user input (the field named "**F**") and plugs into the equation which converts Fahrenheit to Celsius. "document.MyForm" indicates the location of "**F**" and "value" defines what is being typed into "**F**".

Analyzing the Form

Here we have using the HTML attribute NAME to name our form "MyForm".

This tag creates the input box and names it "F".

```
<FORM NAME="MyForm">
Enter Fahrenheit Temperature:
<INPUT TYPE="text" NAME="F"><BR>
<INPUT TYPE="button" VALUE="Compute" onClick="Celsius()"><BR>
<INPUT TYPE="text" NAME="answer">
```

This tag creates the field where the answer is displayed and has been named "answer".

This line creates the "Compute" button and activates our function. The JavaScript code "onClick" is used to call the function "Celsius()".

In English, the code basically says: "When the user clicks the button, take the value the user entered in the field called "F" and place it in the equation where "F" placeholder is. This will tell you what should be displayed in the field called "answer."

To test the script, simply save the file and load it into the browser window. Type a number in the upper field (F) then click "Computer". For example, if you type **212** in the input box, you should get **100** in the output box.

Variation on Example 7: Using Properties (Variables)

As with the previous function, there are several different ways to define the Celsius() function as well. The code below uses variables to define the function.

FAH is set equal to the value the user types in field "F".

ANS uses the FAH variable to determine the degrees celsius and the answer is equal to ANS

```
function Celsius()
{
  Fah=document.MyForm.F.value
  ANS=(5*Fah-160)/9
  document.MyForm.answer.value=ANS
}
```

Variation on Example 7: Using the "onChange" to Look for TAB or ENTER

In the script above, the user must press the "Compute" button to perform the calculation; however, you can also script the code to activate the function if the user presses the TAB or ENTER key after entering the degrees Fahrenheit.

To do this, simply place the "onChange" event handler within the tag which creates the input field as shown in the code below. The "onChange" event handler activates when the field loses focus after being changed. (i.e. you type something there then press ENTER or TAB.) Now the user can perform the calculation by either clicking the button or tabbing out of the input field.

```
<INPUT TYPE="text" NAME="F" onChange="Celsius()><BR
```