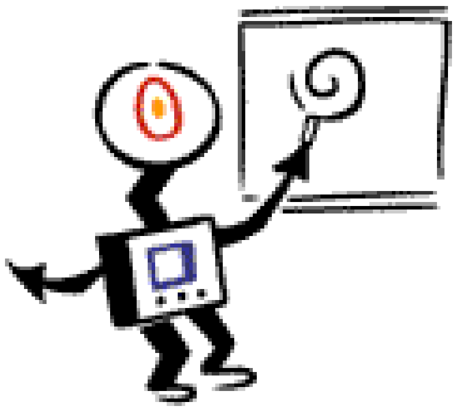
Introduction to JavaScript



Student Training Program
Technology Training Services
Cornell Information Technologies (CIT)
126 Computing and Communication Center (CCC)
255-8000
it-training@cornell.edu

Last updated: 7/23/01

Introduction to JavaScript

Equipment / software

We will use Notepad on Windows 98 PCs, but any test editor capable of saving documents in plain-text format will work. Any web browser can be used; we will use Netscape Navigator. Be aware, however, that there are some browsers that would handle your code differently.

Handouts & class files

- Class handout: Introduction to JavaScript
- base.html
- cat.gif
- demo.html
- dog.gif
- family.gif
- hamster.gif
- harry.gif
- logoscroll.gif
- logostill.gif
- resources.html

Prerequisites

We will assume you have a working knowledge of HTML tags.

Introduction

In this class, you'll become familiar with some concepts and terminology of JavaScript by creating some working examples. We hope you'll learn enough that you are able to use the materials referenced at the end of this handout on your own.

This course will NOT be dealing with programming fundamentals, or rigorous programming structures within JavaScript. That topic is far too broad for a two-hour introduction to JavaScript. For more depth, please try at the online tutorials listed in the references section at the end of this document or take Cornell's course, CS 130 **Creating Web Documents.**

We will use JavaScript to add the following features to your web page:

- Dynamically created HTML
- Pop-up boxes
- Mouse Rollover effects
- Menu
- Changing text in the status bar
- Various effects cutting and pasting from existing scripts
- Drop-down menu with links

Feel free to explore the **demo.html** and take a look at the source code for the scripts we will be creating today. Go to **View > Page Source** to view the code.

Introducing JavaScript

It's important to understand the difference between Java and JavaScript. Java is a full programming language developed by **Sun Microsystems** with formal structures, etc. JavaScript is a *scripting* language developed by **Netscape** that is used to modify web pages. Most JavaScript must be written in the HTML document between **<SCRIPT>** tags. You open with a <**SCRIPT>** tag, write your JavaScript, and write a closing **</SCRIPT>** tag. Sometimes, as an attribute to script, you may add "**Language=JavaScript**" because there are other scripting languages as well as JavaScript that can be used in HTML. We'll go through some examples to demonstrate the syntax of JavaScript.

To understand the workings of JavaScript, it is essential to understand a few basic programming concepts. JavaScript is object-oriented. An *Object* in JavaScript is a resource that has specific characteristics known as *properties* and provides several services known as *methods* and *events*. An example of an object is **document,** which represents the current web page and has properties such as **location** (which stores the URL location of the document) and methods such as **writeln**, which writes dynamically created html text to the document.

A *variable* stores a value. It can be thought of as a labeled box, with the name of the variable as the label and the value as the contents. The JavaScript statement:

var **x**= "hello";

assigns to the variable named **x** the String value "hello".

var **x**=1;

This line of JavaScript assigns to the variable **x** the integer value 1. As you can see, a JavaScript variable can refer to a value of any type; this can be integer, string, or even any type of object. You dont have to specify the type of variable before creating it. Note that object *properties* can be thought of as variables that belong to the object.

A *method* is basically a collection of statements that does something. For example, a *method* "**writeln()**" exists in the **document** object that can be used to write html to your document. *Methods* are predefined in JavaScript. It is possible for you to define functions, which can be thought of as methods you define outside of any object.

When you have the syntax *object.method* as you do with **document.writeln**, the method operates on the object given. In this case, the **writeln** method operates (the operation is writing) to the **document** (the browser window that you see). This syntactic structure is often used in JavaScript.

Javascript and Comments

Some older browsers do not recognize JavaScript. These browsers would sometimes display JavaScript code in the page as if it were part of the contents of the page. Therefore, it is conventional to place JavaScript code between comment tags as follows:

```
<script>

<!--

..JavaScript code goes here..

//-->

</script>
```

Older browsers would just ignore the Javascript code between the <!-- and --> comment tags, while new browsers would recognize it as JavaScript code. The // just before the end comment tag --> is a JavaScript comment symbol, and tells the browser not to execute the end comment tag --> as JavaScript. Using comment tags makes a webpage more accessible to older browsers.

Writing HTML to the document

```
<script language="JavaScript">

<!--

document.writeln("Hello, welcome to my page." );

//-->

</script>
```

Insert the above into your html document between the **<body>** and **</body>** tags. The piece of text (known as a string) between the parentheses is the **parameter** given to the **writeln()** method. The browser will display the HTML between the parentheses when it runs the script.

Note that **writeln()** writes actual HTML to the page, complete with tags, attributes and text. The **parameter** of the **writeln()** method can include tags to format the text. Change the above code to the following:

```
<script language= "JavaScript">

<!--

document.writeln("<p><h1><center>Hello, welcome to my page.</center></h1>" );

//-->

</script>
```

Note how the welcome message displayed by the browser is now a heading, and is centered.

The above example may seem kind of pointless, because we could have just inserted the HTML between the parentheses directly without using a script. This, however, makes it possible to create HTML that would vary depending on certain conditions. Insert the following code into your page:

```
<script language= “JavaScript”>

<!--

document.writeln(“<p><br> This page is located  at ”  +document.location+ “<br>This page
was last modified on ” +document.lastModified);

//-->

</script>
```

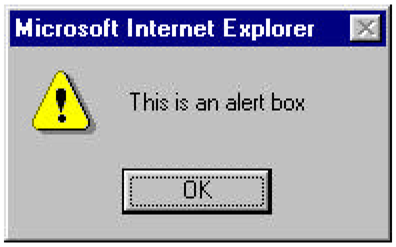
The HTML processed by the browser in this case would vary depending on the values of **document.lastModified** and **document.location**.

Because we wanted this text to be printed in the same position in our page body every time the page is loaded, we placed the JavaScript code between the **<body>** and **</body>** tags wherever we want the text to occur. Sometimes, we place JavaScript code in the head portion of the page if we want it to run before any of HTML in the body is displayed. We can also define JavaScript *functions* in the head of a page, which we will talk about later.

JavaScript has several predefined methods that allow you to create several types of pop-up boxes. Let's look first at "alert" boxes.

Alert boxes

An alert box is a small window that pops up with a yellow warning sign and a (usually) important message. With JavaScript we can either show a message to the user either before a page loads or in response to a user action. Here we will make an alert box that appears before the page loads. It will look like this:



Exercise

1. In Notepad, open **base.html**, your starter HTML page template.
2. Now, within the **<head>** tags, insert the following code:

```
<script language= “JavaScript”>

<!--

      alert("This is a class on JavaScript");

//-->

</script>
```

This makes a pop-up box which says "This is a class on JavaScript" with an **OK** button that allows the user to close it and continue. The message can be changed to whatever you like. This type of pop-up box is called an alert box because it can only be used to alert a viewer. It cannot be used to control where the user may go. **alert()** is a method that takes care of displaying the box. The line “This is a class on JavaScript” is the **parameter** for the **alert()** method. Some more examples of methods are:

confirm(“put message here”) ; This is the next type of box we’ll be looking at.

prompt(“put request here”) ; This asks the user to enter some text.

You probably noticed at the end of a method, after the right parenthesis, there is a semi-colon. In JavaScript, a semi-colon is used to end a "statement".

Now we will make another type of pop-up box.

Confirm Boxes

Place the following code in the body of your HTML document. (Remove the alert box if you like.):

```
<form>

        <input type = "button"  value = "Click here to check the weather"  onClick = "confirm(‘Its
sunny today ' ) ; " >

</form>
```

We have created an input button inside a form. Input elements such as buttons must always be placed inside a **form**, between **<form>** and **</form>** tags.

Take a look in a browser window at the results of this code. The **onClick** event handler means that when you click on the button, a **confirm** box pops up. An event handler allows you to specify what code to execute when an event takes place. You can change the message on the button or in the confirm box by changing the appropriate text in the code. Be sure you’re also using all semicolons, quotation marks, and other punctuation correctly.

Syntax is very important in JavaScript, unlike in HTML, where typos are often overlooked by browsers.

Now we’ll take a look at one more type of pop-up box, the message input box.

Message input boxes

1. Put the following code in the head of your HTML document.

```
<script language= “JavaScript”>
<!—

          var yourname = prompt('Type your name here');
          document.writeln("<center>Welcome, "+ yourname+"</center>");
```

```
//-->

 </script>
```

2. Reload the page in your browser window.
3. Type your name.

The browser will type your name in the window, along with the welcome message. These input strings can be used to personalize your web page. Let's take a closer look at the first line of code.

```
var yourname = prompt('Type your name here');
```

This is an example of a **variable** declaration. As we mentioned, a variable in JavaScript can take on any form, such as integer, character, or string. Just like a person has a name, in order to refer to a variable, you must give it a name. You make a variable with the word **var** as shown above, and the name is the word immediately following **var**. So in this case the name of the variable is **yourname**. The equal sign assigns the variable the value to its right (i.e. 4, 8+5, "c", "chocolate devil"). The prompt method will get a value for **yourname** as you saw when you ran the program. We say that the prompt method *returns* a value, which is stored in the variable **yourname**. Some methods return values, some don't.

The line:

```
document.writeln("<center>Welcome, "+ yourname +"</center>");
```

looks a lot more complicated than it is. It uses whatever value you have for **yourname**, which you get from the text prompt window. As you saw, it types on the screen "Welcome," and then your text. The <center>, </center> tags are familiar looking HTML tags, and indeed, all they do is center your message.

The MouseOver

A mouseover can be used to make an image change when the user rolls their mouse over it. What actually happens is that the browser displays a different image when the mouse is over the text or image. Look at **demo.html** again to see how a rollover works. Here are two commercial websites with good examples of mouseovers:

http://www.saturn.com

http://www.audiusa.com. (Choose a model, then select build your Audi)

To specify a mouse rollover you insert the onMouseOver and//or the onMouseOut attributes into the appropriate tag. These should be followed by the JavaScript code to be executed between quotes.

Making a simple MouseOver

Now that we've looked at what a mouseover can do, let's make one. Included in the class files folder are the two pictures used to make the mouseover, **harry.gif** and **family.gif**. This mouseover will display a picture of Harry the pepper(the CIT mascot) until the mouse is moved over it; then it will display his family photo. The pictures will be part of a link to the CIT webpage.

1. Place the following code in the body of your HTML document:

```
<a href="http://www.cit.cornell.edu"

  onMouseOver="document.logo.src='family.gif ' ; "

  onMouseOut ="document.logo.src='harry.gif ' ; " >

<img name="logo" src="harry.gif " border=0></a>
```

2. View your page with the browser. If the mouseover doesn't work, be sure to check all semicolons, apostrophes, and quotation marks.

Here's how the code works. As you can see, all the code that creates the mouseover effect is contained in an anchor tag and is therefore a hyperlink. The **name** attribute of the image tag assigns a name to the image object. The result is that an object is created representing the image, this object is referred to by the variable named "**logo**" which is located inside the **document** object (it is a *property* of the document object). The **onMouseOver** and **onMouseOut** attributes are "Event Handlers" that tell the browser what code to execute when the events occur. You can make it so that the mouseover affects another image on the page by changing the name of the image accordingly in the event handler code.

Animated MouseOver

We saw that a MouseOver can be used to change the image displayed when the mouse rolls over an image. This has an interesting application in animated images. Let's say we have two versions of a GIF image, one of which is a still and one of which is animated. We can use a MouseOver to animate a still image when the mouse rolls over it. In your class files folder you have two images, **logoscroll.gif** and **logostill.gif** which can be used to create this effect.

1. Place the following code in the body of your HTML document:

```
<center>

<a href="http://www.cit.cornell.edu/training/"

onMouseOver="document.ttslogo.src='logoscroll.gif' ;"

onMouseOut="document.ttslogo.src='logostill.gif' ;">

<img src="logostill.gif" name="ttslogo" border="0">

</a>

</center>
```

2. View your page with the browser. If the mouseover doesn't work, be sure to check all semicolons, apostrophes, and quotation marks.

We have created a link to the TTS website using the **logostill.gif image.** We added mouse event handlers to the anchor tag; These change the image source file to **logoscroll.gif** when the mouse rolls over the image, and change it back to **logostill.gif** when the mouse leaves the image.

Menu

We will now look into creating a menu using Javascript. We will create a simple menu, which has a number of clickable options. There will be a single image which changes when options is selected. We will also create a **function** which takes care of changing the menu image.

You will be using the images **dog.gif**, **cat.gif** and **hamster.gif** provided in the class files folder.

1. Place the following code in the body of your HTML document:

```
<center>

<h1>My Favourite Pets</h1>

<img src="dog.gif" name="pet" height=18% width=8%>

<br>

<font size=5>

<p><a href="javascript:changePet('dog.gif');">.Dog.</a>

<p><a href="javascript:changePet('cat.gif');">.Cat.</a>

<p><a href="javascript:changePet('hamster.gif');">.Hamster.</a>

</font>

</center>
```

Notice how we used anchor tags to enclose the menu entries. We made the entries into links by defining the **href** attribute. This, however, is not your regular sort of link. Instead of defining the **href** attribute to be a URL, we defined JavaScript code to be executed when the link is clicked. When you click any of these “links”, the corresponding JavaScript is executed. We used the **javascript:** prefix to denote that the target of the link is the following JavaScript code, and not a regular URL.

The JavaScript executed when the cat button is clicked, for example, is **changePet(‘cat.gif’);** . Where did the **changePet** function come from??? Well, nowhere really; It doesnt exist, we will create it. So now we want to create a function that takes in a single parameter, which is the name of a file. This function should then set the source (which is the **src** property) of the image object **pet** to equal that filename.

2. Insert the code for the function **changePet** into the **head** portion of your page:

```
<script language= “JavaScript”>

<!--

function changePet(file)

    {

    document.pet.src=file;

   }

//-->

</script>
```

Functions like the above are usually defined in the head of an HTML document. As shown above, function definitions must me enclosed by opening and closing **script** tags. The syntax for function definition is, as shown above, the keyword **function** followed by the name of the function and a list of parameters between parentheses(if there is more than one parameter the list is seperated by commas). Then, the statements composing the function are placed between curly brackets { } . The above function takes in one parameter, and calls it **file**. There is a single statement in this function; This sets the value of **document.pet.src** to equal the value of **file**. Now that we have defined the **changePet** function, calls to that function from the links in our menu can be made.

3. View your page with the browser. If the menu doesn’t work, be sure to check all semicolons, apostrophes, and quotation marks.

When a menu option is clicked, the corresponding JavaScript code is executed. That code calls the **changePet** function and gives that function a single parameter representing the name of the file. The **changePet** function then receives that parameter, calls it **file**, and uses it to change the source of the **pet** image.

Change the status bar text

JavaScript allows us to change the text in the status bar of the window through the **window.status** property. We will now show you how to place a welcome message on the status bar as soon as your page finishes loading. The event handler goes in the opening **<body>** tag

1. Put this code in the body of your HTML document:

```
<body onload=”window.status= ‘Welcome to my Page!!!’ ”>
```

Notice that the text in the bottom of the browser window now displays " Welcome to my Page!!!" whenever the mouse pointer is not on a link. The **onload** event handler is an attribute of the body tag, and it tells the browser what to do after the body of the page is done loading.

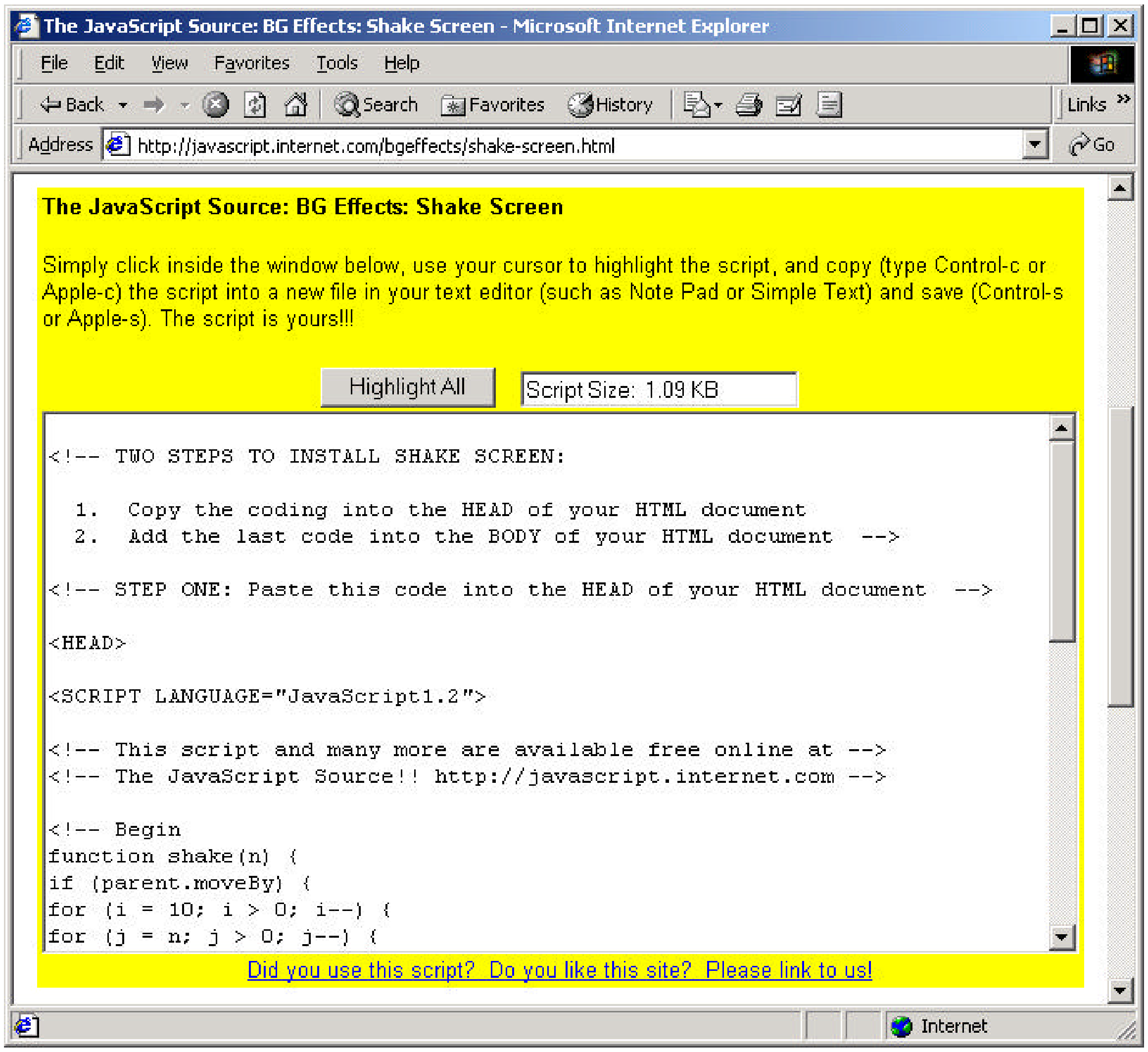
You could also use this effect in conjunction with a mouse event handler to change the status bar text when the mouse rolls over certain links or images, feel free to explore effects like that on your own.

Cutting and Pasting Example Scripts into your Page

Frequently people use existing JavaScript that can be pasted into their code for certain effects, features, etc. There are large archives of free scripts available online for your use. The ones we will implement today are in the **demo.html** file. For any of these effects, you simply cut-and-paste the appropriate code that is given into the appropriate place in your web page. Usually the instructions will tell you where in the document to place it.

Example: Browser Earthquake

1. Open **demo.html**.
2. Click on the link **Click here for the Earthquake source** under the **Shake Screen** button. The embedded text window contains the code you will paste into your page.


The JavaScript Source: BG Effects: Shake Screen - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History Links
Address http://javascript.internet.com/bgeffects/shake-screen.html
Go
Highlight All
Script Size: 1.09 KB
Did you use this script? Do you like this site? Please link to us!
Internet

The JavaScript Source: BG Effects: Shake Screen

Simply click inside the window below, use your cursor to highlight the script, and copy (type Control-c or Apple-c) the script into a new file in your text editor (such as Note Pad or Simple Text) and save (Control-s or Apple-s). The script is yours!!!

```
<!-- TWO STEPS TO INSTALL SHAKE SCREEN:

  1.  Copy the coding into the HEAD of your HTML document
  2.  Add the last code into the BODY of your HTML document  -->

<!-- STEP ONE: Paste this code into the HEAD of your HTML document  -->

<HEAD>

<SCRIPT LANGUAGE="JavaScript1.2">

<!-- This script and many more are available free online at -->
<!-- The JavaScript Source!! http://javascript.internet.com -->

<!-- Begin
function shake(n) {
if (parent.moveBy) {
for (i = 10; i > 0; i--) {
for (j = n; j > 0; j--) {
```

3. Follow the instructions on the screen to paste the code in the appropriate places in your HTML document. There is one part that goes in the head, and another smaller part that goes in the body, which makes the button you press cause the browser earthquake.
4. View the page.

Example: Pull-Down Menus

1. In **demo.html** click on the **Click here for the Menu source** link under the pull down menu near the bottom of the page. As before, insert the appropriate code into your document.
2. View the page. You will see a fully functioning pull-down menu.

The code we pasted into the body section should look like this:

```
<center><form name="form">

<select name="site" size=1 onChange="javascript:formHandler()">

<option value="  ">Go to....

<option value="http://www.yahoo.com">Yahoo

<option value="http://www.metacrawler.com">Metacrawler

<option value="http://www.lycos.com">Lycos

<option value="http://javascript.internet.com">JavaScript Source

</select></form></center>
```

We can change where the links in the pull-down menu take us by changing the **value** attribute in the **option** tag. We can also change link text by changing the text after each **option** tag.

3. Practice changing the URLs and link text.
4. Practice adding a few options to the pull-down menu.
5. View your page.

Design Considerations

Some older browsers are incompatible with JavaScript. JavaScript can also be turned off on many browsers. You can turn it off on Netscape by going to **Edit > Preferences > Advanced** and unchecking the **Enable JavaScript** checkbox. It is therefore a good idea to make sure that someone viewing your page without a JavaScript enabled browser can navigate your page properly. For examp le, creating links as alternatives to using the pull down menu above would provide such accessibility.

Review

Today we have laid the groundwork for you to learn more about JavaScript. Through the examples, you have been exposed to what methods, functions, objects, event handlers, parameters and statements are in the context of JavaScript. You have written some code on your own, as well as implemented existing code. We hope you'll now be comfortable continuing your study of JavaScript through tutorials, books, and perhaps a full-semester course.

Introduction to JavaScript: A short glossary

statement

A statement in JavaScript is comparable to a sentence in English – both contain complete ideas that the reader can understand independent of the rest of the document. A sentence is terminated with a period, but a statement in JS always ends with a semicolon.

Example**: x = 5+2;**

variable

A storage container for values. Once a value has been assigned to a variable, that variable's identifier can be used in place of it's value anywhere in a program. A variable's value can be changed at any point in the program (hence the name *variable*). Below **myname** is the **identifier** and **"Paul"** is the **value**.

Example: **var myname = "Paul";**

object

A part of the web page or browser that JavaScript can control. Primary objects in JavaScript include:

window	The browser window – has control over things like the status bar and scroll bars.
document	The contents of the web page – has control over things like text color and size.

An object can contain other objects as properties.

property

All objects have a unique set of properties accessible to the programmer. Changing one or more of these properties will change the way an object looks or behaves. In the following example, **window** is the **object** and **status** is the **property**.

Example: **window.status = "Hi there";**

Other examples of objects and properties:

Object Property

document.bgColor	Changing this changes the background color of the web page
window.location	Changing this loads a different page into the browser.

method

All objects also have a unique set of methods available to the programmer. Each method will invoke some sort of action on the object. Methods are like functions (see the function handout) in that you pass them parameters and they perform some action with those parameters, but they are not user-defined like functions are. Rather, methods are built into the JavaScript language.

Example: **window.alert("This is an alert message");**

Other examples of objects and methods:

Object Method	
window.home()	This makes the browser load its user-defined homepage.
document.write(“bla”)	This writes the text “bla” onto the webpage.

event handler

Many objects also have event handlers, which catch user triggered events (such as mouse clicks and keystrokes) and allow the programmer to specify an action for each of these events.

Example: onClick= “alert(‘you clicked a button’);”

A few event handlers:

onLoad	Performs specified actions when page is loaded
onClick	Performs specified actions when mouse is clicked on the object that contains the handler statement.
onKeyPress	Does something when a key is pressed on the keyboard.

JavaScript Resources

Language References

DevGuru JavaScript Introduction
http://www.devguru.com/Technologies/ecmascript/quickref/javascript_intro.html

JavaScript Tutorials

WebMonkey's JavaScript Resource Page
http://www.hotwired.com/webmonkey/programming/javascript/

Website Abstraction's JavaScript Tutorials
http://wsabstract.com/javaindex.shtml

WDVL: Authoring JavaScript
http://wdvl.internet.com/Authoring/JavaScript/

Script Archives

JavaScript Source
http://javascript.internet.com/

DevHead's JavaScript Library
http://www.zdnet.com/devhead/resources/scriptlibrary/javascript/

Cornell Class

Computer Science 130
http://www3.cs.cornell.edu/cs130/