

Why Script?

When the World Wide Web first became popular, HTML was the only language used to create Web pages. HTML is not a programming language but a markup language and still quite limited in what it can do. HTML positions text and graphics on a Web page but offers users relatively limited interactivity with the page itself.

Most computer users are now accustomed to graphical application interfaces, whether they use Windows, Macintosh, UNIX or some combination thereof. Users click buttons to execute command sequences, enter values into text boxes and choose from menu lists. This increase in user abilities and expectations has resulted in the continuous improvement of HTML and the advent of powerful scripting languages such as JavaScript.

Even with its evolution and improvements, X/HTML has little or no ability to interact with your site users. This interaction can be achieved through the use of server-side technologies such as Common Gateway Interface (CGI) or programming languages such as Java.

JavaScript is a client-side scripting language. This means that it runs on the user's computer after the page is downloaded. Therefore, some of the script-processing burden can be passed from the server to the client machine. Allowing the client to do the work frees the server to perform other, more important functions and services.

JavaScript allows you to create Web pages that interact with your users, based on their keyboard and mouse inputs. For example, if a user types his or her name into a form input field, you can use JavaScript to check the field for errors. Alternatively, you can write a script for a mouse event that displays Help files at locations on your pages where users need them.

JavaScript and Common Programming Concepts

Before learning about JavaScript in particular, you should understand some key programming concepts. Some of these concepts are simple and others are more advanced. Do not be concerned if some concepts do not make sense right away. You will acquire a better understanding of these concepts as you apply them.

Scripting languages are subsets of larger languages. Unlike full programming languages, scripting languages are typically interpreted at run-time. This means that, instead of the entire program's code being compiled into a single executable file, individual instructions in the scripts are compiled and run as needed. The benefit to the Web developer is that scripting code can be inserted into an X/HTML page and modified as easily as modifying X/HTML markup.

Scripting languages may provide less functionality than full programming languages such as C, C++, Java or Visual Basic but they are generally easier to learn. Your investment in learning a scripting language is valuable if you are interested in programming languages because you will gain basic conceptual awareness of programming practices. Many concepts are handled similarly among these types of languages.

object

A programming function that models the characteristics of abstract or real objects using classes.

property

A characteristic, such as colour, width or height, that the programmer stipulates in the creation of the object.

method

An action performed by an object.

Objects, properties and methods

You were introduced to some basic programming terms in an earlier lesson. In programming, **objects** encapsulate predetermined attributes and behaviors. They are often grouped with similar objects into classes.

In object-oriented and object-based languages, every object has attributes that characterise it and behaviors that it can perform. Attributes and behaviors are known by two other terms in programming: **properties** and **methods**. Properties represent various attributes of the object, such as height, colour, font size, age and so forth. Methods are the actions an object can be made to perform, such as a calculation, onscreen movement or the writing of text.

Think of a pen as an object. It has definite, discernible properties, such as length, ink colour, point style and so forth. Two pens may be similar yet have different values for these same properties. All pens have a colour, for example, but not all pens have the same colour.

A pen has methods, as well. It can write, spin, leak, flip, have its cap removed or replaced and so forth. A developer creating a virtual pen object would want to simulate natural attributes and behaviors so that, for example, if the user were trying to write with the pen, he might receive an error telling him he must first remove the cap from the pen.

Now consider a sentence, which is a string of words. To a programmer, that sentence might be an object, which has minimum properties of colour, font style and character length. In addition, it might have a method by which that sentence could be converted to all uppercase or all lowercase.

You already use and change objects, properties and methods, although you may not have thought of them in these terms. Each time you load a Web page, you access the window and document objects. When you specify a background colour for your page, you specify a property for that document object. When you click the Back and Forward buttons on your browser to change pages, you access the history method associated with the window object, allowing you to navigate among previously loaded pages.

In this lesson, you will learn how to work with some of the basic objects, properties and methods available for scripting purposes.

What Is JavaScript?

Before you start writing in JavaScript, you will take a close look at the features of this language.

JavaScript is a scripting language

A scripting language is a simple programming language designed to enable computer users to write useful programs easily. JavaScript was designed to allow designers to manipulate Web page elements simply and easily.

If you have ever written a macro in Microsoft Excel or used WordBasic to perform some task in a Microsoft Word document, you have already used a

scripting language. Smaller and less powerful than full programming languages, scripting languages provide easy-to-access functionality.

JavaScript is a scripting language. Its syntax is similar to that of C or Pascal. JavaScript is part of the code within your X/HTML document. When your browser retrieves a scripted page, it executes the JavaScript programs and performs the appropriate operations.

JavaScript is object-based, not object-oriented

Object-oriented is a term commonly used in relation to programming languages. An object-oriented program is handled as a collection of individual objects that each perform a different function, and not as a sequence of statements that performs a specific task. These objects are usually related in a hierarchical fashion, in which new objects and subclasses of objects inherit the properties and methods of the objects above them in the hierarchy. JavaScript is not considered object-oriented because it does not allow for object inheritance and subclassing. However, JavaScript is an *object-based* language in that it depends for functionality on a collection of built-in objects. With JavaScript, you can also create your own objects.

JavaScript is event-driven

The World Wide Web is based upon an event-driven model. For example, whenever you click an element on a Web page, an event occurs. The other common programming model is the procedural model. In the procedural model, the user is expected to interact with the program in a fairly sequential manner. On a Web page, by contrast, the user is in control and can click or not click, move the mouse or not move the mouse or change the URL at will. Because of the unpredictability of a user's actions, you can create programming modules (called subroutines or functions) that are independent of each other and do not require a sequential set of operations.

Events trigger functions. Event triggers can be as simple as the user clicking a button, moving the mouse over a hyperlink or entering text into a field. Scripting can be associated with any of these events. You will learn to use JavaScript to instruct the browser what to do upon the event that you designate as the trigger for the script.

Strengths of JavaScript

JavaScript provides several benefits to the Web developer, including a short development cycle, a mild learning curve and platform independence. These benefits mean that JavaScript can be used to extend X/HTML pages on the Web easily and quickly. JavaScript is well suited for small, simple programs. It also handles repetitive tasks well. For larger programs, Java is the better choice.

Quick development

Because JavaScript does not need time-consuming compilation, scripts can be developed in a relatively short period. This advantage is enhanced by the fact that most of the interface features, such as forms, frames and other graphical user interface (GUI) elements, are handled by the browser and X/HTML code.

JavaScript programmers need not worry about creating or handling these elements in their applications.

Easy to learn

JavaScript does not include the complex syntax and rules associated with Java. Even if you know no other programming language, you can learn JavaScript without much difficulty. JavaScript also makes troubleshooting easy. You can write, test and change the program without having to recompile the script.

Platform independence

By its nature, the World Wide Web is platform-independent. Because JavaScript programs are designed to run inside X/HTML documents, they are not limited to any specific hardware platform or operating system. The same program can be run on any platform using Netscape Navigator 2.0 or later or Microsoft Internet Explorer 3.0 or later.



Movie Time!

Insert the CIW v5 Master Designer Movie CD to learn even more about this topic.

JavaScript (approx. playing time: 04:30)

All movie clips are © 2009 LearnKey, Inc.

OBJECTIVE:
4.2.1: Client-side vs. server-side technologies

JavaScript v. Other Languages

Although JavaScript can be used outside a Web browser, it is most often used within X/HTML documents to add interactivity to Web pages without using server-based applications, such as CGI programs.

JavaScript v. Java

Although the names are similar, Java and JavaScript are different languages. Java is a full-fledged object-oriented programming language developed by Sun Microsystems, Inc. Java can be used to create stand-alone applications and a special type of mini-applications called Java applets.

Applets are written in Java, compiled and then referenced via the <object> tag, according to the current X/HTML standards. (Recall that the HTML 3.2 standard used the <applet> tag, which was deprecated in HTML 4.0 but is still widely used for compatibility purposes.) Applets can provide a great variety of added functionality to Web sites. You will learn about applets in more detail later in this course.

Although it uses some of Java's expression syntax and basic program flow controls, JavaScript is separate and does not require Java.

JavaScript v. VBScript

VBScript, a scripting language developed by Microsoft, is a subset of the Visual Basic programming language. VBScript is considerably easier to learn than Visual Basic because VBScript is an interpreted language, not a compiled language.

Both JavaScript and VBScript extend browser capabilities. JavaScript was the first scripting language developed for Web page design and VBScript is the Microsoft response. The VBScript visual approach might be easier to understand initially but the language is only supported by the Internet Explorer browser. JavaScript works relatively smoothly with nearly every Web browser and is therefore a better choice for client-side scripting.

JavaScript v. JScript

The Microsoft implementation of the Netscape JavaScript language is called JScript. Several minor differences exist between these two implementations of essentially the same language. However, these differences can cause problems. If you decide to learn JScript, be sure to test your code's execution in browsers other than Internet Explorer.

JavaScript, JScript and ECMAScript

The many scripting language choices have created a bit of a battle among developers. In an effort to standardise, Netscape and Microsoft are moving toward the European Computer Manufacturers Association (ECMA) version of scripting language called ECMAScript.

ECMAScript development began in 1996; the first edition was adopted by the ECMA General Assembly of June 1997. Now, all major browsers comply with the ECMAScript standard. ECMAScript and JavaScript are essentially the same and the two terms are often used interchangeably.



Technically, ECMAScript is a standardised language that contains the common functionality between JScript and JavaScript. Microsoft's JScript and Netscape's JavaScript expand on this core functionality for use in their respective browsers. However, all three languages are usually referred to as 'JavaScript.'

Embedding JavaScript into X/HTML

JavaScript resides within X/HTML documents. It is usually placed within the <head> section of an X/HTML document using the <script> </script> tags. You can add script to the <head> or <body> section (or both) of an X/HTML document. You can also embed scripting instructions directly into certain X/HTML tags; this technique is called inline scripting.

The basic structure of an X/HTML file with JavaScript is as follows:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title> Page Title Here </title>

<script language="JavaScript" type="text/javascript">
<!--
JavaScript code goes here
// -->
</script>

</head>
<body>

Page content here

</body>
</html >
```

The *language* attribute in the `<script>` tag tells the browser which type of script is supplied. If you do not add the *language* attribute, the default scripting language in both Internet Explorer and Firefox is JavaScript. However, to ensure that the browser correctly interprets even JavaScript code, it is recommended that you always add the *language* attribute. By doing this, you give the browser an explicit instruction rather than implying one. This attribute will also support a designated version of the language. For example, consider the following attribute in the `<script>` tag:

```
<script language="JavaScript1.1">
```

This attribute tells the browser to ignore code that is not part of the JavaScript 1.1 standard.

Notice that the comment tag `<!-- //-->` is used to ‘comment out,’ or hide, text in the script. You can add this tag in case a user visiting your page has an old browser that is incapable of interpreting JavaScript. As more people upgrade their browsers, using the comment tag in this fashion will become less necessary.

Notice too that the closing comment tag line has a special addition. The double slash characters (`//`) at the beginning of this line demonstrate the JavaScript method of commenting out an item. JavaScript does not expect to see the double hyphen characters (`--`) at the beginning of the closing comment tag and considers this a decrement operator; therefore it must be commented out with the double slash. Failure to include these slash or hyphen characters will result in an error message when the page is loaded and will prevent the script from executing properly.

Dot notation

OBJECTIVE:
4.2.4: JavaScript dot notation

As you learned briefly in the previous lesson, dot notation is used to associate an object's name with its properties or methods. The syntax for JavaScript dot notation is as follows:

```
objectName. objectProperty
objectName. objectProperty = value

objectName. objectMethod()
```

```
objectName. objectMethod(argument1, argument2, . . . )
parentObjectName. objectName. objectProperty
parentObjectName. objectName. objectMethod
```

For example, a statement to close a window uses dot notation to associate the `close()` method with the `window` object named `myWindow`, as follows:

```
onClick="myWindow.close()"
```

Dot notation is also used to show the hierarchical relationship between objects. For example, you can use the `document.write()` method to write text to the screen. Strictly speaking, the method could also be written as `window.document.write()`. It is not incorrect to write code in this manner but it is ultimately less efficient. Because `window` is the default object, you need not write it out; in this case, the `window` object is the parent object and also the default object.

Using JavaScript to Communicate with the User

OBJECTIVE:
4.2.2: JavaScript objects, properties and methods

In many programs, you will see message windows from time to time alerting you to changing conditions, choices to make, warnings and other notices. These pop-up windows are also frequently used for advertising purposes and are often considered annoying in this capacity.

As a Web page developer, you can quickly and easily script user messages that will pop up when the page loads or upon some other event, such as a mouse click. For now, you will use the primary event — the loading of the page into the browser window — to launch your scripts. You will examine two ways of communicating with the user: a text message that displays in a pop-up window and a request for information that displays in a pop-up window.



The primary event is the first that occurs on any Web page. No user events can take place before the page is loaded in the browser.

Giving the user a message: The `alert()` method

OBJECTIVE:
4.2.13: Pop-up / pop-under windows

The `alert()` method is a simple JavaScript method that allows you to communicate with the user. The `alert()` method is a method of the `window` object, part of the JavaScript language. You will learn about the `window` object in more detail later.

statement
A single line of code to be executed in a script or program.

To call the `alert()` method, you need only have a simple line of code, called a **statement**, in a script block somewhere in your document. If you have no other code in your script, the message will pop up every time the page is loaded. If your script has other code, the message window will pop up when it is called in the code sequence.

The syntax for the `alert()` method is as follows:

```
alert("message");
```

The `alert()` method can be used to create pop-up or pop-under windows for advertising, which you learned about earlier in this course. Carefully consider

the purpose of pop-ups in your site. Many pop-ups (especially advertising) are considered annoying by users and are therefore disabled in their browsers. Always consider your site's purpose and audience (and the usefulness of the pop-up window) before adding pop-up functionality to your Web sites.

In the following lab, you will create your first script using the JavaScript *alert()* method. Suppose you want to greet users of your Web site when the page loads. Rather than adding a greeting to the content of your home page, you want a brief message to pop up in its own dialog box, allowing the user to delete it after receiving it. You can add such a greeting to your Web page using the JavaScript *alert()* method, which creates a pop-up alert box that can display any message. Alert messages can be used to display any text you want: copyright information, notices of Web site changes, current date and time information or a quote of the day. In time, you will discover the usefulness of these messages when you need to debug your own script.

**Lab 30-1: Using the JavaScript *alert()* method**

OBJECTIVE:
4.2.3: Using
JavaScript

In this lab, you will use the JavaScript *alert()* method to script a message to the user. This lab uses a script example with a single statement.

1. Notepad: From the **C:\CIW\Dsgn_Mthd_Tech\LabFiles\Lesson30\Lab_30-1** folder, open the file **alert.htm** in your text editor.

2. Notepad: Add the following source code indicated in bold:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Alert</title>
<script language="JavaScript" type="text/javascript">
<!--
alert("Good Morning!");
// -->
</script>
</head>
<body>
<h1>Welcome to my page!</h1>
<hr />
</body>
</html>
```

3. Notepad: Save the changes to **alert.htm**.

4. Browser: Open the file **alert.htm** in your browser. Compare the screen displayed in your browser with Figure 30-1. They should be similar depending on which browser you are using. If not, verify that the source code you entered is correct.



Figure 30-1: Alert message

- 5. Browser:** Click the **OK** button to acknowledge and dismiss the alert box. After clicking OK, your screen should resemble Figure 30-2.

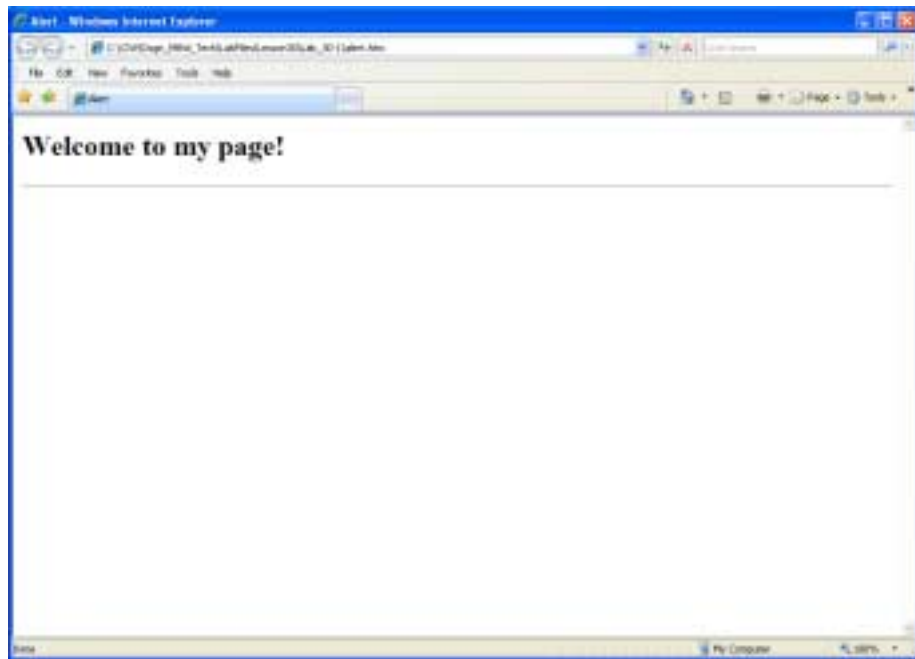


Figure 30-2: Page displayed after acknowledging JavaScript alert