**Introducing JavaScript**

JavaScript was developed by Netscape Communications Corporation, the maker of the Netscape web

browser. JavaScript was the first web scripting language to be supported by browsers, and it is still

by far the most popular.

JavaScript is almost as easy to learn as HTML, and it can be included directly in HTML documents.

Here are a few of the things you can do with JavaScript:

• Display messages to the user as part of a web page, in the browser’s status line, or in alert boxes

**Note**

A bit of history: JavaScript was originally called LiveScript and was first introduced in Netscape

Navigator 2.0 in 1995. It was soon renamed JavaScript to indicate a marketing relationship with Sun’s

Java language—although there is no other relationship, structurally or otherwise, between Java and

JavaScript.

• Validate the contents of a form and make calculations (for example, an order form can

automatically display a running total as you enter item quantities)

• Animate images or create images that change when you move the mouse over them

• Create ad banners that interact with the user, rather than simply displaying a graphic

• Detect the browser in use or its features and perform advanced functions only on browsers that

support them

• Detect installed plug-ins and notify the user if a plug-in is required

• Modify all or part of a web page without requiring the user to reload it

• Display or interact with data retrieved from a remote server

You can do all this and more with JavaScript, including creating entire applications. We’ll explore

the uses of JavaScript throughout this book.

**How JavaScript Fits into a Web Page**

Using the <script> tag, you can add a short script (in this case, just one line) to a web document, as

shown in Listing 4.1. The <script> tag tells the browser to start treating the text as a script, and the

closing </script> tag tells the browser to return to HTML mode. In most cases, you can’t use JavaScript

statements in an HTML document except within <script> tags. The exception is event handlers, described

later in this chapter.

Listing 4.1 A Simple HTML Document with a Simple Script

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<title>The American Eggplant Society</title>

</head>

<body>

<h1>The American Eggplant Society</h1>

<p>Welcome to our site. Unfortunately, it is still

under construction.</p>

<p>We last worked on it on this date:

<script type="text/javascript">

<!-- Hide the script from old browsers

document.write(document.lastModified);

// Stop hiding the script -->

</script>

</p>

</body>

</html>

JavaScript’s document.write statement, which you’ll learn more about later, sends output as part of the web

document. In this case, it displays the modification date of the document, as shown in Figure 4.1.

**Figure 4.1** Using document.write to display a last-modified date.

In this example, we placed the script within the body of the HTML document. There are actually four

different places where you might use scripts:

• **In the body of the page**—In this case, the script’s output is displayed as part of the HTML

document when the browser loads the page.

• **In the header of the page between the** <head> **tags**—Scripts in the header can’t create output

within the HTML document, but can be referred to by other scripts. The header is often used for

functions—groups of JavaScript statements that can be used as a single unit. You will learn more

about functions in Chapter 14, “Getting Started with JavaScript Programming.”

• **Within an HTML tag, such as** <body> **or** <form>—This is called an *event handler* and enables the

script to work with HTML elements. When using JavaScript in event handlers, you don’t need to

use the <script> tag. You’ll learn more about event handlers in Chapter 14.

• **In a separate file entirely**—JavaScript supports the use of files with the .js extension containing

scripts; these can be included by specifying a file in the <script> tag.

**Using Separate JavaScript Files**

When you create more complicated scripts, you’ll quickly find your HTML documents become large

and confusing. To avoid this, you can use one or more external JavaScript files. These are files with

the .js extension that contain JavaScript statements.

External scripts are supported by all modern browsers. To use an external script, you specify its

filename in the <script> tag:

<script type="text/javascript" src="filename.js"></script>

Because you’ll be placing the JavaScript statements in a separate file, you don’t need anything

between the opening and closing <script> tags—in fact, anything between them will be ignored by the

browser.

**Tip**

External JavaScript files have a distinct advantage: You can link to the same .js file from two or more

HTML documents. Because the browser stores this file in its cache, this can reduce the time it takes for

your web pages to display.

You can create the .js file using a text editor. It should contain one or more JavaScript commands and

only JavaScript—don’t include <script> tags, other HTML tags, or HTML comments. Save the .js file in

the same directory as the HTML documents that refer to it.

**Understanding JavaScript Events**

Many of the useful things you can do with JavaScript involve interacting with the user and that means

responding to *events*—for example, a link or a button being clicked. You can define event handlers

within HTML tags to tell the browser how to respond to an event. For example, Listing 4.2 defines a

button that displays a message when clicked.

Listing 4.2 A Simple Event Handler

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"

"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>

<title>Event Test</title>

</head>

<body>

<h1>Event Test</h1>

<button type="button"

onclick="alert('You clicked the button.')">

Click Me!</button>

</body>

</html>

In various places throughout this book, you’ll learn more about JavaScript’s event model and how to

create simple and complex event handlers.