**Creating Output**

You now have two variables—localtime and utctime—which contain the results we want from our script.

Of course, these variables don’t do us much good unless we can see them. JavaScript includes a

number of ways to display information, and one of the simplest is the document.write statement.

The document.write statement displays a text string, a number, or anything else you throw at it. Because your

JavaScript program will be used within a web page, the output will be displayed as part of the page.

To display the result, add these statements before the final </script> tag:

document.write("<strong>Local time:</strong> " + localtime + "<br/>");

document.write("<strong>UTC time:</strong> " + utctime);

These statements tell the browser to add some text to the web page containing your script. The output

will include some brief strings introducing the results and the contents of the localtime and utctime

variables.

Notice the HTML tags, such as <strong>, within the quotation marks—because JavaScript’s output

appears within a web page, it needs to be formatted using HTML. The <br/> tag in the first line ensures

that the two times will be displayed on separate lines.

**Note**

Notice the plus signs (+) used between the text and variables in Listing 4.3. In this case, it tells the

browser to combine the values into one string of text. If you use the plus sign between two numbers,

they are added together.

**Adding the Script to a Web Page**

You should now have a complete script that calculates a result and displays it. Your listing should

match Listing 4.3.

Listing 4.3 The Complete Date and Time Script

<script type="text/javascript">

now = new Date();

localtime = now.toString();

utctime = now.toGMTString();

document.write("<strong>Local time:</strong> " + localtime + "<br/>");

document.write("<strong>UTC time:</strong> " + utctime);

</script>

To use your script, you’ll need to add it to an HTML document. If you use the general template you’ve

seen in the chapters so far, you should end up with something like Listing 4.4.

Listing 4.4 The Date and Time Script in an HTML Document

<?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>Displaying Times and Dates</title>

</head>

<body>

<h1>Current Date and Time</h1>

<script type="text/javascript">

now = new Date();

localtime = now.toString();

utctime = now.toGMTString();

document.write("<strong>Local time:</strong> "

+ localtime + "<br/>");

document.write("<strong>UTC time:</strong> " + utctime);

</script>

</body>

</html>

**Note**

Notepad and other Windows text editors might try to be helpful and add the .txt extension to your script.

Be sure your saved file has the correct extension.

Now that you have a complete HTML document, save it with the .htm or .html extension.

**Testing the Script**

To test your script, you simply need to load the HTML document you created in a web browser. If

you typed the script correctly, your browser should display the result of the script, as shown in Figure

4.2. (Of course, your result won’t be the same as mine, but it should be the same as the setting of your

computer’s clock.)

**Figure 4.2** Firefox displays the results of the Date and Time script.

A note about Internet Explorer 6.0 and above: Depending on your security settings, the script might

not execute, and a yellow highlighted bar on the top of the browser might display a security warning.

In this case, click the yellow bar and select Allow Blocked Content to allow your script to run. (This

happens because the default security settings allow JavaScript in online documents, but not in local

files.)

**Modifying the Script**

Although the current script does indeed display the current date and time, its display isn’t nearly as

attractive as the clock on your wall or desk. To remedy that, you can use some additional JavaScript

features and a bit of HTML to display a large clock.

To display a large clock, we need the hours, minutes, and seconds in separate variables. Once again,

JavaScript has built-in functions to do most of the work:

hours = now.getHours();

mins = now.getMinutes();

secs = now.getSeconds();

These statements load the hours, mins, and secs variables with the components of the time using

JavaScript’s built-in date functions.

After the hours, minutes, and seconds are in separate variables, you can create document.write statements to

display them:

document.write("<h1>");

document.write(hours + ":" + mins + ":" + secs);

document.write("</h1>");

The first statement displays an HTML <h1> header tag to display the clock in a large typeface. The

second statement displays the hours, mins, and secs variables, separated by colons, and the third adds the

closing </h1> tag.

You can add the preceding statements to the original date and time script to add the large clock

display. Listing 4.5 shows the complete modified version of the script.

Listing 4.5 The Date and Time Script with Large Clock Display

<?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>Displaying Times and Dates</title>

</head>

<body>

<h1>Current Date and Time</h1>

<script type="text/javascript">

now = new Date();

localtime = now.toString();

utctime = now.toGMTString();

document.write("<strong>Local time:</strong> "

+ localtime + "<br/>");

document.write("<strong>UTC time:</strong> " + utctime);

hours = now.getHours();

mins = now.getMinutes();

secs = now.getSeconds();

document.write("<h1>");

document.write(hours + ":" + mins + ":" + secs);

document.write("</h1>");

</script>

</body>

</html>

Now that you have modified the script, save the HTML file and open the modified file in your

browser. If you left the browser running, you can simply use the Reload button to load the new

version of the script. Try it and verify that the same time is displayed in both the upper portion of the

window and the new large clock. Figure 4.3 shows the results.

**Figure 4.3** Firefox displays the modified Date and Time script.

**Dealing with JavaScript Errors**

As you develop more complex JavaScript applications, you’re going to run into errors from time to

time. JavaScript errors are usually caused by mistyped JavaScript statements.

To see an example of a JavaScript error message, modify the statement you added in the previous

section. We’ll use a common error: omitting one of the parentheses. Change the last document.write

statement in Listing 4.5 to read:

document.write("</h1>";

**Note**

The time formatting produced by this script isn’t perfect; hours after noon are in 24-hour time, and there

are no leading zeroes, so 12:04 is displayed as 12:4. See Chapter 17 for solutions to these issues.

Save your HTML document again and load the document into the browser. Depending on the browser

version you’re using, one of two things will happen: Either an error message will be displayed, or the

script will simply fail to execute.

If an error message is displayed, you’re halfway to fixing the problem by adding the missing

parenthesis. If no error was displayed, you should configure your browser to display error messages

so that you can diagnose future problems:

• In Firefox, you can also select Tools, JavaScript Console from the menu. The console is shown in

Figure 4.4, displaying the error message you created in this example.

**Figure 4.4** Firefox’s JavaScript Console displays an error message.

• In Chrome, select Tools, JavaScript Console from the Customizations (Options) menu. A console

will display in the bottom of the browser window.

• In Internet Explorer, select Tools, Internet Options. On the Advanced page, uncheck the Disable

Script Debugging box and check the Display a Notification About Every Script Error box. (If this

is disabled, a yellow icon in the status bar will still notify you of errors.)

The error we get in this case is missing ) after argument list (Firefox) or Expected ')' (Internet Explorer), which

turns out to be exactly the problem. Be warned, however, that error messages aren’t always this

enlightening.

Although Internet Explorer displays error dialog boxes for each error, Firefox’s JavaScript Console

displays a single list of errors and enables you to test commands. For this reason, you might find it

useful to install Firefox for debugging and testing JavaScript, even if Internet Explorer is your

primary browser.