**Exploring JavaScript’s Capabilities**

If you’ve spent any time browsing the Web, you’ve undoubtedly seen lots of examples of JavaScript

in action. Here are some brief descriptions of typical applications for JavaScript, all of which you’ll

explore further, later in this book.

**Improving Navigation**

Some of the most common uses of JavaScript are in navigation systems for websites. You can use

JavaScript to create a navigation tool—for example, a drop-down menu to select the next page to

read or a submenu that pops up when you hover over a navigation link.

When it’s done right, this kind of JavaScript interactivity can make a site easier to use, while

remaining usable for browsers that don’t support JavaScript.

**Validating Forms**

Form validation is another common use of JavaScript. A simple script can read values the user types

into a form and can make sure they’re in the right format, such as with ZIP Codes or phone numbers.

This allows users to notice common errors and fix them without waiting for a response from the web

server. You’ll learn how to work with form data in Chapter 26, “Working with Web-Based Forms.”

**Special Effects**

One of the earliest and most annoying uses of JavaScript was to create attention-getting special

effects—for example, scrolling a message in the browser’s status line or flashing the background

color of a page.

These techniques have fortunately fallen out of style, but thanks to the W3C DOM and the latest

browsers, some more impressive effects are possible with JavaScript—for example, creating objects

that can be dragged and dropped on a page or creating fading transitions between images in a

slideshow.

**Remote Scripting (AJAX)**

For a long time, the biggest limitation of JavaScript was that there was no way for it to communicate

with a web server. For example, you could use it to verify that a phone number had the right number

of digits, but not to look up the user’s location in a database based on the number.

Now that some of JavaScript’s advanced features are supported by most browsers, this is no longer

the case. Your scripts can get data from a server without loading a page or send data back to be

saved. These features are collectively known as AJAX (Asynchronous JavaScript And XML), or

*remote scripting*. You’ll learn how to develop AJAX scripts in Chapter 24, “AJAX: Remote

Scripting.”

You’ve seen AJAX in action if you’ve used Google’s Gmail mail application or recent versions of

Yahoo! Mail or Microsoft Hotmail. All of these use remote scripting to present you with a responsive

user interface that works with a server in the background.

**Displaying Time with JavaScript**

One common and easy use for JavaScript is to display dates and times. Because JavaScript runs on

the browser, the times it displays will be in the user’s current time zone. However, you can also use

JavaScript to calculate “universal” (UTC) time.

**Note**

UTC stands for Universal Time Coordinated, and is the atomic time standard based on the old GMT

(Greenwich Mean Time) standard. This is the time at the Prime Meridian, which runs through

Greenwich, London, England.

As a basic introduction to JavaScript, you will now create a simple script that displays the current

time and the UTC time within a web page, starting with the next section.

**Beginning the Script**

Your script, like most JavaScript programs, begins with the HTML <script> tag. As you learned earlier

in this chapter, you use the <script> and </script> tags to enclose a script within the HTML document.

**Caution**

Remember to include only valid JavaScript statements between the starting and ending <script> tags. If

the browser finds anything but valid JavaScript statements within the <script> tags, it will display a

JavaScript error message.

To begin creating the script, open your favorite text editor and type the beginning and ending <script>

tags as shown.

<script type="text/javascript"></script>

**Adding JavaScript Statements**

Your script now needs to determine the local and UTC times, and then display them to the browser.

Fortunately, all the hard parts, such as converting between date formats, are built in to the JavaScript

interpreter.

**Storing Data in Variables**

To begin the script, you will use a *variable* to store the current date. You will learn more about

variables in Chapter 16, “Using JavaScript Variables, Strings, and Arrays.” A *variable* is a container

that can hold a value—a number, some text, or in this case, a date.

To start writing the script, add the following line after the first <script> tag. Be sure to use the same

combination of capital and lowercase letters in your version because JavaScript commands and

variable names are case sensitive.

now = new Date();

This statement creates a variable called now and stores the current date and time in it. This statement

and the others you will use in this script use JavaScript’s built-in Date object, which enables you to

conveniently handle dates and times. You’ll learn more about working with dates in Chapter 17,

“Using JavaScript Functions and Objects.”

**Note**

Notice the semicolon at the end of the previous statement. This tells the browser that it has reached the

end of a statement. Semicolons are optional, but using them helps you avoid some common errors. We’ll

use them throughout this book for clarity.

**Calculating the Results**

Internally, JavaScript stores dates as the number of milliseconds since January 1, 1970. Fortunately,

JavaScript includes a number of functions to convert dates and times in various ways, so you don’t

have to figure out how to convert milliseconds to day, date, and time.

To continue your script, add the following two statements before the final </script> tag:

localtime = now.toString();

utctime = now.toGMTString();

**Note**

The localtime and utctime variables store a piece of text, such as January 1, 2001 12:00 PM. In programming

parlance, a piece of text is called a *string*.

These statements create two new variables: localtime, containing the current time and date in a nice

readable format, and utctime, containing the UTC equivalent.