

## Chapter 1

# All You Ever Wanted to Know about JavaScript (But Were Afraid to Ask!)

---

### *In This Chapter*

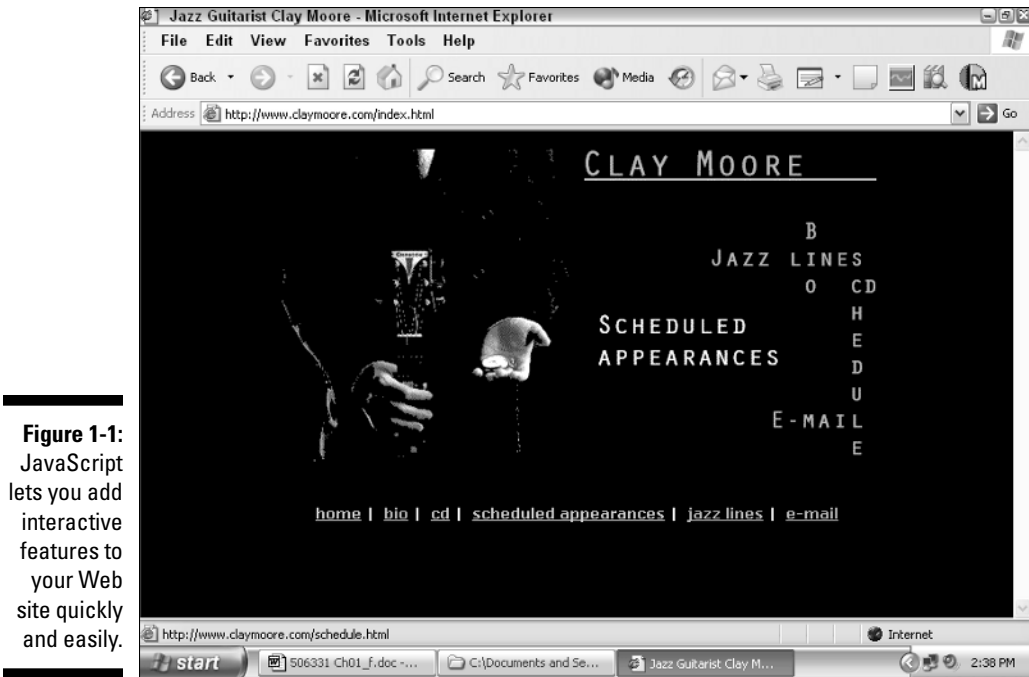
- ▶ Understanding a working definition of JavaScript
  - ▶ Dispelling common JavaScript misconceptions
  - ▶ Getting started with JavaScript tools
  - ▶ Finding information online
- 

**M**aybe you've surfed to a Web site that incorporates really cool features, such as

- ✔ Images that change when you move your mouse over them
- ✔ Slideshow animations
- ✔ Input forms with pop-up messages that help you fill in fields correctly
- ✔ Customized messages that welcome repeat visitors

By using JavaScript and the book you're reading right now you can create all these effects and many more! The Web page in Figure 1-1 shows you an example of the kinds of things that you can look forward to creating for your own site.

A lot has changed since the previous edition of JavaScript For Dummies came out. Perhaps the biggest change is the evolution of DHTML, or dynamic HTML. DHTML refers to JavaScript combined with HTML and cascading style sheets, and it's a powerful combination you can use to create even more breathtakingly cool Web sites than ever before.



**Figure 1-1:** JavaScript lets you add interactive features to your Web site quickly and easily.

Along with this increased power comes increased complexity, unfortunately — but that’s where this new, improved, better-tasting edition of *JavaScript For Dummies* comes in! Even if you’re not a crackerjack programmer, you can use the techniques and sample scripts in this book to create interactive Web pages bursting with animated effects.

Before you hit the JavaScript code slopes, though, you might want to take a minute to familiarize yourself with the basics that I cover in this chapter. Here I give you all the background that you need to get started using JavaScript as quickly as possible!

## *What Is JavaScript? (Hint: It’s Not the Same Thing as Java!)*

JavaScript is a scripting language you can use — in conjunction with HTML — to create interactive Web pages. A scripting language is a programming language

that's designed to give folks easy access to prebuilt components. In the case of JavaScript, those prebuilt components are the building blocks that make up a Web page (links, images, plug-ins, HTML form elements, browser configuration details, and so on).



You don't need to know anything about HTML to put this book to good use; I explain all the HTML you need to know to create and use the JavaScript examples that you see in this book. If you're interested in discovering more about HTML, I suggest checking out a good book devoted to the subject. A good one to try is *HTML 4 For Dummies*, 4th Edition, by Ed Tittel and Natanya Pitts (Wiley Publishing, Inc.).

## *It's easy! (Sort of)*

JavaScript has a reputation of being easy to use because

- ✓ The bulk of the document object model (the portion of the language that defines what kind of components, or objects, you can manipulate in JavaScript) is pretty straightforward.
- ✓ For example, if you want to trigger some kind of event when a person clicks a button, you access the `onClick` event handler associated with the button object; if you want to trigger an event when an HTML form is submitted, you access the `onSubmit` event handler associated with the form object. (You become familiar with the JavaScript object model in this book by examining and experimenting with working scripts. You can also check out Appendix C, which lists all the objects that make up the document object model.)
- ✓ When you load a cool Web page into your browser and wonder how the author created the effect in JavaScript, 99 times out of a 100 all you have to do to satisfy your curiosity is click to view the source code (choose `View` ⇨ `Page Source` in Navigator or choose `View` ⇨ `Source` in Internet Explorer). (Chapter 3 describes the `.js` files that are responsible for that 100th time.) This source code free-for-all, which is simply impossible with compiled programming languages such as Java, helps you decipher JavaScript programming by example.

However, becoming proficient in JavaScript isn't exactly a no-brainer. One of the biggest factors contributing to the language's growing complexity is the fact that the two major JavaScript-supporting browsers on the market (Netscape Navigator and Microsoft Internet Explorer) implement JavaScript differently. Netscape supports JavaScript directly — hardly a surprise because Netscape

was the one that came up with JavaScript in the first place! Internet Explorer, on the other hand, supports JavaScript indirectly by providing support for JScript, its very own JavaScript-compatible language. And despite claims by both Netscape and Microsoft that JavaScript and JScript, respectively, are “open, standardized scripting languages,” neither company offers explicit, comprehensive, all-in-one-place details describing all of the following:

- ✔ Precisely which version of JavaScript (or JScript) is implemented in each of their browser releases.
- ✔ Precisely which programming features are included and which objects are accessible in each version of JavaScript and JScript.
- ✔ How each version of JavaScript compares to each version of JScript. (As you see in Chapter 4, JavaScript and JScript differ substantially.)

The upshot is that creating cross-browser, JavaScript-enabled Web pages now falls somewhere around 6 on a difficulty scale of 1 to 10 (1 being the easiest technology in the world to master and 10 being the hardest).

Fear not, however. Armed with an understanding of HTML and the tips and sample scripts that you find in this book, you can become a JavaScript jockey in no time flat!

## What's in a name?

A long time ago, JavaScript was called *LiveScript*. In a classic “if you can’t dazzle them with brilliance, baffle them with marketing” move, Netscape changed the name to take advantage of the burgeoning interest in Java (another programming language that Netscape partner Sun Microsystems was developing at the time). By all accounts, the strategy worked. Unfortunately, many newbies still mistake JavaScript for Java, and vice versa.

Here’s the scoop: Java is similar to JavaScript in that they’re both object-based languages developed for the Web. Without going into the nitty-gritty details of syntax, interpreters, variable typing, and just-in-time compilers, all you have to remember about the difference in usage between

JavaScript and Java is this: On the gigantic client/server application that is the Web, JavaScript lets you access *Web clients* (otherwise known as *browsers*), and Java lets you access *Web servers*. (Note: In some cases, you can also use Java for Web client development.)

This difference might seem esoteric, but it can help you determine which language to use to create the Web site of your dreams. If what you want to accomplish can be achieved inside the confines of a Web client (in other words, by interacting with HTML, browser plug-ins, and Java applets), JavaScript is your best bet. But if you want to do something fancier — say, interact with a server-side database — you need to look into Java or some other server-side alternative.

## *It's speedy!*

Besides being relatively easy, JavaScript is also pretty speedy. Like most scripting languages, it's interpreted (as opposed to being compiled). When you program using a compiled language, such as C++, you must always reformat, or compile, your code file before you can run it. This interim step can take anywhere from a few seconds to several minutes or more.

The beauty of an interpreted language like JavaScript, on the other hand, is that when you make changes to your code — in this case, to your JavaScript script — you can test those changes immediately; you don't have to compile the script file first. Skipping the compile step saves a great deal of time during the debugging stage of Web page development.

Another great thing about using an interpreted language like JavaScript is that testing an interpreted script isn't an all-or-nothing proposition, the way it is with a compiled language. For example, if line 10 of a 20-line script contains a syntax error, the first half of your script may still run, and you may still get feedback immediately. The same error in a compiled program may prevent the program from running at all.



The downside of an interpreted language is that testing is on the honor system. Because there's no compiler to nag you, you might be tempted to leave your testing to the last minute or — worse yet — skip it altogether. However, remember that whether the Web site you create is for business or pleasure, it's a reflection on you, and testing is essential if you want to look your very best to potential customers, associates, and friends. (A few years ago, visitors to your site might have overlooked a buggy script or two, but frankly, Web site standards are much higher these days.) Fortunately, Chapter 17 is chock-full of helpful debugging tips to help make testing your JavaScript code as painless as possible.

## *Everybody's doing it! (Okay, almost everybody!)*

Two generally available Web browsers currently support JavaScript: Microsoft's Internet Explorer and Netscape/AOL's Navigator. (Beginning with version 4.0, Navigator became synonymous with Communicator, even though technically Netscape Communicator includes more components than just the Navigator Web browser.) Between them, these two browsers have virtually sewn up the browser market; almost everyone who surfs the Web is using one or the other — and thus has the ability to view and create JavaScript-enabled Web pages.

## JavaScript and HTML

You can think of JavaScript as an extension to HTML; an add-on, if you will.

Here's how it works. HTML tags create objects; JavaScript lets you manipulate those objects. For example, you use the HTML `<BODY>` and `</BODY>` tags to create a Web page, or document. As shown in Table 1-1, after that document is created, you can interact with it by using JavaScript. For example, you can use a special JavaScript construct called the `onLoad` event handler to trigger an action — play a little welcoming tune, perhaps — when the document is loaded onto a Web browser. (I cover event handlers in Chapter 13.) Examples of other HTML objects that you interact with using JavaScript include windows, text fields, images, and embedded Java applets.

<i>Object</i>	<i>HTML Tag</i>	<i>JavaScript</i>
Web page	<code>&lt;BODY&gt; . . . &lt;/BODY&gt;</code>	<code>document</code>
Image	<code>&lt;IMG NAME="myImage"&gt;</code>	<code>document.myImage</code>
HTML form	<code>&lt;FORM name="myForm"&gt;</code> <code>. . . &lt;/FORM&gt;</code>	<code>document.myForm</code>
Button	<code>&lt;INPUT TYPE="button"</code> <code>NAME="myButton"&gt;</code>	<code>document.myForm.</code> <code>myButton</code>

To add JavaScript to a Web page, all you have to do is embed JavaScript code in an HTML file. Below the line in which you embed the JavaScript code, you can reference, or call, that JavaScript code in response to an event handler or an HTML link.

You have two choices when it comes to embedding JavaScript code in an HTML file:

- ✔ **You can use the `<SCRIPT>` and `</SCRIPT>` tags to include JavaScript code directly into an HTML file.**

In the example below, a JavaScript function called `processOrder()` is defined at the top of the HTML file. Further down in the HTML file, the JavaScript function is associated with an event handler — specifically, the `processOrder` button's `onClick` event handler. (In other words, the JavaScript code contained in the `processOrder()` function runs when a user clicks the `processOrder` button.)

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
  // JavaScript statements go here
  function processOrder() {
    // More JavaScript statements go here
  }
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="myForm">
<INPUT TYPE="button" NAME="processOrder" VALUE="Click to process your
  order" onClick="processOrder();">
...
</HTML>
```

- ✓ You can use the `<SCRIPT>` and `</SCRIPT>` tags to include a separate, external JavaScript file (a file containing only JavaScript statements and bearing a `.js` extension) into an HTML file.

In the example below, the JavaScript `processOrder()` function is defined in the `myJSfile.js` file. The function is triggered, or called, when the user clicks the Click Here to Process Your Order link.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript" SRC="myJSfile.js">
</SCRIPT>
</HEAD>
<BODY>
<A HREF="javascript:processOrder();">Click here to process your order.</A>
...
</BODY>
</HTML>
```



Keep in mind that most of the examples in these printed pages focus on the JavaScript portion of the code (naturally!). But I include the HTML that you need to create the examples on the CD-ROM, so you don't have sweat re-creating the Web pages from scratch!

Because Web pages aren't made of HTML alone, however, JavaScript provides access to more than just HTML objects. JavaScript also provides access to browser- and platform-specific objects. Browser plug-ins (such as RealPlayer and Adobe Acrobat), the name and version of a particular viewer's browser, and the current date are all examples of non-HTML objects that you can work with by using JavaScript.



Together, all the objects that make up a Web site — HTML objects, browser- and platform-related objects, and special objects built right into the JavaScript language — are known as the document object model (DOM).

## JavaScript and Your Web Browser

You need to use Netscape Navigator 7.1 (or higher) or Microsoft Internet Explorer 6.0 (or higher) to use the latest JavaScript enhancements that I demonstrate in this book.



Not all browsers are created equal: Internet Explorer's support for JavaScript differs significantly from Navigator's, and support for JavaScript varies from browser version to browser version. For details, check out Chapter 5.

Although you can create and view JavaScript scripts with an old version of one of these browsers, I recommend that you install the most current version of Navigator or Internet Explorer. (What the heck — they're both free!) The latest versions of each product boast the very latest JavaScript features and bug fixes; they're also the versions that you see in the figures and examples in this book.

You can use another browser, such as Opera or America Online (or even another Internet protocol, such as FTP) to download the latest version of Navigator or Internet Explorer and try it out. The section "What Do I Need to Get Started?" later in this chapter is devoted to the ins and outs of obtaining and installing a JavaScript-enabled browser. For now, suffice it to say that

- ✔ You need Navigator or Internet Explorer to work with JavaScript, which means that you have to be running one of the client platforms that supports these browsers. (The Macintosh operating system and Windows both support Navigator and Internet Explorer.)
- ✔ You need to be aware that people might use other, non-JavaScript-enabled browsers to view your Web pages — or they might use JavaScript-enabled browsers with JavaScript support turned off. Either way, you have no way to guarantee that everyone who visits your page can view your JavaScript handiwork. (Check out Chapter 5 for more information on this topic.)

### JavaScript and browser security

In an era when computer viruses proliferate faster than crab grass, browser security is an important issue. You might be relieved to know that JavaScript poses no special security threats. Because JavaScript can't access any objects other than browser-contained objects (with the exception of cookies, which I discuss

in Chapter 6), no one can use JavaScript to open up secret dial-up connections, wipe users' hard drives, or perform other malicious acts, even by accident. In other words, JavaScript is subject to the security controls built into JavaScript-supporting browsers.

## What Can I Do with JavaScript That I Can't Do with Web Languages?

HTML. DHTML. XML. JavaScript. Java. Flash. When it comes to Web development, the sheer array of languages and development tools can be confusing — and you might be left wondering which language is best for which task.

The fact is that each language was designed with a particular kind of task in mind, and JavaScript is no exception. Table 1-2 shows you the types of tasks that JavaScript is best (and least) suited to perform. JavaScript is best suited for client-side (browser-based) tasks.

<i>Task</i>	<i>Is JavaScript Useful?</i>	<i>Are JavaScript and CSS (DHTML) Useful?</i>
Provide users with helpful feedback	Yes	No
Customize page appearance	Yes	Yes (more sophisticated than JavaScript alone)
Examine or change HTML form data	Yes	No
Create simple animations	Yes	Yes (more sophisticated than JavaScript alone)
Create complex animations	No	No
Perform server-side processing	No	No

JavaScript performs its magic by working together with HTML and cascading style sheets (CSS). Here's how it works: HTML and CSS let you create static Web pages by using tag building blocks, or objects. JavaScript lets you inspect and manipulate the objects to punch up static pages with interactivity and simple animations. (In other words, to use JavaScript, you need to use HTML; to take advantage of dynamic HTML, or DHTML, features, you need to use both HTML and CSS.)

By using JavaScript, you can make a Web site easy to navigate and even customize your page depending on who's viewing it, what browser the visitor is using to view it, and what time of day it is. You can even create simple (but effective) animated effects.

## *Make your Web site easy for folks to navigate*

The most common way to perk up your pages with JavaScript is to make them easier to navigate. For example, you can use JavaScript to

- ✓ Create expandable site maps.
- ✓ Add tooltips — helpful bits of text that appear when a user moves a mouse over a particular section of your Web site.
- ✓ Swap images when a user drags a mouse over a certain area of the screen. (This effect is called a mouse rollover, and it helps users determine at a glance which parts of your Web page are interactive, or clickable.)
- ✓ Inspect the data that your users enter and pop up helpful suggestions if they make an invalid entry.
- ✓ Display a thank-you message after a user submits a form.
- ✓ Load content into multiple frames when a user clicks a button so that the user can view multiple chunks of related information at the same time.



In addition to user-initiated events, such as clicking and dragging a mouse, JavaScript also recognizes automatic events — for example, loading a Web page onto a browser. (Check out Chapter 5 for details, including sample scripts that run in response to automatic events.)

## *Customize the way your Web site looks on-the-fly*

Everyone likes to feel special, and the folks who visit your Web site are no exception. By using JavaScript, you can tailor the way your pages look to different users based on criteria such as

- ✓ The specific kinds and versions of browser that visitors use to view your page
- ✓ The current date or time
- ✓ Your users' behaviors the last time they visited your pages
- ✓ Your users' stated preferences
- ✓ Any other criteria you can imagine

## *Create cool, dynamic animated effects*

Many folks assume that you need Java to create animations for the Web, but that's just not so. Although JavaScript certainly won't be mistaken for the most efficient way to create high-density animations, you can use JavaScript with cascading style sheets (the combination is sometimes known as DHTML) to create a variety of really neat animated effects. As a matter of fact, using JavaScript is the easiest way to implement common effects, such as rollovers, as you can see in Chapter 8.

## *What Do I Need to Get Started?*

I hope you're chomping at the bit to get started on your first JavaScript-enabled Web page! First things first, though . . . You have an idea of what JavaScript can do for you, and you might already have something specific in mind for your first attempt. Now's the time to dive into the preliminaries: what you need to get started and how to get what you need if you don't already have it. After you complete the setup, you can go on to the really fun stuff!

### *Hardware*

For the purposes of this book, I assume that you're beginning your JavaScript adventure with a personal computer or a Mac. Your machine (or box, to use the vernacular) should be a Pentium PC or better (unless it's a Power Mac) and should have at least 32MB of RAM and at least 25MB free hard drive space. If none of this makes sense, try asking your local hardware guru; every organization seems to have at least one guru. (I've found, through extensive trial and error, that most hardware gurus are fairly responsive to sugar-based snack foods.)

You also need hardware installed that lets you connect to the Internet. This hardware usually consists of a modem and a phone line, although some folks opt for even faster options such as cable or DSL (digital subscriber line). Depending on your computer, you might have an internal modem installed — many come complete with a built-in modem. If not, you can buy a modem at your local computer discount store. The differentiating factor among modems is line speed: the faster the better. (Most computers these days come with a 56.6 Kbps model preinstalled, but 28.8 works just fine.) If you don't already have a modem, consider buying the fastest modem in your price range; you'll be very glad you did when you try to look at spiffy Web pages with multiple graphics, each of which takes a loooong time to load (because graphics files are typically very large).

## Software

For the purposes of this book, I assume that you have a Mac OS 0 or later or a personal computer loaded with Windows 95, Windows NT, Windows 98, Windows 2000, Windows XP, or Linux. (Currently, only Netscape Navigator is available for use with Linux.)

I also assume that you have some way to create text files. (Most operating systems come packaged with a variety of text editors and word processors, any of which work just fine for creating JavaScript scripts.)



On the CD included with this book you can find some great text-editing utilities that are designed specifically for creating JavaScript files.

### *JavaScript-specific software*

You need a Web browser. Navigator (Netscape Communication's commercial Web browser) and Microsoft's Internet Explorer are the only generally available browsers that support JavaScript at the time of this writing. So, the first thing to do is to get a copy of Navigator or Internet Explorer. (The examples that you see in this book are demonstrated by using both Netscape Navigator and Internet Explorer running on Windows XP.)

Most personal computers come with Internet Explorer already installed. To find out if this is the case for your particular computer, choose Start→All Programs and look for Internet Explorer.

### *Netscape Navigator*

Netscape Navigator version 7.x bundles the Navigator browser with messaging, Web construction, and other Internet-related goodies.

You can download a copy by visiting the following site (which offers step-by-step installation instructions):

```
http://channels.netscape.com/ns/browsers/default.jsp
```

Of course, I'm assuming that you already have a Web browser installed or that you have access to FTP. (FTP is short for file transfer protocol, which is an Internet application that enables you to download files from other people's machines.)

### *Internet Explorer*

If you're a Microsoft buff, you might want to download a copy of Internet Explorer. Download it for free (or order your copy on CD-ROM for a nominal fee) from the following site, which offers easy-to-follow installation instructions:

```
www.microsoft.com/windows/ie/default.htm
```

## *Documentation*

For the latest Netscape Navigator and Microsoft Internet Explorer documentation and technical support, respectively, check out the following URLs:

```
http://channels.netscape.com/ns/browsers/default.jsp
```

```
www.microsoft.com/windows/ie/default.htm
```

To view or download a copy of the Core JavaScript Reference, the documentation from Netscape that explains JavaScript basics and language concepts, visit the following Web page:

```
http://devedge.netscape.com/central/javascript/
```

Microsoft's documentation for its JavaScript-compatible scripting language, called JScript, can be found at

```
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/script56/html/js56jsoriJScript.asp
```

or you can visit <http://msdn.microsoft.com> and search for documents on scripting.

