



Introduction



It would appear that we have reached the limits of what it is possible to achieve with computer technology, although one should be careful with such statements, as they tend to sound pretty silly in 5 years.

—John von Neumann (circa 1949)

In the early days of the Web, there was only HTML. HTML was a wonderful invention: it provided us with a simple, yet effective, way to share information, images, etc. with others regardless of their computer or operating system type. However, authors and Web developers soon discovered that HTML had one serious drawback: it is static and unchanging. Once a document loads, it sits there like a bump on a log. HTML alone cannot dynamically respond to a visitor's actions, other than to open a document in response to a click on a link. HTML cannot, for instance, ask for the visitor's name and then greet him or her personally. HTML alone cannot query a visitor's preferences, store them, and retrieve them on the next visit. In short, HTML alone cannot interact with the user or write content dynamically. Enter JavaScript.

JavaScript is an **object-based** scripting language specifically designed to make Web pages dynamic and interactive. Client-Side JavaScript, so named because it runs on the visitor's or *client's* computer, derives most of its objects from the Web page itself. For instance, an image on a Web page is an object that can be manipulated with JavaScript. We can change its source property

(`src` attribute) to display a different image in response to some user event such as the visitor moving the mouse over an image link. This is called an **image rollover**.

JavaScript allows us to obtain information from visitors in several ways. We can even store that information in a cookie and retrieve it on the user's next visit. Suppose the visitor prefers the non-frame version of a Web site. The next time she visits, we check the cookie, see that this person (actually this computer) prefers no frames, and refer the visitor to the appropriate Web document.

JavaScript is also wonderful for form validation. Because JavaScript runs on the client, the visitor's own computing power is used to validate form data, eliminating the need for long waits for a response from a **server-side** program. A server-side program is one that executes on the Web server using the resources and computing power of that computer. Server-side programs are more demanding on a server than client-side programs. The results of using Client-Side JavaScript to validate a form are a faster, perkier response for the user and less strain on the Web server.

✂ Rollovers	✂ Cookies
✂ Status Bar Messages	✂ Slide Shows
✂ Browser Detection	✂ Calculations
✂ Redirecting the Visitor	✂ Plug-in Detection
✂ Random Images and Quotes	✂ Random Sounds
✂ Pop-Up Windows	✂ Cycling Banners
✂ Form Validations	✂ Displaying Current Date
✂ Loading Multiple Frames at Once	✂ Display Last Modified Date

Table I.1 Popular Uses for JavaScript

The table above lists only a smidgen of JavaScript's capabilities. As you work through this book, you will quickly discover that the possibilities are endless, limited only by your imagination.

Other Approaches to Making Web Pages Dynamic and Interactive

Before the debut of JavaScript (and since), Web developers tried other approaches to creating dynamic Web pages including CGI (Common Gateway Interface) scripts, Java applets, client pull, server push, and a variety of plug-ins. While there are times when each of these methods may be a more appropriate choice than JavaScript, each has its drawbacks for client-side interaction. Let's take a closer look.

CGI Scripts

CGI scripts, often written in Perl (Practical Extraction and Report Language), are programs that run on the server and are capable of interacting with other applications and server

resources. CGI scripts can be quite powerful and are often used to provide Web-database interaction, create shopping carts, etc.

Perhaps the two most common ways that CGI scripts have been used to add interactivity to Web pages are through form validations and image maps. However, client-side image maps provide a more responsive alternative to CGI-scripted image maps. While form validations are still regularly performed by CGI scripts, JavaScript form validations typically execute faster and are often easier to implement. Please don't think that this makes CGI scripting useless, far from it. While Client-Side JavaScript can check to see if a zip code *looks* valid, a CGI script can determine if it *is* valid by looking it up in a database of valid zip codes. When it comes to opening, reading, and searching files, CGI scripting is the clear winner. JavaScript, except for the server-side version, is of no help whatsoever; it does not have permission to open, read, and manipulate files.

Java Applets

Another approach Web developers have used and continue to use to create interactive Web sites is **Java**. Java is a full-fledged, object-oriented programming language, unlike JavaScript, which is only an **object-based** programming language. Java was developed by Sun Microsystems to provide a **cross-platform** programming language. "Cross-platform" means able to run on a variety of operating systems and computer types. Java can be used to write complex applications as well as small programs, called **applets**, that can be embedded in Web pages and run by a browser.

All Java programs are first compiled into **byte code** and then interpreted by a Java Virtual Machine (JVM). "Byte code" is a term coined by Sun to describe its not-quite-machine-code equivalent. In the case of applets, the Java Virtual Machine is part of a Java-enabled Web browser.

Because Java is a full-fledged, object-oriented programming language, it is much more complicated to use, is not forgiving in the least (one missing semicolon can break your program), is **strongly typed**, and has a much steeper learning curve. Assuming you get past those barriers and successfully write your own applets, applets can still sometimes take a while to load.

Because JavaScript is still often confused with Java, let's take a closer look at their differences:

programmer's tip

Use JavaScript to validate a form, but back it up with a call to a CGI script form validator. That way if the visitor's browser doesn't support JavaScript, then the form data still gets validated by the CGI script. If the visitor's browser does support JavaScript, then that's one less form validation the server had to perform and you've lightened its load. Multiply that by thousands or even millions of transactions, and if the server has to perform 50% of the validations (though 5 to 10% is more likely), you've still lightened its load considerably.

JavaScript	Java
Object-based scripting language, uses built-in objects, but no classes.	Object-oriented programming language.
Developed for the express purpose of enhancing Web pages.	General-purpose language. <i>One</i> of its uses is to enhance Web pages through Java applets. Originally developed as a cross-platform language for home electronics.
Interpreted.	Compiled into byte code, then interpreted by a Java Virtual Machine. In the case of applets, the JVM is contained in the browser.
Loosely typed—don't have to declare a variable's type before using it. Netscape describes JavaScript as “dynamically” typed because a variable's data type can change during program execution. (We'll discuss data types in detail in Chapter3.)	Strongly typed—must declare a variable's type before using it. (We'll discuss data types in detail in Chapter 3.)
Code embedded into HTML.	Applet code is distinct from HTML because it is compiled. The applet itself is a class file, called by an applet HTML tag.
Forgiving syntax.	Strict syntax.
Easy to learn, easy to use.	Difficult to learn, complicated to use.
Not fully extensible. Has a limited set of base objects, properties, methods, and data types.	Fully extensible. Programmers can define their own classes.
Good for client-side interactivity.	Good for client-side special effects in the form of applets for client-server interaction and stand-alone applications.
Developed by Netscape Communications.	Developed by Sun Microsystems.

Table 1.2 *JavaScript vs. Java*

Client Pull

Client pull is a dynamic document mechanism whereby a Web page contains a directive, usually in a <meta> tag, that says, “Reload this document in seven seconds.” Well, it doesn't

use those words exactly, and it doesn't have to be seven seconds per se. The tag looks like this:

```
<meta http-equiv="refresh" content="7">
```

Alternatively, and more often, the directive might say, "Load this other document in seven seconds":

```
<meta http-equiv="refresh" content="7;
url=http://www.someplace.com/otherdoc.html">
```

Cool, huh? Notice that the quotation marks in the second example surround "7; url=http://www.someplace.com/otherdoc.html". This is correct, as the quoted text, including the time period and the URL, is the value for the content attribute.

Client pull is how most Web developers accomplish **splash pages**. Splash pages show a fancy graphic or run a Flash animation, then sweep the visitor on to the main site. Client pull is also excellent for "We've moved" Web pages and auto-running slideshows. The refresh value also works on images and audio files. Imagine using it to continuously update a live image from a camera feed!

Server Push

Server push is client pull's opposite. In this case, the server sends a requested document, but instead of terminating the connection as it normally does after serving a request, it maintains the connection and after some period of time delivers more data. The server has total control over when and how often data is sent. The major drawback of server-push technology is that it is extremely demanding on the server. This can be a problem if the server has few TCP/IP ports. One very cool thing about server push is that you can use it to send a single inline image; that is, the rest of the Web document can stay where it is and just the image changes. There is no need to reload the entire document when only that image changes (for example, a live camera feed) and the visitor won't see any blank screens between updates. Server-push applications are generally created with a CGI-compatible language like Perl or C, though they can be written in shell script as well.

Plug-ins

Plug-ins such as Shockwave, Flash, QuickTime, RealAudio, RealVideo, and Windows Media Player are wonderful ways to add interactivity and animation to Web sites. The major drawback of plug-ins is that the visitor must have the appropriate plug-in installed in order to access the animation, movie, sound file, etc. If the visitor doesn't already have the appropriate plug-in installed, chances are very high that he or she will simply surf on, rather than take the time to download and install the plug-in. The good news is that many newer browsers now have built-in support for Flash; users don't have to download the plug-in separately. Here's a short list of some of the most popular file types and their associated plug-ins:

File Extension:	.au
File Type:	Next/Sun audio format
Plug-in:	Apple QuickTime, Windows Media Player
Where to Get It:	http://www.apple.com/quicktime/download/ http://windowsmedia.com

File Extension:	.avi
File Type:	Audio-Video Interleaved (Windows video format)
Plug-in:	Apple QuickTime, Windows Media Player
Where to Get It:	http://www.apple.com/quicktime/download/ http://windowsmedia.com

File Extension:	.dcr, .dir, .dcr
File Type:	Shockwave
Plug-in:	Macromedia Shockwave Player
Where to Get It:	http://www.macromedia.com/shockwave/download/

File Extension:	.mid, .midi
File Type:	Musical Instrument Digital Interface
Plug-in:	Apple QuickTime, Windows Media Player
Where to Get It:	http://www.apple.com/quicktime/download/ http://windowsmedia.com

File Extension:	.mov, .qt
File Type:	QuickTime movie
Plug-in:	Apple QuickTime
Where to Get It:	http://www.apple.com/quicktime/download/

File Extension:	.mp3
File Type:	MPEG-1 Layer 3
Plug-in:	Apple QuickTime, RealJukebox, Windows Media Player
Where to Get It:	http://www.apple.com/quicktime/download/ http://www.real.com/jukebox/ http://windowsmedia.com

File Extension:	.mpeg
File Type:	Moving Pictures Expert Group
Plug-in:	Apple QuickTime, Windows Media Player
Where to Get It:	http://www.apple.com/quicktime/download/ http://windowsmedia.com

File Extension:	.pdf
File Type:	Portable Document Format
Plug-in:	Adobe Acrobat Reader
Where to Get It:	http://www.adobe.com/products/acrobat/readstep.html

File Extension:	.ra, .ram
File Type:	Real Audio
Plug-in:	RealPlayer
Where to Get It:	http://www.real.com/player/

File Extension:	.rm, .ram
File Type:	Real Video
Plug-in:	RealPlayer
Where to Get It:	http://www.real.com/player/

File Extension:	.swf
File Type:	Flash animation
Plug-in:	Macromedia Flash Player
Where to Get It:	http://www.macromedia.com

File Extension:	.wav
File Type:	Waveform (Windows audio format)
Plug-in:	Apple QuickTime, Windows Media Player
Where to Get It:	http://www.apple.com/quicktime/download/ http://windowsmedia.com

File Extension:	.wm, .wma, .wmv
File Type:	Windows Media, Windows Media Audio, Windows Media Audio/Video
Plug-in:	Window Media Player
Where to Get It:	http://windowsmedia.com

Table I.3 Popular Plug-ins

Summary

Hopefully this short introduction has given you a better understanding of why JavaScript has become the most popular scripting language on the Internet. As you read and work through this book, you'll learn how to add interactivity to your own Web sites with JavaScript.

Review Questions

1. What is the major benefit of HTML?
2. How is HTML limiting?
3. List five popular uses of JavaScript in the order of your interest in learning to integrate them into your own Web sites.
4. List three technologies, besides JavaScript, that Web developers have used to create dynamic, interactive Web pages.
5. What is a CGI script?

6. What kinds of functions can a CGI script perform?
7. What is Java?
8. What is a Java applet?
9. What kinds of things is Java good for?
10. What is client pull?
11. What is server push?
12. What are plug-ins?
13. List your three favorite plug-ins in order of preference.
14. What kind of application does JavaScript perform best? Why?
15. How does JavaScript differ from Java? What do they have in common?

Exercises

1. Make a list of at least three terms that you didn't understand from this chapter. Visit the supporting Web site for this book (<http://www.javascriptconcepts.com>). Use the searchable glossary to look up those terms.
2. Visit whatis.com. Enter any terms in your list of unknown terms that you didn't find in WebWoman's searchable glossary.
3. Visit the supporting Web site for this book (<http://www.javascriptconcepts.com>). Check the errata page to see if there have been any corrections/updates to the book and make the appropriate notes in your copy.
4. Visit the supporting Web site for this book (<http://www.javascriptconcepts.com>). Skim the list of frequently asked questions and read the details of any that you're curious about.
5. Many Web sites use JavaScript to add interactivity to their Web pages and to create special effects. Search the Web for sites (personal, professional, or commercial) that you think use JavaScript. Describe the interactivity or special effect you think was created with JavaScript. Explain whether you think the scripting was effective; that is, was it helpful, fun, useful, or just annoying? Would you use something similar on your Web site?