CHAPTER 1

What Is JavaScript to a Programmer?

In the beginning, there were Assembly and compiled languages. Later came scripting languages such as sed, awk, and Perl, which many programmers used to perform a variety of tasks. Followed by, in the late 80s and early 90s, the Internet, which exploded into a technological revolution that allowed anyone with a modem to communicate and retrieve information from around the world. As the Internet grew in number of users, it was obvious that an increase in functionality was needed in the browsers and the data they were rendering.

HTML, even with its advantages, was falling short of providing the control many developers wanted when creating Web pages and applications. This prompted the use of server-side programs, or scripts as they were often called, to handle some of the page dynamics developers needed from their sites.

These programs helped Web developers by allowing them to increase a site's functionality as well as process user-submitted information. However, *CGI*, or *common gateway interface*, scripts had to generate and return a response when the user sent incorrect or incomplete information. This led to the unnecessary back-and-forth transmission of data between browser and server. But, overall, it was a minor price to pay for the functionality it provided.

With time, and an increase in traffic, it became increasingly obvious that client-side intelligence was needed to offload some of the CGI functionality. Something was needed to perform this error checking and to decrease the amount of time a user spent connecting to a server to validate data. This would

also enable the Web site to offload some of its processing load to the browser machine, which meant an increase in the overall performance of a site.

It was partially this lack of client-side functionality and efficiency that helped spawn a new scripting language—one that could be executed within a browser's environment and not on the server. This language could be used to perform client-side tasks such as form validation and dynamic page content creation—one that would put the programming into HTML publishing. Welcome to the birth of JavaScript.

Welcome to JavaScript

On December 4, 1995, Netscape and Sun jointly introduced JavaScript 1.0, originally called LiveScript, to the world. This language, unlike its server-based predecessors, could be interpreted within the then new Netscape Navigator 2 browsers. As an interpreted language, JavaScript was positioned as a complement to Java and would allow Web developers to create and deploy custom applications across the enterprise and Internet alike. JavaScript gave Web developers the power to truly program—not just format data with HTML.

In addition to the client-side control developers desired, Netscape implemented server-side JavaScript. This allowed developers to use the same programming language on the server as they did in their pages for browsers. Database connection enhancements were added to the language (called LiveWire), allowing the developer to pull information directly from a database and maintain user sessions for common functionality such as shopping carts. JavaScript had truly bridged the gap between the simple world of HTML and the more complex CGI programs on the server. It provided a common language for Web developers to design, implement, and deploy solutions across their networks and distributed the overall processing load of their applications.

The next level of acceptance in the world of JavaScript was Microsoft's implementation of the language in its Internet Explorer 3 browser—the implementation was called JScript. Similar to Netscape, Microsoft also implemented the language on the server-side (JScript 2.0) within its ASP (Active Server Pages) environment. It also allowed developers the flexibility of using a common language on both the client and server-side, while providing many of the robust features, such as object invocation and usage, in compiled languages.

JAVASCRIPT VERSUS JSCRIPT, AND WHAT IS ECMASCRIPT?

JScript 1.0 was based on the published documentation from Netscape, so essentially it is the same thing as JavaScript 1.0. However, there were a few "features" that Netscape did not publish, as well as some functionality that was not re-created by Microsoft correctly. The result of this is that there are some discrepancies between JScript 1.0 and JavaScript 1.0.

Since the release of these initial browsers, JavaScript and JScript were both submitted to the ECMA (European Computer Manufacturers Association) standardization

body and have become the standard known as ECMAScript (ECMA-262). Because of this standardization, it is now considered that JavaScript is Netscape's implementation of ECMAScript while JScript is Microsoft's implementation.

The adoption of the first edition of ECMAScript occurred in June 1997 followed by its adoption by the International Organization for Standardization and International Electrotechnical Commission in April 1998 (ISO/IEC 16262). A second edition of the standard was approved by ECMA in June 1998, and a third edition was adopted in December 1999.

NOTE

Because Netscape's JavaScript was the foundation of all this, the book will refer to JavaScript, JScript, and ECMAScript simply as JavaScript except where a differentiation is needed.

So, what is JavaScript to the programmer? Well, in its purest form, it is an object-based, cross-platform, loosely-typed, multi-use language that allows a programmer to deploy many types of solutions to many clients. It not only involves adding functionality to Web pages as rendered within a browser, it also allows server-side processing for Netscape and Microsoft Web servers.

JScript has also been included in Microsoft's Windows Script Host (WSH), to allow programmers to write scripts to be executed on the operating system itself, and most recently as a major language under their .NET strategy (more on that later). When operating within the WSH environment, JScript is similar to the old DOS batch files, but gives programmers more functionality and versatility in what they can accomplish. This type of advancement has allowed the language to take hold in the computer world and continue to progress.

In addition to the benefits of these environments in which JavaScript can be executed, security measures are in place to protect end users against malicious code. Even though it is still young in terms of age, JavaScript is very mature and powerful. This functionality, ability, and versatility positions JavaScript as the best solution for many programmers.

Now that you've learned about what JavaScript is, you should dive a little deeper into what it means to a programmer. Being programmers ourselves, we know that a few strategically placed words do not make a language useful; so first, we'll look at the object-based characteristics of JavaScript.

Object-Based Technology

The fact that you are reading this reference somewhat implies that you have programmed in JavaScript or at least one other language before, even if only for one semester in college. Going one step further, I bet the language you programmed in was either C++, Java, or Perl—with each having various levels of object orientation (OO). Java specifically is OO by virtue of having all programmer created objects extend from core Java language classes or their own.