

JavaScript

Constantine Roussos, *Lynchburg College*

Introduction	1	JavaScript as a Client-Side HTML Scripting Language	9
The Internet, HTML, and the Need for a Scripting Language	1	The Document Object Model (DOM) Defined	9
Programming the Web—Java	1	DOM—Web Page Object Hierarchy	10
History of Javascript	2	HTML Element Control	10
The JavaScript Scripting Language	2	Events in the DOM	11
Java vs. JavaScript	2	Control of Java and Plug-in Components	11
JavaScript’s Success	3	External Scripts	11
Standards for JavaScript	3	JavaScript I/O	11
Advantages and Disadvantages of Javascript	3	Server-Side Scripting and CGI	12
The Popularity of JavaScript	3	Server-Side JavaScript	12
Low “Cost of Entry”	4	ASP and VBScript from Microsoft	12
JavaScript Functionality	4	PHP	12
What JavaScript Can Do	4	Perl	12
JavaScript Shortcomings	5	Common Gateway Interface (CGI)	13
Roles and Applications of JavaScript in the Internet and E-commerce	6	The Future of JavaScript	13
Principles of the JavaScript Language	6	A Core Web Technology	13
Scripting Languages Defined	6	New Technologies	13
A General Purpose Scripting Engine	6	Corporate Pacts	13
An Object-Based Language	6	Glossary	13
JavaScript Security	7	Cross References	14
Overview of JavaScript Syntax	7	References	14
Distinguishing Features of JavaScript Syntax	8	Further Reading	14

INTRODUCTION

JavaScript is most commonly used to enhance the functionality of Web pages. It enables the Web page developer to create a variety of dynamic responses to user actions. These responses include changing colors, moving images, performing computations, and making components appear and disappear. The JavaScript language has evolved with the World Wide Web and has established itself as a popular and important technology. It will continue to evolve in response to increasing user demand for greater Web functionality.

THE INTERNET, HTML, AND THE NEED FOR A SCRIPTING LANGUAGE

Prior to 1995 HTML documents were mostly plain text in paragraph format. In response to user demand for more functionality in displaying Web pages, in April 1995 Netscape Version 1.1 added tables and many more elements and attributes. Users, however, were accustomed to programs on their individual computers that interacted with them in dynamic ways. Web pages could do this in only the most minimal fashion. At the core of the problem was HTML. Based upon a commercial art markup paradigm, it was not designed to be a programming language. HTML “tags” specify where text and graphic images are to be placed, how large they should be, and what

their relationship is to other elements in a document. HTML did not possess the features of the programming languages that were used to create the sophisticated applications of the day. In order to meet the demand for more dynamic interactivity Web pages needed some kind of embedded programming language.

Programming the Web—Java

By 1995 the World Wide Web was ubiquitous and the need for a supporting programming language to manage Web content was painfully evident. The Java language, a precursor to JavaScript, was designed in the summer of 1992. Given the widespread acceptance of the C++ programming language it was no surprise that Java employed C++ syntax and its object-based approach to programming. Java’s first commercial application was to control an interactive, handheld home-entertainment device controller with an animated touchscreen user interface. On May 23, 1995 John Gage, Sun Microsystems, and Marc Andreessen, NetscapeCommunications, announced that Java programming language support would be incorporated into Netscape Navigator. In October 1995 Netscape Navigator version 2.0B1 included support for Java applets, small Java programs that ran within a browser window. The days of purely static Web pages had ended. Java incorporated methods to make Web pages dynamic in numerous ways. Java applets could run almost independently in their own windows using the Web page as

an enclosure. HTML elements could be modified with Java code, and Java could manage add-in components (plug-ins) that users could download. Except for restrictions due to security concerns the Web programming environment was now nearly as robust as that of programs running locally on a computer.

Unfortunately, Java did not provide solutions for all needs. Java applets were often slow. Many users were reluctant to download add-in components due to security concerns, inconvenience, and lengthy download times. Furthermore, using Java was more complicated to use than simply creating standard HTML documents. Non-programmers were now creating Web pages and many who were able to use HTML found Java difficult to master. Java appeared to be more of a tool for professional programmers. Also, even programmers were not entirely satisfied with the additional time required to create Java applets to perform relatively simple tasks like determining whether a user had filled an input box on a Web page. There continued to be a demand for a simpler means of creating and controlling sophisticated Web content. This demand set the stage for the development of a scripting language specifically designed for the Web page environment. A scripting language is a type of programming language that can be utilized without much of the complexity associated with traditional “full-strength” programming languages. Scripting languages had been utilized by programmers for many years for numerous purposes. Even many technically competent nonprogrammers had discovered that they could write useful code using a scripting language since its use does not require knowledge of compilers, linkers, object libraries, or most of the other technical roadblocks to writing computer programs.

HISTORY OF JAVASCRIPT

The JavaScript Scripting Language

Java was designed to be a complete programming language, and Java applets were typically created by professional programmers. The popularity of the Web derived in large part from the fact that nonprogrammers were able to deliver Web content. The introduction of Java brought forth a new level of Web page functionality that was not accessible to most nonprogrammers. Furthermore, much of what Java enabled programmers to do was quite simple in concept. For example, those using HTML saw no reason why operations like changing colors and images, modifying font size, and performing calculations interactively should require a sophisticated programming language. Netscape concluded that what was needed was a relatively simple language, embeddable in Web pages, that could perform these tasks and also enable the Web page designer to easily incorporate Java applets written by programmers. In early 1995 Netscape hired Brendan Eich to take charge of the design and implementation of such a new language. Eich decided that a loosely typed scripting language suited the environment and the audience. Since the purpose of the language was to bring life to static Web pages it was dubbed “LiveScript.” For the same reasons that Java employed C++ syntax and principles so did LiveScript. Due to marketing considerations the language was soon renamed JavaScript. On

December 4, 1995 Netscape and Sun jointly announced the new language in Netscape Navigator version 2.0B3, calling it a “complement” to both HTML and Java.

Java vs. JavaScript

The similarity of the names still causes confusion, as the perception is that there is a strong link between the two languages. There is not. The common C++ syntax shared by the two languages belies the significant differences in purpose, principals, and implementation.

Java is a fully functional object-oriented programming language. Java programs may be run at the command line as most other languages. Java programs may also be written as “applets.” These applets run in a window embedded in a Web page and have little direct access to Web page elements. JavaScript may only run embedded in an environment (normally a Web page) but was designed to directly interact with Web page elements. Java is often utilized by programmers for its graphics capabilities. JavaScript has no inherent graphic capabilities. It may, however, utilize the graphic capabilities of the environment in which it is running (the Web page). JavaScript code only exists as plain text. Java is compiled to a lower-level language that is interpreted by the Java Virtual Machine. This means that Java source code is not normally viewable to the user but that JavaScript code is.

Despite their significant differences Java and JavaScript have one important similarity. The driving force behind their popularity is that each is commonly and successfully used to enhance the functionality of Web pages.

```
<HTML>
<HEAD><TITLE>Temperature Conversion</TITLE>
</HEAD>
<BODY>
Temperature Conversion
<form name=TempConv>
Fahrenheit<BR>
  <input type=text name=Fahrenheit value=
32 >
  <input type=button name=ConvertF2C
value="Convert Fahrenheit to Celsius"
onClick="Celsius.value = Math.round
((Fahrenheit.value - 32)*5/9);">
<BR>
Celsius<BR>
  <input type=text name=Celsius value=0 >
  <input type=button name=ConvertC2F
value=
"Convert Celsius to Fahrenheit"
onClick="Fahrenheit.value = Math.
round (Celsius.value * 9/5 + 32);
"><BR>
</form>
</BODY>
</HTML>
```

Listing 1: JavaScript temperature conversion

In Figure 1, we see a simple Web page that enables the user to convert from Fahrenheit temperatures to Celsius

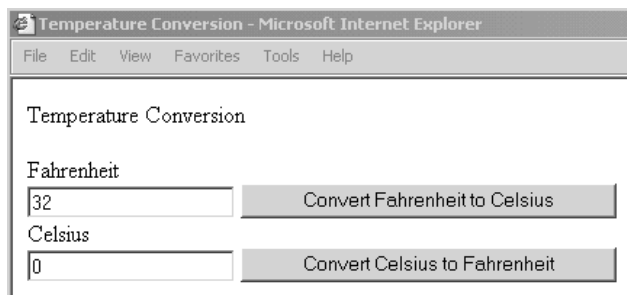


Figure 1: Temperature conversion Web page.

and vice versa. We also see the HTML and JavaScript code that created the Web page in Listing 1. The user of this Web page simply types a temperature into either text box and clicks on the appropriate button to perform the temperature conversion and write the computed value into the other text box.

Understanding the Code. The first line, `<HTML>`, tells the browser that this is an HTML document. The second line creates a title for the document and the third line defines the beginning of the body of the HTML code.

The next line of code simply displays the text “Temperature Conversion” `<form name = TempConv>` creates a form named TempConv into which the two text boxes and two command buttons are placed.

The first line beginning with `<input` creates a text box named Fahrenheit to hold a Fahrenheit temperature. The user may enter whatever numeric value he/she wishes into this text box.

The next line creates a button named ConvertF2C that the user should click on to convert Fahrenheit to Celsius. Up to this point all of the code has been pure HTML.

In the next line of code `onClick =` is followed by JavaScript code (in quotes) that is executed when the user clicks on the ConvertF2C button. The action of clicking on the button is referred to as an event. The code executed in response to this event is called an event handler. The formula embedded in the code is the standard formula for converting Fahrenheit to Celsius. `Fahrenheit.value` represents the value in the Fahrenheit textbox and the `round` method simply rounds a number to the nearest whole number (integer). In this case the whole number represents degrees in Celsius. `Celsius.value =` indicates that the new computed value is now to be placed into the Celsius text box, which has been created with another `<input` statement. The final `<input` statement creates the button that converts Celsius to Fahrenheit.

The JavaScript code (that enclosed in quotes after `onClick =`) may be immediately preceded by JavaScript: to identify it as JavaScript code. However, the browser assumes that the statements after `onClick =` are JavaScript from the context and from the fact that JavaScript is the default scripting language for HTML.

As one can readily see, the Web page functionality described above is made possible through the interaction between HTML and JavaScript. Dynamic HTML or DHTML is sometimes defined to be “the ability of the browser to alter a Web page *after* the document has loaded.” By this

definition the above is an example of DHTML. Some require greater activity such as animation or pop-up windows before the DHTML label is applied. More information on and examples of DHTML are given in the chapter on DHTML.

JavaScript’s Success

JavaScript became an immediate success because of its ease of use and its ability to increase Web page functionality and visual appeal. Its popularity continues to grow. Microsoft responded to the Sun/Netscape alliance and the popularity of JavaScript in a two-pronged strategy. In May 1996, with the release of Internet Explorer 3.0B1, Microsoft introduced its own Web scripting language called VBScript, which was based upon Microsoft’s popular Visual Basic programming language. At about the same time Microsoft also introduced its own version of JavaScript called “JScript.” At that time, however, Netscape was the dominant Web browser and would not execute VBScript code. JavaScript managed to stay one step ahead by adding significant additional features. With the final release of Internet Explorer 3.0 in August 1996 the browser wars began in earnest. Incompatibilities between the two languages and their interaction with browser environments were problematic for programmers and users alike.

Standards for JavaScript

Eventually programmers and users brought about significant pressure for standardization. The European Computer Manufacturers Association (ECMA) created ECMAScript, a standard for the core of Web scripting languages (ECMA, n.d.). The standardization effort began in November 1996 and a standard was adopted in June 1997 by ECMA and by the International Standards Organization (ISO) in April 1998. Although the core of the language was now standardized, the environments in which it lived (e.g., the Web browser) were not. Microsoft, Netscape, and dozens of other companies worked with the World Wide Web Consortium (W3C) to try to lay the groundwork for a truly universal document object model (DOM) that would be a compromise between an ideal standard and one that would be backward compatible with existing technologies to protect the billions of dollars invested in existing scripts. The document object model is a standard managed by the W3C. It defines the logical structure of HTML documents. More information about the DOM is given below.

ADVANTAGES AND DISADVANTAGES OF JAVASCRIPT

The Popularity of JavaScript

Probably the single most important reason for JavaScript’s, now immense, popularity is that its ease of use allows the greatest number of individuals to productively utilize this product. Just as the numbers of computers sold skyrocketed when IBM brought computing to the masses with the IBM-PC in 1981, so HTML and JavaScript have now enabled millions of individuals to produce the most

popular form of computer-based, information presentation: Web pages. Although designed to be easy to use, JavaScript has the functionality to be employed in very complex and powerful ways in the hands of professional programmers. The scripting language has established itself as a core technology of the Web and JavaScript users now span the range from personal homepage developers to Fortune 500 developers.

Low “Cost of Entry”

The term “cost” here refers both to money and time. The JavaScript interpreter comes with the Web browser so there is no additional cost to the programmer to write and run JavaScript code. JavaScript source code and executable code are the same so no development environment or compiler is required. An ordinary text editor is all one actually needs to produce executable JavaScript code.

JavaScript Functionality

Due to the fact that JavaScript programs download quickly and are interpreted and executed on the client computer, Web pages utilizing JavaScript are much more responsive than Web pages that only utilize Java applets or computer programs that must execute on the Web server. Even in cases where JavaScript cannot perform all of the programming tasks required in an application it is often able to significantly reduce the amount of processing that must take place on the Web server. JavaScript’s functionality has steadily increased as the language has developed. For example, add-in components (plug-ins) may now be detected and loaded with JavaScript. Due to its pragmatic approach it has incorporated many of the best features of several popular programming languages.

Although JavaScript has many powerful features that expert programmers might utilize, the language was designed to be easy to use for beginning programmers and even nonprogrammers. A search of the Web will reveal numerous Web sites promoting literally millions of lines of JavaScript code that may simply be copied into a Web page. This code is often packaged nicely within functions with an explanation of the purpose of each function. Additionally, JavaScript tutorials, reference materials, and commentary abound on the Web. A Web search or walk through the local bookstore will reveal the fact that more than one hundred books on JavaScript are now available. With the establishment of standards such as ECMA-262 and DOM, JavaScript is now more portable than ever.

What JavaScript Can Do

Dynamically Manipulate Web Page Content

JavaScript, as a scripting language embedded in HTML documents, is a means of dynamically controlling Web content. JavaScript can be used to modify almost any attribute of any object or element of the Web browser. For example, JavaScript can be used to change the color, size, and position of HTML elements in response to user events such as moving the mouse or pressing a key. These elements may include images, tables, input boxes, and blocks

of text. It can be used to make elements visible or invisible. It can cause the browser to navigate to another Web page or to move forward or back in its history list. It can even create new browser windows with content that JavaScript defines. That is, JavaScript can dynamically create HTML to be executed by the browser.

The ability of JavaScript to dynamically change Web content is referred to as dynamic HTML. Some of the most popular dynamic uses of JavaScript are the following.

Create crude animation by loading images sequentially to simulate movement similar to how the sequential individual frames of a movie tape simulate movement. JavaScript may also change the position of individual images in the document window.

Cause text to change color and size or cause images to change in response to mouse movements to indicate the action that may be invoked by clicking the mouse over a particular object.

Cause text to appear or disappear in order to make information available to the user or to reveal additional links to other documents when the mouse travels over a specific area on the screen. This facility is quite useful in producing drop-down menus.

Additionally, JavaScript is adept at creating and managing the content of frames in Web pages. JavaScript’s ability to modify style attributes in cascading style sheets (CSS) accounts for much of its power to modify Web page elements.

For example, the following JavaScript code causes the background color of a text box named *area* to turn yellow. The input box is contained in a form named *CircArea*:

```
<script>javascript:CircArea.area.style.  
backgroundcolor='yellow';</script>
```

Validation of Data. JavaScript is particularly powerful and useful when used in conjunction with forms. JavaScript can determine the content of text boxes, which radio buttons are turned on, which check boxes are checked, and what item(s) the user has selected from a list of choices from a list box. It can then check the data it has acquired for correctness, accuracy, and consistency and communicate its finding to the user.

Computation. JavaScript can perform arithmetic calculations using data it has acquired from the user and other sources such as internal program code. For example, JavaScript can compute the total cost of an order based upon the cost and quantity of the items selected, sales tax rates for various states, shipping costs for different destinations and item weights, and other factors without having to access information outside the HTML page. Numerous general and special purpose calculators have been programmed entirely in JavaScript.

Example: Data validation and computation. The following JavaScript function, *ComputeArea*, returns the area of a circle if its argument, *radius*, is a valid floating point

number and returns the string "Invalid radius" if the radius is not valid:

```
function ComputeArea(radius) {  
    if (parseFloat(radius)) return Math.  
        PI*radius*radius;  
    else return "Invalid radius";  
}
```

Use Methods

JavaScript may also invoke methods that have been defined for HTML objects and elements. In simplest terms a method is an operation that an object can perform. The method may take parameters. One example of a method is displaying the alert pop-up dialog boxes we often see. The browser window is the object that displays the alert pop-up dialog box. The text displayed in the box is the parameter of the alert method. A JavaScript statement to display an alert box containing the message "hello" is the following:

```
window.alert('hello');
```

Note that the format of the above statement is the *object name* (window), followed by a *dot* followed by the *method name* (alert), followed by *parameters in parentheses* ('hello').

Additionally JavaScript can invoke and, to some degree, control other components that have been loaded into the browser. These include Java applets and plugins such as controls that play music or display video. (See the example under Control of Java and Plugin Components.)

Respond to Events

When JavaScript programs change the content of a Web page they normally do so in response to user-initiated events. Events typically utilized by JavaScript include the following:

Clicking on a button;
Passing the mouse over an element such as a link or image;
and
Leaving or entering a Web page.

Communicate with the Server and Other Web Pages

JavaScript can dynamically create and pass information to programs that reside on the Web server. It does this through the use of forms and query strings. Web pages that contain forms can use JavaScript to cause the contents of the elements of a form to be passed to a program on the server. Although ordinary HTML can also cause form data to be passed to the server via the SUBMIT button element, JavaScript has much more robust capabilities. For example, JavaScript can analyze data entered by the user, perform computations using the data, and determine the appropriate server to which to send the data. Additionally, a programmer can send information that may not be contained in form elements to the server by using a query string. The query string normally consists of the address or URL (uniform resource

locator) of a program on the server followed by the question mark character followed by the information to be sent to the server. This same method may be used to send information to other Web pages that may be invoked or even created with JavaScript.

Example. The following line of JavaScript code loads a Web page into the browser window and passes it information:

```
window.location=http://GlobalU.edu/  
admissions/Info.htm?frosh=500
```

Understanding the code. The destination Web page is located on the main Web server at GlobalU.edu. It is in a folder named admissions. The Web page is named Info.htm. The information being passed to Info.htm is named "frosh" and its value is 500.

Save Server-Side Communication and Processing Time

Delays in the presentation of Web page information are usually due to the slowness of Web browser/Web server communications. Some operations, however, such as changing the color of a Web page link as the mouse passes over it appear to happen almost instantaneously. The reason is that the code to perform that operation is executed locally on the client machine. No communication with the Web server is required. JavaScript can be used to reduce communication with the server in a number of ways. For example, a Web page that contains a registration form typically contains some required fields like, perhaps, the registrant's name. If the form is submitted to the server with a blank name field the server will have to send back a Web page requesting the user to fill in the name field before the form can be processed. This communication with the server can be eliminated. When the user clicks on the submit button JavaScript can determine whether all of the required fields have been properly completed. If they have not, then it can notify the user. This requires no communication with the server. If JavaScript determines that all required fields have been properly completed, it can then submit the form information to the Web server.

JavaScript Shortcomings

Several of the items listed below under the category of "shortcomings" are intentional restrictions on the JavaScript language. They are listed because the missing functionality is present in many other languages.

Because it is an interpreted language JavaScript program source code cannot be effectively hidden from the user or other programmers. JavaScript has no object libraries although source libraries may be used equally effectively for some purposes. Additionally, interpreted languages typically run much more slowly than compiled languages and JavaScript is no exception.

JavaScript is a loosely typed language. The JavaScript interpreter often makes decisions about how to interpret the content of variables depending upon the context. This fact makes it easier for novices to use the language but can occasionally hide subtle and serious programming errors. Thus, in critical applications where errors may result in

dire consequences JavaScript would probably not be the language of choice.

Primarily due to the fact that JavaScript executes within a host application it does not have its own development environment or debugger. It is dependent upon the host application to provide these. Various vendors including Netscape (Netscape JavaScript Debugger) and Microsoft (Visual Studio Interdev) provide such environments but, typically, they are not as comprehensive as those supplied for C++, Visual Basic, and other languages.

Client-side JavaScript executes solely on the client computer. It has no access to server resources such as databases, files, or information about the client that may be stored on the server. JavaScript must rely on programs located and run on the server to retrieve such information. This protects the server from being compromised by malicious JavaScript code. Only programs that reside on the server may access server resources. These programs may validate the source of requests for data if needed.

For technical and practical reasons, JavaScript has no direct access to the underlying hardware (such as registers or I/O ports) of the computer on which it is executing. Contrast this to C and C++ where the programmer potentially has almost complete control. JavaScript cannot be used for “real-time” applications such as monitoring sensors and controlling switches. Without this restriction the possibility would exist for malicious JavaScript code to damage the client machine through actions like reformatting the hard drive or overwriting the system BIOS.

ROLES AND APPLICATIONS OF JAVASCRIPT IN THE INTERNET AND E-COMMERCE

The World Wide Web has proven itself to be important if not essential for companies to present and market their wares. Consumers today look for the comparative shopping information they need on the Web. Shoppers expect to be able to browse through catalogs, be advised of current sales and place orders—all online. JavaScript has and will continue to play a significant role in this environment. Because JavaScript executes much faster than server-based programs programmers try to place as much functionality in the JavaScript code as possible. Although entire e-commerce applications have been built using JavaScript the norm is that JavaScript is used as a supplement to other technologies.

JavaScript is also used extensively in intranets, Web-based networks that are internal to a company. These may be used for distributing personnel policies, coordinating company projects, and numerous other purposes. Intranet information content tends to be maintained by departments rather than by a company’s IT personnel. Thus, nonprogrammers are often tasked with maintaining the information. JavaScript’s ease of use makes it a good candidate for bringing greater functionality to intranet Web pages.

Finally, JavaScript’s programming language capabilities together with its ability to control Java applets and plug-ins means that functionality, special to a particular application, may be incorporated into Web pages with a

minimum of effort. For example, a lending institution can easily place a loan calculator on a Web page to enable potential customers to calculate monthly payments. A music store can utilize a music player control to allow potential customers to hear samples of CDs. Other technologies can be used to implement such functionality but each requires some special knowledge on the part of the programmer. Only JavaScript has demonstrated the ease of use and versatility required of a one-size-fits-all tool.

PRINCIPLES OF THE JAVASCRIPT LANGUAGE

Scripting Languages Defined

“A scripting language is a programming language that is used to manipulate, customize, and automate the facilities of an existing system” (ECMA-262 standard document, ECMA, n.d.). The scripting language is a mechanism for exposing already available functionality to program control and is intended for use by both professional and non-professional programmers. The existing system provides a “host environment” of objects and facilities.

A General Purpose Scripting Engine

JavaScript is a general purpose, cross-platform, object-based, scripting language. The most common use of JavaScript is, of course, as an interface to HTML objects and elements in Web pages. However, JavaScript may be embedded in any number of applications. For example, a C or Java application may be made scriptable using JavaScript just as Netscape makes its Web browser and Web server scriptable using JavaScript. To facilitate this the JavaScript engine including a JavaScript interpreter is available from <http://www.mozilla.org> for licensing by third parties for inclusion in their client and server products and various tools. The engine includes an application programming interface (API) for developers to expose their own objects to the JavaScript programming environment. JavaScript statements may reference these objects and elements. For example, one might consider defining objects such as paragraphs and tables in a text-processing system and enable a scripting language like JavaScript to manipulate these objects. A method called `changeFont` might be exposed to the scripting language. If `P1` is the name of a paragraph then the statement `P1.changeFont(size:12)` would change the size of paragraph `P1`’s font to 12. The core scripting language would understand the format of statements of the form `Object.Method(Parameters)` and would know that this requires a function call. Creation of computer code to actually record and display a change of font size would be the responsibility of the programmer implementing the embedding.

An Object-Based Language

The object-based programming paradigm has proven to be a very powerful one and JavaScript’s implementation of objects is quite effective, especially for its intended environment. JavaScript supports classes (in the form of object prototypes), objects, attributes, and methods.

Because JavaScript's syntax is so similar to C++ and Java, programmers skilled in these languages generally find no difficulty in utilizing JavaScript's somewhat different implementation of objects, attributes, and methods.

JavaScript Security

The core JavaScript language has no facility for interacting with the outside world and so has no security issues. However, JavaScript is most commonly utilized as an HTML scripting language and security issues abound in the World Wide Web environment.

Inherent Security

Once the Web evolved beyond purely static HTML documents the threat of security breaches was introduced. A program executing on a user's computer has the potential to do damage or appropriate confidential information such as passwords or credit card numbers. To address these issues and gain users' confidence strict security restrictions were placed upon Java and JavaScript programs. In particular, file access was severely limited. JavaScript may write cookies and read cookies written from scripts originating from the same server. Cookies are special files that a Web server, through a script, can write to the user's disk. The location, size, and content of the cookie file are restricted. Only one cookie file per Web server is allowed on a user's system. The Web server may later, through a script, read the information it placed in the cookie earlier. Additionally, the World-Wide-Web Consortium's HTML standard contains the *type = file* input control. This allows cooperating users to upload local files to a Web server through a Web page.

With the exception of loading URLs and sending form data to a server and to e-mail addresses, JavaScript has no network capabilities and so cannot be an agent in most of the common Internet security exploitations. Still, numerous, complex, and/or subtle security holes have been found in JavaScript under certain software configurations. Known flaws have been fixed but no one can predict when others may surface. Experience has shown that diligence in employing security mechanisms can reduce the likelihood that a security hole can be exploited.

Same Origin Policy

JavaScript employs the *same origin policy*, which specifies that when loading a document from one origin a script loaded from a different origin cannot get or set certain *predefined* properties of certain browser and HTML objects in a window or frame. That is, the policy prevents a script that originated from one server and is running in a browser window from accessing potentially confidential data located in a different browser window whose content originated from a different server. The policy also applies to cookies since cookies written by one server may contain passwords or credit card information that an intruding script should not be able to access.

Security Zones

Microsoft has implemented a security mechanism called *security zones*. Briefly, security zones allow the user to group Internet sites by specified levels of trust. Unfortu-

nately, the technology does not enable the user to specify in complete detail what privileges may be granted or revoked. Additionally, security zones are not supported by Netscape, making it difficult for programmers to utilize this security mechanism.

Signed Digital Certificates

Through Netscape, JavaScript can utilize *digital certificates* and *signed scripts* to address identification, authentication, and privacy concerns. A digital certificate is an electronic identification that the creator of a JavaScript program attaches to a signed script (program). Digital certificates employ advanced cryptographic methods to enable the recipient of a message to confirm the identity of the message sender and ensure that the message content was not altered in transit.

The *signed script* policy for JavaScript is based upon the Java security model, called *object signing*. To make use of the new policy in JavaScript, the programmer must use the new Java security classes and then sign the JavaScript scripts. Signed scripts can request, from the user, privileges and the lifting of certain security restrictions. The digital signature allows the user to identify the author of a JavaScript program. By signing the JavaScript program the signer acknowledges itself as the author and accepts responsibility for the program's actions. Unfortunately Microsoft's Internet Explorer does not support signed scripts. Ease-of-use issues for programmers and users have also played a role in preventing signed scripts from being widely employed.

Downloaded Software

Perhaps the greatest potential for security compromise is downloaded software. Once a program is running on a user's computer there is little or no protection from malicious code. Unfortunately some devious individuals are very clever about tricking users into loading software onto their systems. Some viruses, worms, etc. come disguised as e-mail messages or image files. Others may masquerade as ActiveX controls, plug-ins, or even security patches. JavaScript can be used to attempt to trick users into downloading and running such damaging code. The safe and simple rule is *do not download* unless you are sure of the origin and purpose. The rule not only applies to the Web but also to programs on CD or floppy diskette.

OVERVIEW OF JAVASCRIPT SYNTAX

JavaScript is an interpreted, high-level, object based language with syntax similar to C and Java. As is C and Java, JavaScript is case-sensitive. It follows naming conventions similar to C and Java as well. Significant particulars about JavaScript syntax are given below.

Competing versions of JavaScript and incompatibility in the way browser objects were exposed to JavaScript fueled an outcry from programmers, vendors, and users for standardization of the core language and the browser environment. A standard (ECMA-262) for the core language was adopted in June 1997 by the ECMA and governs current implementations of JavaScript (ECMA, n.d.). In June 1998 the ECMA General Assembly approved the second edition of ECMA-262 to keep it fully aligned with the

ISO/IEC 16262 standard of Joint Technology Committee 1 of the International Standards Organization and the International Electrotechnical Commission.

Although some inconsistencies exist, nearly all current implementations of the JavaScript interpreter for the core language are ECMA-262 compliant. In order to ensure compatibility between browsers, however, browser developers must ensure that their products are compliant with the current DOM standards of the W3C as this standard affects how JavaScript references objects in an HTML document. Incompatibilities still exist in the browser environment. Both the ECMA-262 and the DOM standards have been updated over the past several years and will continue to be into the foreseeable future. Thus JavaScript interpreters must also be updated to remain conformant.

Distinguishing Features of JavaScript Syntax

JavaScript programming constructs are very similar to those of C++ and Java. For example, comments in JavaScript begin with two forward slashes (`//`). All text after `//` on a line are ignored by the JavaScript interpreter.

```
//This entire line is a JavaScript comment  
and is ignored by the interpreter  
x = x+1; //Only the text past the first  
// is ignored on this line
```

Listing 2: JavaScript comments

We will now look at a few distinguishing features of the language.

Primitive Data Types

Primitive data types are Number, Boolean, and String. All numbers in JavaScript are floating points. Objects are complex data types. JavaScript supports built-in objects and user-defined objects. Objects are comprised of one or more components called properties or attributes that may be of any data type. Objects may also contain methods, which are functions specific to a particular object.

Literals

JavaScript supports the usual literals for primitive data types as well as special character sequences and named constants. Additionally, JavaScript has special constants such as NaN (a value that is not a number), undefined (the content of an uninitialized, declared variable), MAX_VALUE (the largest representable number), and POSITIVE_INFINITY.

Variable Declarations

Variable declarations are recommended but optional in JavaScript. One may assign a value to an undeclared variable but may not read the value of an uninitialized, undeclared variable.

Loops

JavaScript supports the common loop constructs found in C++. Additionally it supports the for-in loop construct. The following example displays the name and value of each property of the document object.

```
<script language=javascript>  
  //write the name and value of each  
  property of a document  
  for (var prop in document) {  
    document.write(prop + " = " +  
      document[prop] + " <BR>");  
  }  
</script>
```

Listing 3: A for-in loop

Note that the *document* object is contained in *window* so technically we should refer to `window.document`. However, it is common practice to omit the reference to `window`.

Arrays

Arrays in JavaScript are more flexible than in most other programming languages. Array elements may contain any data type. Arrays are accessed by reference. That is, the accessing program code is given the starting location of the array elements. Individual data elements are accessed by value; the accessing program code is given a copy of the value contained in that element.

JavaScript supports several different types of array declarations.

```
var eArray = new Array(); //declare an  
array with no elements  
  
var uArray = new Array(12); //declare an  
array containing 12 undefined elements  
  
//declare an array and initialize it with  
5 integers  
var oddNums = new Array(1, 3, 5, 7, 9);  
  
//declare an array and initialize with  
5 elements of differing data types World,  
var mixedData = new Array("Hello World",  
1.5, 2, false, "bye");  
  
//Array literals may be defined using  
square brackets  
  
var evenNums = [0, 2, 4, 6, 8]
```

Listing 4: JavaScript array declarations

Array elements are indexed/accessed with integers beginning with 0. So, for example, the following code places the number 5 into the zeroth element of array `numArray`:

```
numArray[0] = 5;
```

The length of an Array in JavaScript is determined in an unusual manner. In the example declaration of `uArray` above, we declared an array containing 12 undefined elements so the length of `uArray` is 12. If we wish to increase the number of undefined elements to 25 we may simply change the length property of `uArray` to 25:

```
uArray.length = 25; //change the length  
of the array to 25
```

Additionally we may add a new element anywhere in the array, even past its current length:

```
//put a value in the 49th element of
uArray, increasing its length to 50 (0, 1,
.., 49) uArray[49] = "dog"
//now uArray[0] through uArray[48] contain
undefined values.
```

Conversely we can truncate an array by setting its length to a value smaller than its current length:

```
uArray.length = 40; //array elements
40 - 49 no longer exist
```

We may determine which values of an array are undefined by comparing each array element to the special *undefined* constant:

```
for(var indx = 0; indx < uArray.length;
indx++) {
if (uArray[indx] != undefined) document.
write(uArray[indx]);
}
```

Listing 5: Writing the defined elements of an array

JavaScript supplies the programmer with several useful array methods. These include the following:

`concat()`—Appends data elements and/or arrays to the end of an array and returns a new array;
`sort()`—Sorts the elements of an array;
`join()`—Creates a string containing the elements of an array;
`slice()`—Returns a new array that is a subarray of the specified array; and
`splice()`—Inserts and removes array elements of an array.

Functions and Properties of the Global Object

Functions and properties of the global object are few but useful. The global object serves as a home for special functions and properties as well as programmer-defined global variables. For example, *isNaN* and *isFinite* are functions of the global object. These functions allow the programmer to test for valid and finite numbers respectively. Another interesting and quite useful JavaScript function of the global object is *eval*, which takes a JavaScript expression or statement as a parameter, executes it, and returns the result. Other “top-level” objects such as Array, Number, and Math are considered properties of the global object. Other commonly used global functions include `parseFloat()` and `parseInt()`, which convert strings to numbers, and `toString()`, which converts various objects including numbers to strings. For client-side JavaScript code the window object serves as the global object.

User-Defined Objects

User-defined objects in JavaScript are similar to but defined somewhat less rigorously than Java objects. Instead of classes JavaScript utilizes object prototypes, which

perform roughly the same function as classes. In fact the similarities are so great that we often refer to JavaScript classes. In keeping with JavaScript’s great flexibility we can define properties and methods that pertain only to an individual object or to all objects with the same prototype (i.e., same class).

JavaScript does not support pointers as C and C++ do. Pointers commonly provide semistructured access to the memory addresses of objects. Such access can expose an opportunity for malicious code to access memory that contains data that are private to other applications. JavaScript does support the use of nested (recursively defined) objects and the null object. Judicious use of this facility can be very powerful, eliminating the need for pointers in many circumstances.

JAVASCRIPT AS A CLIENT-SIDE HTML SCRIPTING LANGUAGE

The Document Object Model (DOM) Defined

“The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents” (Document Object Model, 2002). The DOM is a standard managed by the W3C and defines the logical structure of HTML documents. The DOM is language-independent. Its specifications allow programmers to build and modify documents. The W3C describes the DOM as a programming API for documents based on an object structure that closely resembles the structure of the documents it models. With such a standard, programs that manipulate documents should work correctly independent of the hardware or software that hosts the document. This means, for example, that a JavaScript function that changes the background color of a table in a particular HTML document should work consistently within Netscape Navigator and Internet Explorer. It should also not matter whether the Web server is Apache or Internet Information Server (IIS) or if the computer on which the document is located is running the Linux or Windows 2000 operating system. Although 100% compatibility has yet to be achieved, the DOM has greatly improved the portability of Web software from the earliest attempts at DHTML.

In recent years the DOM has grown from its *Core* to cover HTML, XML, CSS, events, and other features. Due to its increasing size and broader scope the DOM has been broken into modules to make it more manageable. Consequently most Web browsers and other software currently only fully support the DOM modules they deem essential.

Objects are, by nature, tree-structured and the DOM can be used to describe these objects down to the finest granularity. For example, there are DOM descriptions for the Web page window, for the document contained in the window, and for each individual element of the document. Although a complete DOM description of the Web page environment would be impractical to illustrate, a high-level picture of a few of the most important elements is useful (see Figure 2).



Figure 2: Some important elements of the DOM description of Web pages.

DOM—Web Page Object Hierarchy

Embedding JavaScript Code in Web Pages

Perhaps the most straightforward means of embedding JavaScript into a Web page is to simply put it in line with ordinary HTML statements. The JavaScript must be enclosed in `<SCRIPT>` and `</SCRIPT>` tags so that the browser knows that it is JavaScript code.

Web pages are often moved from one location to another. This can be frustrating for Web surfers unless the page's author leaves a Web page in the original location to redirect them to the new page. In the following example we create a Web page that redirects the browser to a new page. (See Figure 3 and Listing 6.)

```

<HTML>
<HEAD><TITLE>Our Page Has Moved!</TITLE>
</HEAD>
<BODY BGCOLOR="#E8F4FF" >
<CENTER><H2>Our Page Has Moved!</H2>
</CENTER>
<script language="javascript">
    alert("The page you are looking for
    has moved!");
    window.location.href="http:
    //localhost/newpage.htm";
</script>
</BODY>
</HTML>
    
```

Listing 6: Redirect the browser to another Web page

Understanding the Code. The first line, `<HTML>`, tells the browser that this is an HTML document. The second line creates a title for the document, and the third



Figure 3: Web page redirect.

line defines the beginning of the body of the HTML code and sets the background color to light blue.

The fourth line of code displays the text "Our Page Has Moved" as a number 2 size header and centers it.

The `<script>` tag tells the browser that what follows is JavaScript code.

The first JavaScript statement displays an alert box informing the user that the page has moved. After the user clicks OK the next line of JavaScript will direct the browser to a page named `newpage.htm`.

The `</script>` tag ends the JavaScript code.

The most common means of embedding JavaScript is by creating JavaScript functions and placing them in the HEAD of the HTML document. These functions may then be called from event handlers (see Events in the DOM) or other embedded JavaScript code.

Below is an html document that contains a JavaScript function that computes the area of a circle given the radius. The function is embedded in the HEAD section of the HTML document:

```

<html>
<head>
<!-- A JavaScript function to compute the
    area of a circle -->
<SCRIPT LANGUAGE = "JavaScript">
function ComputeArea(radius)
{return (Math.PI * radius * radius)}
</SCRIPT>
</head>
<body>
    <script>alert(ComputeArea(2));
    </script>
</body>
</html>
    
```

Listing 7: CircArea.htm computes the area of a circle

Understanding the Code. The `ComputeArea` function is embedded in the HEAD section of the HTML document above. The function must also be enclosed in `<script>`, `</script>` tags.

The *function* key word tells the browser that a function is being defined. After the function name, enclosed in parentheses is the function's input (parameter), the radius of the circle. The function executes the standard formula for computing the area of a circle and then *returns* the answer. Note that the instructions contained in a function are enclosed in curly braces `{` and `}`.

In the body of the HTML document is a JavaScript statement that displays an alert box containing the computed area of a circle with radius equal to 2. `ComputeArea(2)` causes the function to execute (perform its instructions) using the number 2 as the radius parameter.

HTML Element Control

Perhaps the most common and important use of JavaScript is to modify attributes of HTML objects and elements. These objects and elements include tables, images, forms, frames, the current Web page, and the status bar to name a very few. Attributes they possess that may be

programmatically controlled include color, text content, size, and position. JavaScript may also be used to send to the server data that users have entered into forms. It is also commonly used to store and retrieve data stored in cookie files.

Events in the DOM

JavaScript functions that control HTML objects and elements as well as plug-ins are usually activated by events. In the context of Web pages events may be defined as actions detectable by the Web browser. Examples of events include clicking on a button, moving the mouse over an element, navigating to a new page, or pressing a key on the keyboard. For each event to which a programmer wishes to respond, he/she can define a JavaScript program that executes whenever the event occurs. Such a program is called an event handler.

Example—Event Handling

In this example we define a table element called CoursePB that contains the text “Course” and three event handlers. The event handlers are simply single JavaScript statements. When the user clicks on element CoursePB the browser navigates to a page named course.htm. When the mouse is moved over the table element the word Course changes color to red and when the mouse is moved off the element the color of the word Course is changed to black.

```
<TD ID=CoursePB
  onclick=javascript:document.
    location="course.htm"
  onmouseover=javascript:style.color=
    "RED"
  onmouseout=javascript:style.color=
    "BLACK">
  Course
</TD>
```

Control of Java and Plug-in Components

JavaScript may be used to detect, define, and load Java applets and plug-ins. Plug-ins are software components that allow the processing of audio, video, and other data not directly supported by the browser. For example, JavaScript can detect whether the RealPlayer plug-in has been loaded onto your computer. If not, it can load the plug-in to play a streaming audio file from the Web.

JavaScript may also exercise limited control over such plug-ins. For example, it can start and stop the playing of the streaming audio in response to user action. Similarly, JavaScript can load, start, and stop Java applets.

External Scripts

Full-strength programming languages have sophisticated code-management capabilities. Due to the fact that JavaScript is interpreted and that it executes within the context of a host application (most commonly a Web browser) its code management features are minimal. The Web browser does, however, support the use of included files, a simple, yet effective, form of code management.

Skilled programmers write general-purpose functions that can be reused in many different Web pages. A JavaScript function that determines whether a required input box on a form has been completed and notifies the user if it has not is an example. Another similarly useful function is the example below. Function isBadNumber takes one parameter, a text box, and returns true if the content of the text box is not a valid number; otherwise, isBadNumber returns false.

```
function isBadNum(textBox) {
  if (isNaN(textBox.value) || (textBox.
    value == "")) return true;
  else return false;
} //end function isBadNum
```

Copying such a function into every Web page in which it is used can be very inconvenient for the programmer, make the page unwieldy, and waste resources. Additionally, as the programmer, over time, makes improvements to the function he/she will find that many older and less effective versions of the function still exist. Retaining a single copy of the function and having the newest version accessible to all Web pages that use it is preferable.

The following line of code when placed in an HTML file causes the JavaScript functions contained in the file named CalFuncs.js to become accessible to the HTML document:

```
<Script language=JavaScript src="CalFuncs.
  js"></Script>
```

JavaScript functions stored external to the Web page in which they are used are often referred to as *external scripts*. Only one copy of such functions need exist. When the function is updated, the most recent version is used in every Web page that calls that function the next time the Web page is accessed. Programmers do have to take care that the updated function will work in every previously created Web page that uses it.

JavaScript I/O

In most languages there are three separate types of I/O (input/output):

- Keyboard/mouse input and monitor/printer output to interact with the user;
- File I/O that reads and writes files to disk; and
- Network I/O that connects to and communicates with other computers.

Core JavaScript has no I/O capabilities. However, client-side JavaScript, through interaction with a Web browser, has a minimal set of I/O capabilities.

File and Network I/O

File and Network I/O in client-side JavaScript is severely restricted due to security concerns. JavaScript may write and read cookies and, by using the *file*-type HTML input control, initiate file uploads with user permission. With

the exception of loading URLs and sending form data to a server and to e-mail addresses, JavaScript has no network I/O capabilities. (See the section JavaScript Security.)

Interactive I/O through Dialog Boxes

JavaScript can interact with the user in two ways. The first is through the use of the window methods, `alert`, `confirm`, and `prompt`, all of which present dialog boxes to the user with increasing functionality.

Alert. The purpose of the `alert` method is to present a message to the user. After the alert dialog box containing the message is displayed, the user must press the OK button to clear the dialog box and resume the program.

Confirm. The `confirm` method presents a query to the user together with an OK and a Cancel button in a dialog box. The user must click one of the two buttons to clear the dialog box. The confirm function returns either true or false, depending upon whether the user clicked on OK or Cancel.

Prompt. The `prompt` method presents to the user a query, a text input box, and two buttons labeled OK and Cancel in a dialog box. The text input box may be blank or contain a default response. The user may type into the input box and must finally click on OK or Cancel to clear the dialog box and resume the program. Clicking OK will pass the data typed into the input box to the program; clicking Cancel or closing the prompt box will return null to the program.

Interactive I/O through Web Pages

JavaScript may also communicate with the user through the Web page that hosts the script. JavaScript may write directly to the Web page using the `write` method of the `document` object.

For example, the JavaScript statement

```
document.write('Today is ' + Date());
```

writes the host computer's system date to the Web page.

JavaScript may also use HTML elements to communicate with the user. For example, JavaScript may write to a text input box to communicate to the user and it may read what the user has typed into a text input box to receive communication from the user.

SERVER-SIDE SCRIPTING AND CGI

Server-Side JavaScript

Core JavaScript is a general purpose scripting language that can be embedded in most applications. Netscape's Web servers include a JavaScript interpreter, which makes possible the use of JavaScript as a server-side scripting language on these Web servers. Server-side JavaScript is composed of core JavaScript and additional objects and functions for accessing databases and file systems, sending e-mail, etc. Microsoft supports the use of JScript as a server-side scripting language. Although having a nearly identical language available on the Web server and the client is a convenience for programmers, server-side

JavaScript has not enjoyed great popularity. Instead several other technologies are commonly employed to interact with the server to access databases and other resources on the server. These technologies include ASP, Perl, and PHP. Client-side JavaScript programs may interact with these technologies.

ASP and VBScript from Microsoft

Microsoft's Web server, IIS, supports a number of technologies including the Microsoft-specific ASP (Active Server Pages) and VBScript. ASP files are programs that run through IIS on the server, are able to access databases, files, and other resources, and communicate with the browser on the client computer. VBScript is a client- and server-side scripting language. It is used most extensively for programming ASP files. After collecting information from the server the ASP program may send information and client-side programming code to the client in the form of a Web page. Typically client-side programs are written in JavaScript. JavaScript may also be used to invoke ASP programs and send information, such as the contents of forms, to an ASP program.

PHP

PHP is a widely used, general-purpose scripting language especially well suited for Web development. PHP is a project of the Apache Software Foundation. JavaScript code can be created using PHP code just as in ASP programs. As with ASP, PHP has access to server resources such as databases and environment variables. For example, the environment variable `$HTTP_USER_AGENT` holds the name of the Web browser the client is using to display the Web page. PHP files have an extension of `.php`. `<?php` denotes the beginning of a PHP script and `?>` denotes the end. Note that the abbreviated `<?` may also be used to denote the beginning of a PHP script. The echo statement is used to display data.

Example

The program `PHPTest.php` below writes a Web page containing the name of the Web browser (e.g., Microsoft Internet Explorer 6.0), the name of the Web server, and the system date on the client computer.

```
<html><head><title>PHP Test</title></head>
<body>
<?
echo $HTTP_USER_AGENT, "<BR>";
echo $SERVER_NAME, "<BR>";
echo "<script language=\"JavaScript\"
">\n"
echo "document.write('Today is ' + Date());
\n";
echo "</script>";
?>
</body></html>
```

Perl

Perl is an acronym for "practical extraction and report language." Created by Larry Wall, it is an interpreted

language optimized for string manipulation, I/O, and system tasks. It is widely used by Web programmers due to its functionality and flexibility. It is able to access databases and files on a server and send information back to the client via a Web page as PHP and ASP do. The Perl interpreter is located on the server. Perl scripts are executed by running the Perl interpreter through the CGI (common gateway interface—see below) and passing to the interpreter the name of the file containing the Perl script.

When executing a Perl script from the operating system command line one simply invokes the Perl interpreter and passes it the name of the Perl script with any parameters.

For example, the command

```
perl myscript.pl
```

will cause the Perl interpreter to execute the Perl program named myscript.pl.

In order to run a Perl script from a Web page the script is normally specified as the action object of a form. That is, the Perl script will be run when the form is submitted and the form data will be passed to the script. The example below illustrates this:

```
<FORM METHOD=POST ACTION="/cgi-bin/  
myscript.pl">  
  <!-- Input data elements -->  
  <INPUT TYPE=SUBMIT VALUE="Submit Data">  
</FORM>
```

Common Gateway Interface (CGI)

The common gateway interface (CGI) is not a programming language. It is a standard for interfacing external applications such as Web browsers to information servers such as Web servers. Most programming languages that run on a server can utilize CGI to implement two-way communication with Web browsers. CGI programs implement the CGI standard, run on the Web server, and, potentially, have access to all resources located on the server such as databases, files, and server-based environment variables. CGI programs are normally executed in response to requests from an HTML document (Web page). Normally, CGI programs retrieve information from the Web server and package that information into an HTML document that is then displayed on the client browser. The Web page created by the CGI program may contain JavaScript that can execute when the Web page is loaded into the browser or in response to events such as the user clicking on a button element or moving the mouse over an image.

THE FUTURE OF JAVASCRIPT A Core Web Technology

JavaScript has become a core Web technology. It is used by programmers and nonprogrammers alike. Millions of Web sites rely on it. These facts alone ensure that it will remain viable for a number of years. Additionally, JavaScript has evolved to meet the ever-increasing

demands of Web programmers. By all indications it will continue to evolve, improve, and become more powerful. As JavaScript gains more sophisticated programming capabilities in the future, developers will be able to write full-featured applications in JavaScript. It has demonstrated its resiliency by weathering bad publicity over a number of security breaches and has made some progress in developing standards to address security issues, although security will continue to be a concern. The establishment of the ECMA-262 standard has addressed browser incompatibility and future development issues. The ECMA technical committee is currently working on significant enhancements to the specifications and is also working with other standards groups to coordinate the development of the language. More programmers now code to the ECMA and DOM standards, thus putting pressure on browser manufacturers to ensure that their JavaScript interpreters conform to the standards.

New Technologies

As rosy as the future looks for JavaScript, there are no guarantees. Netscape's JavaScript interpreters are now "open source" hosted by mozilla.org, and many products have faltered when ownership appeared to be in question. Microsoft now owns the bulk of the browser market, and it has a reputation for going its own way. New technologies spring up regularly in the Internet world and these could have an impact on the viability of any current product. Microsoft has long promoted VBScript without great success but the expected popularity of Microsoft's .NET technology raises new questions for the future of Web-based programming. JavaScript interpreters must be continually updated to keep current with changing standards. Faced with competing technologies those standards may need to change rapidly. Keeping "free" software up to date in this environment may be a challenge.

Corporate Pacts

Agreements between major vendors are commonplace and have an impact on browser and other product development. On November 24, 1998, AOL announced that it had purchased Netscape (the creator of JavaScript) and consummated a separate but almost simultaneous strategic development agreement with Sun Corp. (creator of Java). The final impact of this has yet to be felt. Finally, it bears note that the World Wide Web is less than 10 years old. Ten years ago no one could have predicted the present state of network computing. Its state 10 years hence may be just as unknowable.

GLOSSARY

Attribute A component of an object (same as a property).

Cookie A computer file that may be written to or read from a specially designated location on a user's disk drive by a Web browser. The cookie typically contains information that a Web application will use at a future time.

Compiler A computer program that reads programming language statements, translates them into instructions that the computer hardware can execute, and writes the translated instructions to a file for later execution.

Debugger A computer program that aids a programmer to locate errors in program code.

Element A component of a Web page such as a text box, table, link, or command button.

Event An action, such as a mouse click, that can be detected by a computer program.

Function A sequence of programming language instructions executed in response to a call.

Hypertext markup language (HTML) A language used to specify the content of Web pages.

Hypertext transport protocol (HTTP) A protocol that specifies how Web page information is communicated over a network.

Integrated development environment (IDE) A computer program that provides a programmer-friendly environment in which a programmer can write, test, and debug computer code.

Interpreter A computer program that reads programming language statements and causes them to be executed while the interpreter is running.

Method A computer language function specific to an object. See function.

Object A complex data type that contains components that may be primitive data types such as numbers or strings of characters or complex data types such as other objects.

Property See attribute.

Prototype A model for a collection of one or more objects having a common structure.

Scripting language A programming language used to manipulate, customize, and automate the facilities of an existing system.

Source code Computer programming language statements that are readable by humans. Compilers and interpreters translate source code into code that a computer can execute.

Tag A command in an HTML document that defines an element or specifies an attribute of text or elements of a Web page. Tags are enclosed in angle brackets.

CROSS REFERENCES

See *Java*; *HyperText Markup Language (HTML) /Extensible HyperText Markup Language (XHTML)*; *Perl*.

REFERENCES

Document Object Model (DOM) (2002). Retrieved April 11, 2003, from <http://www.w3.org/DOM/>
ECMA (n.d.). Retrieved March 31, 2003, from <http://www.ecma-international.org>

FURTHER READING

Cohen, Y. (1997). *JavaScript cookbook*. New York: Wiley.
DevEdge Online Archive. (n.d.). *Technologies*. Retrieved December 27, 2001, from <http://developer.netscape.com/tech/>

Dickson, P. (2001). *Sputnik—The shock of the century*. New York: Walker.

Flanagan, D. (2002). *JavaScript: The definitive guide* (4th ed.). Sebastopol, CA: O'Reilly.

Free On-Line Dictionary of Computing (n.d.). Retrieved May 14, 2002, from <http://wombat.doc.ic.ac.uk/pub/darpa>

Gosselin, D. (2002). *JavaScript*. Boston, MA: Course Technology.

JavaScript language resources. (n.d.). Retrieved December 27, 2001, from <http://www.mozilla.org/js/language>

NCSA HTTPd Development Team. (n.d.). *Common gateway interface*. Retrieved from <http://hoohoo.ncsa.uiuc.edu/cgi/intro.html>

O'Reilly (n.d.). *The source for Perl*. Retrieved from <http://www.perl.com>

PHP Group. (n.d.). Retrieved May 14, 2002, from <http://www.php.net>

Sun Microsystems Inc. (n.d.). *Java*. Retrieved December 27, 2001, from <http://java.sun.com>

University of Albany Learning Technology Library. (n.d.). *The Internet & the WWW: A history and introduction*. Retrieved May 14, 2002, from <http://www.albany.edu/ltl/using/history.html>

W3C: *World Wide Web Consortium*. (n.d.). Retrieved December 27, 2001, from <http://www.w3c.org>

Wilton, P. (2000). *Beginning JavaScript*. Chicago: Wrox.