

# Understanding JavaScript

*JavaScript and JScript are both implementations of a notional standard language called ECMAScript, and are not related to Java. We explain the origin of these related languages and show how they can be used to enhance Web pages.*

*By Mike Lewis*

**W**ith the growing sophistication of the World Wide Web, it's easy to forget that HTML is little more than a simple rendering tool: a language which describes how text and graphics should appear in a browser. HTML pages have no inherent intelligence, they offer only limited user interaction, and they perform no behind-the-scenes processing. It was partly to make up for those deficiencies that tools like JavaScript were invented.

JavaScript is an example of a Web-based scripting language. Its aim is to allow browsers to go beyond the rendering of simple text and images, by adding a measure of intelligence to what would otherwise be static documents. By combining JavaScript with HTML code you can create pages which respond to user inputs, perform calculations, validate user-entered data, display alerts, and do quite a lot more. JavaScript is not a general-purpose programming language, but it can go a long way towards providing the functionality which HTML lacks.

## **Applications**

An example of how JavaScript might be used is for validating the details which a user enters in an order form for an online shopping site. Without JavaScript, or something similar, the browser would have to send all the user's data to the server, which would then perform the necessary validation. If any invalid data was detected, the server would generate a new page containing an error message and return this to the browser.

With JavaScript, the round trip can be avoided. All the validation and user interaction can be done within the browser, without the server being involved. The result is less network traffic, a reduced load on the server, faster turnaround and a more responsive application.

In this article, I will present an overview of JavaScript's capabilities, and show some simple ways of putting it to work. This is not meant to be a detailed tutorial - there are plenty of those available elsewhere (see the Resources box for some examples) - but it will give you a flavour for what JavaScript can help you to achieve.

Although I will refer to the language as JavaScript throughout the article, this is not strictly correct. JavaScript is the proprietary name of a language which was invented by Netscape. Microsoft has a similar language, called JScript. JavaScript and JScript are both implementations of a notional standard language called ECMAScript. Although I will use the term JavaScript for convenience, what you read here applies equally to JScript and to any future implementations of the ECMAScript standard.

## **Beginnings**

JavaScript first appeared in 1995, although its roots go back to an earlier Netscape project called LiveScript. The first browser to support JavaScript was Navigator 2.0. With successive releases of Navigator, Netscape took the opportunity to enhance the language. The current version is 1.3. Microsoft also took an early decision to support

JavaScript; the language is available in all versions of Internet Explorer from 3.0 (see also Figure 1).

In late 1996, Netscape and Microsoft joined forces to promote an international standard for the language, under the auspices of the European Computer Manufacturers Association (ECMA). The resulting standard, known as ECMA 262, was formally adopted in June 1997, and has since also been recognised by the International Standards Organisation (ISO). The standard contains the specification of the notional ECMAScript language.

When the ECMA standard first appeared, JavaScript stood at version 1.2. This did not fully conform to the standard - it did not support Unicode, and it was not fully platform-independent in certain respects. However, Netscape subsequently released a compliant version, 1.3, which appeared with Navigator 4.06. In 1999, Netscape issued a document describing the next version of the language, 1.4, but this has not yet been implemented.

At about the same time that the ECMA standard was published, Microsoft issued its own implementation of the language. This was called JScript and was given the version number 3.0. Unlike the then current Netscape offering, this fully conformed to the standard. With the release of IE 5, Microsoft upgraded JScript to version 5.0. However, Microsoft is also continuing to support the Netscape implementation: IE 5 supports all the features of JavaScript 1.2.

One other JavaScript-aware vendor is Opera Software. It has offered sup-

port for JavaScript 1.1 since version 3.21 of its Opera browser.

ECMA has been preparing a second edition of the standard. Generally known as ECMAScript 2, this was designed to catch up with additional features which the two main vendors have added to their implementations (including regular expressions, new control constructs, better string handling and new exception-handling features). This new edition will be the third edition of the standard, as an update was published in August 1998; however, this contained editorial changes only.

### **JavaScript Vs VBScript**

JavaScript's main rival is Microsoft's Visual Basic Scripting Edition, otherwise known as VBScript. For some developers, VBScript will be a more attractive choice. It has the advantage of being a subset of Visual Basic and is therefore already widely known. It is usually regarded as being easy to learn. By contrast, JavaScript more closely resembles C and C++ in its syntax, which is something of a turn-off for many programmers.

The disadvantage of VBScript is that it is only supported by Microsoft browsers. This might not be a problem if you are developing Web pages for an in-house intranet and you can be sure that all your users have Internet Explorer. But if you are publishing on the Web, and you are anxious not to lock out a substantial part of your audience, you will have no choice but to opt for JavaScript. Your pages will then be equally accessible to users of IE, Navigator and Opera.

The choice between the two scripting languages does not have to be mutually exclusive. It is perfectly possible to use them alongside each other, even within the same HTML page.

### **JavaScript And Java**

Despite the similarity of names, JavaScript is not related to Java, either technically or in terms of ownership. It's true that both languages are used to enhance Web pages, and there are several ways in which they can interface with each other, but in all other

respects they are quite separate.

In particular, JavaScript is in no way a lightweight version of Java, as is sometimes supposed. In fact, it is not a lightweight anything. Despite being relatively easy to learn, it is a powerful language in its own right, and it can be used to create quite sophisticated applications.

That said, Java is much more of a "pure" language. Unlike JavaScript, it supports static typing (variables must be declared and typed in advance) and strong type checking, and it has a class-based object model with full inheritance. Also, it is a compiled language; it compiles to intermediate bytecode which is interpreted in the browser.

JavaScript, by contrast, is fully interpreted. As a result, it is inherently slower than Java, and its source code is less secure: anyone can view and amend JavaScript code in an ordinary text editor. Another difference is that JavaScript is generally only used within Web pages. Java, on the other hand, can be used to create applets which run inside the browser but which are substantially independent of the Web page. You can also use it to write standalone applications.

### **Client Or Server**

Although more often used within a Web browser, it is also possible to run JavaScript on a Web server. In client-side JavaScript, the code is embedded in an HTML page. The browser reads the page and interprets each element of JavaScript code as it encounters it, just as it does with HTML.

In server-side JavaScript, the code is also embedded in an HTML page, but it is pre-processed by a run-time engine on the server. Exactly how this works depends on whether the server is from Netscape or Microsoft. In general, the engine executes the code in response to a request from a browser. The code typically generates a new HTML page, and this is sent back to the browser, which renders it in the usual way.

An interesting point about server-side JavaScript is that it can be run from a browser which does not itself support JavaScript. That's because the server simply returns standard HTML

for the browser to render. That said, it is open to the server-side code to embed client-side JavaScript in the returned page; in this case, the browser would need to be JavaScript-compliant.

People sometimes talk about client-side JavaScript and server-side JavaScript as if they were two distinct languages, but this is not the case. Although certain language elements are specific to one "side" or the other, essentially the syntax is common to both. For this article, I will concentrate on client-side JavaScript; I will say more about server-side scripting towards the end.

### **Security**

One thing which client-side JavaScript cannot do is to access the resources of the user's computer - except in a very limited and controlled way. In particular, JavaScript cannot arbitrarily read, update or delete files on the user's hard disk, and these restrictions enhance its security.

However, JavaScript is able to control the operation of ActiveX components. These in turn are capable of performing potentially harmful operations such as reading and writing files. Users who are concerned about security might want to consider disabling ActiveX support in their browsers, or limiting it to trusted sources.

*This article will be concluded in a future issue of PCSA.*

**PCSA**

*Copyright ITP, 2000*

### **The Author**

Mike Lewis is a freelance IT writer and can be contacted as [mike.lewis@itp-journals.com](mailto:mike.lewis@itp-journals.com).

## Recent Reviews from [Tech Support Alert](#)

### [Reviews of the Best Windows Backup Software](#)

In this detailed comparative review, we checked out eighteen backup software utilities designed for home or SOHO use. Many of the products reviewed were disappointing. However 6 products passed our tests with flying colors and 2 of these were so impressive, they were awarded our "Editor's Choice."

### [Suppliers of Cheap Inkjet Printer Cartridges Reviewed and Rated](#)

With hundreds of companies all claiming to have the "*cheapest and best inkjet printer cartridges*," our editors decided to put their claims to the test. Not unexpectedly, many suppliers flunked but we did manage to come up with a number of web sites that sell good quality inkjet printer cartridges at heavily discounted prices.

### [The Best Anti Trojan Software](#)

Our editors took a close look at the 6 leading anti-trojan/trojan remover software utilities. Unfortunately, they found only 2 products that were effective in their ability to detect and remove dangerous modern polymorphic and process injecting trojans.

### [The 46 Best Ever Freeware Utilities](#)

This is our Editor, Ian "Gizmo" Richards, personal selection of the best freeware utilities. He's hunted down some real gems, many of which perform better than expensive commercial products.