

ECMA?

Article published at:

http://www.morebusiness.com/running_your_business/technology/d923325289.brc

I'm sorry, can you take something for it? I know a good pharmacy

Not exactly. When people talk about ECMA they aren't discussing a personal hygiene problem or skin disease, but a scripting language standard. Brad Edwards, writing for CMPNet [<http://www.wcmh.com/ltparts/discuss/discuss24.htm>], says ECMAScript, also known as ECMA 262, is "the standard JavaScript, but." He notes that whereas ECMA formerly stood for "European Computer Manufacturers Association," it now stands for "absolutely nothing."

Okay, what's scripting again?

According to the official ECMA standards document itself (to be found by the hardy at [<ftp://ftp.ecma.ch/ecma-st/e262-pdf.pdf>] on the official "ECMA-262 ECMAScript Language Specification" page at [<http://www.ecma.ch/stand/ecma-262.htm>]) "a scripting language is a programming language that is used to manipulate, automate and customize the facilities of an existing system. In such a system, useful functionality is already available through a user interface, and the scripting language is a mechanism for exposing that functionality to program control."

David Boroditsky of Emergence by Design [<http://www.DTPScripting.com/WhatIsScripting.html>] has explained that "a script is basically a program. There are many varieties of scripting languages," and they differ from full-fledged programming languages by "not being compiled into processor native code, being faster to develop in, being easier to learn and not being as full-featured." Application programmers can make their applications scriptable, he says, "allowing scripters to automate program functions. Scriptable applications allow scripters to directly manipulate a program and its data files, unlike macros, which merely mimic user actions."

Boston area-based Nombas Inc., a provider of multi- and cross-platform scripting tools and technology, says that scripting is "now recognized as the fastest, simplest and most-flexible method to create IT programming solutions." It's your answer when "system programming languages are too complicated and time-consuming for most IT tasks and users," and "static tools such as markup languages, compiled applications, pure-visual or table-based development are too weak and inflexible for your tasks." Scripting covers the large span between system programming and static tools, "the span where solutions are made quickly and by a large class of computer users -- "you don't have to be a programming expert to script," they say, although "if you are a programming expert then you'll be a lot more productive with scripting."

Scripting lets you link components. ActiveX, JavaBeans, applications -- be they thin, fat, new or legacy -- libraries, databases, HTML and servers -- allowing off-the-shelf applications to be customized, or automating workflow procedures.

An excellent technical overview of the ECMA scripting language itself is to be found at NetscapeWorld's Guide to ECMA Script, at [<http://www.netscapeworld.com/nw-07-1997/nw-07-javascript.html#comp>].

I see. So, exactly how is scripting better than systems languages?

According to Nombas [<http://www.nombas.com/us/scriptaz/advntage.htm>], there are a several reasons:

- Faster development. Scripts take much less time to write than system code. The programmer must do less housecleaning as the script platform handles most of the tedious memory and data problems.
- Safer. The script environment protects against programming bugs. At the Web site listed above, Nombas gives an example of a divide-by-zero situation "where dividing by zero would crash most systems, but is caught by the scripting system."
- Greater security. All calls in a script environment must pass through the script environment, which can then impose any level of security on the code. Most script languages offer a security model with a huge variety of security options.
- Fewer bugs. Easier to debug. Run-time evaluation and system independence make debugger implementation easy.
- Smaller code. A single line of scripting code often accomplishes many lines of system-level code.
- Faster to market, smaller, cheaper. R&D nirvana, at last.

As a result of these reasons, scripting languages have gotten quite popular as of late. There's one more reason: the Internet. John K. Ousterhout, who has written an exhaustive paper, "Scripting: Higher Level Programming for the 21st Century," [<http://www.scriptics.com/people/john.ousterhout/scripting.html>] says, "The growth of the Internet

has also popularized scripting languages. The Internet is nothing more than a gluing tool. It doesn't create any new computations or data; it simply makes a huge number of existing things easily accessible." Therefore, "the ideal language for most Internet programming tasks is one that makes it possible for all the connected components to work together, i.e., a scripting language."

Right. So back to ECMAScript.

Back to ECMAScript. Shelley Powers has written a fine, detailed and convincing proof of how incompatibilities still exist in the Microsoft and Netscape versions of supposedly "standardized" products [<http://www.ne-dev.com/netscapeworld/nw-08-1997/nw-08-crossbrowser.html>], and discusses ECMAScript as one battle in their ongoing standards turf war.

"Adhering to standards is as much a matter of marketing as it is a design philosophy," Powers observes. She gives the example of Netscape's setting the "de facto standard for scripting Web pages" with JavaScript. Yet Microsoft began to challenge Netscape by gaining popularity for its browser, which uses JavaScript in addition to its own version of JavaScript, called JScript, and VBScript. So, she explains, "Netscape takes the approach to get its language defined as the formalized standard as well as the de facto standard."

Now Netscape could approach the International Standards Organization as the organization to initiate this process, but "ISO is not known for speed," Powers says. So "Netscape approached ECMA, a European-based standards organization primarily associated with media interface formats. Soon after, Microsoft joined the ECMA membership. Both pronounce their promise to adhere to whatever standard is produced by this organization. ECMA didn't know what hit it."

Yet the result of "this flurry of big corporate support is ECMA-262, also known as ECMAScript," Powers says. "Released in July 1997, ECMAScript is basically JavaScript 1.1 with the addition of some new or changed data types and support for Unicode." As Powers says wryly, "That's one standard down." Skirmish goes to Netscape, but the war's still far from over.

So you're saying ECMA doesn't solve the problem it's supposed to solve.

Writing shortly after ECMAScript's introduction in June 1997, Edwards says, "ECMA announced the adoption of a standard Internet scripting language, ECMA-262, also known as ECMAScript until ECMA members can settle on a final name."

"ECMAScript was originally designed to be a Web scripting language," says the official standards paper, published in mid-1998, "providing a mechanism to enliven Web pages in browsers and to perform server computation as part of a Web-based client-server architecture."

The idea was to have one scripting language that would work on Microsoft, Netscape or what-have-you. Upon its introduction, that appeared to happen. Microsoft and Netscape "and other industry leaders," Edwards says, "welcomed ECMAScript, which is based on Netscape's JavaScript 1.1." In fact, Microsoft announced full support for ECMAScript in its JScript 3.0. Everyone, developers and end users, were happy: This meant that developers, tired of "small incompatibilities between JavaScript implementations," Edwards says, "could write code that they knew would run in any ECMA-262 compliant browser."

However, they soon found that just because a script runs in one ECMAScript implementation does not mean it will run in another. Edwards explains this is because "ECMA-262 is a specification for minimum compatibility; companies are free to add extra features beyond those found in ECMA-262." He adds that "you can expect Microsoft, Netscape and other companies to take full advantage of this to add value to, and woo you over to, their products."

He gloomily predicts that when persuasion fails, force will be used: "First, you'll find it difficult to tell which features are part of ECMAScript and which are browser-specific additions. If they can get you adding features that work with only their browser, then they've scored a point in the browser war." From there it's but a short step to creating scripts that don't run with other browsers in an effort to develop THE standard, which they can then of course control to their advantage. So the Web page developers and end users stuck with yet another incompatible scripting language.

Will they ever just give us one really standard language everyone uses?

Someday, Grasshopper, someday.