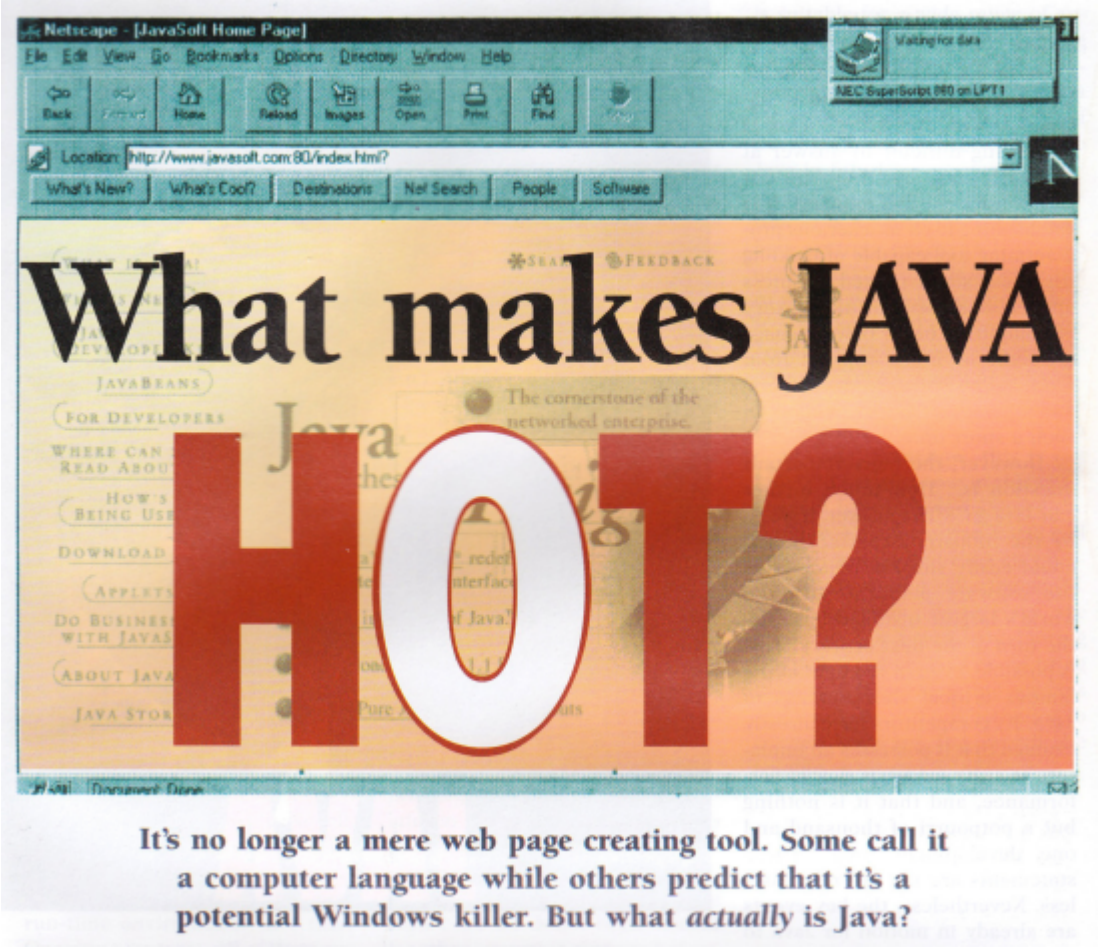
What makes JAVA Hot?

Posted on



If you have ever seen the bouncing heads of cartoon characters that roam in the Internet Web sites, and letters that behave weirdly and run everywhere on the screen-remember, that it is Java behind their creations.

Those who have surfed the net for sometime may remember that in the good old days, the web for all its multimedia marvels, which included text, images and even sound and video clips, was a fairly static medium. A computer user connected to the Internet could visit web sites, but doing so was similar to watching the world through a store front window. Java

essentially removed the glass from the window. It served as a universal translator,

enabling programs to move fluidly between incompatible operating systems. With the help of a browser like Netscape, it ran across the Web and the rest of the Internet to fetch programs from the powerful computers that run web sites and brought the programs and the information they generated back to a user's home computer.

That is history. Java has come a long way since those days. Now, nobody calls it just a cute, web page creating tool. Some call it a computer language or an application software development tool.

Others are eager to correct that and call it an operating system. To a certain extent both these parties are correct. It is impossible to classify Java into any one category of Software. Reason-it is one of its own. Before Java or so far after it, there was no software in the market that can be put in the same group with Java.

Developed for over four years by an elite team of software designers at Sun Microsystems, led by James Gosling, and introduced to the market in the beginning of 1995, Java burst into the already rapidly expanding commercial world of the Internet and the World Wide Web in a way no software marketeer had thought possible.

Is Java a serious programming language? Of course. Will it be come a Windows killer? This is something difficult to answer at this early stage but there are fair possibilities. At this moment, it is establishing itself as a serious programming tool capable of tackling the most sophisticated business applications. Never in the history of computing has a new language attracted so much support from toolmakers, software developers, and Operating System Vendors in such a short time.

However, the more important question is, "How much further can Java go"? The answer, if we go by the industry experts, then it could easily surpass Windows as the software platform with the world's largest installed base, by the turn of the next century. This is possible, even in the light of the famous critical shortcomings of Java. Its competitors have already claimed that it possesses an immature security model, is slow in performance, and that it is nothing but a potpourri of thousand and one development tools. These statements are not entirely baseless. Nevertheless, the key events are already in motion for Java to overtake Windows and cause massive changes in the commercial and corporate software development. The bottom line is clear-all those who have anything to do with software development

can ignore Java at their own peril.

At present, many commercial developers write their software for Windows first, because it is the most popular operating system in the world. Windows runs on about 90% of the world's Personal Computers. Then they either convert their programs to the other lesser used platforms (e.g., Mac, OS/2, and Unix) or simply ignore them. If Java is going to command a larger installed base than Windows in the near future, developers may write for Java and only for it because Java software is inherently cross-platform and can run on any system with any operating system, provided that it has a Java emulation mode. In effect, Windows too would become another minority platform. Java, therefore, is going to cause the largest revolution in the known history of the operating systems, which will most certainly result in a louder bang than the one we heard at the time of DOS to Windows conversion.

Still, before Java can become a dominant platform, corporate and commercial software developers have to embrace it as a language.

This was seen as a distant dream even as recently as end of 1995, when many of the programmers were asking if Java was powerful enough for serious software development. Today, we find that question has become obsolete. In October 1996, Sun Microsystems estimated that more than 200,000 professional programmers were using Java.

Cross-platform compatibility was a tremendous factor in favour of Java's early success. Java compilers (available for Windows, the Mac OS, and Unix) convert Java source code into class files of byte code. The class files correspond to the executable binary files generated by compilers for other languages. Unlike native binary files, Java byte code isn't specific to a particular microprocessor architecture. Its 'native' architecture is the Java, which today exists only. in software. As a result, Java class files are portable to any hardware platform that has a Java run-time environment. The environment consists of the Java, some standard class libraries, a byte-code verifier (for security), and a byte- code interpreter. The interpreter runs the class files on a Virtual Machine without requiring programmers to rewrite or even recompile their source code.

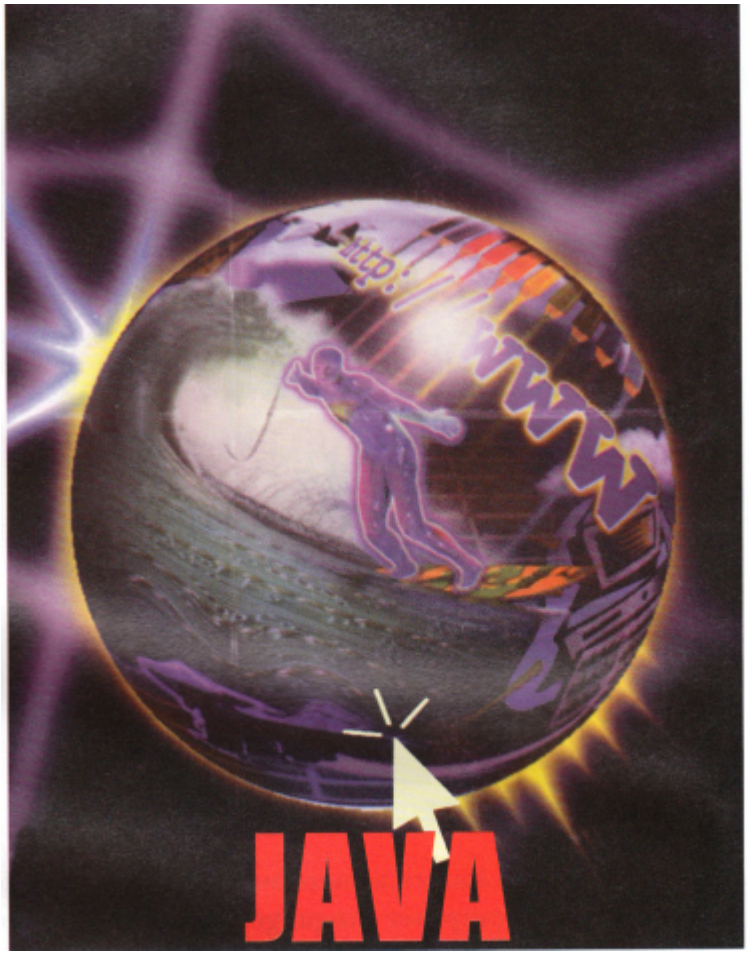
Java is also a platform implemented in software that runs on practically any machine, and software spreads much faster than hardware. If you've installed a Java-compatible Web browser, such as Sun's HotJava, Netscape Navigator 3.0, or Microsoft Internet Explorer 3.0, the Java run-time environment is already on your

computer. Java isn’t self- replicating like a virus, but it’s nearly as contagious.

Apple, IBM, Microsoft, Novell, Silicon Graphics, and Sun are paradoxically accelerating the process by integrating the Java run-time environment into their Operating Systems. All of them say their Operating Systems will be Java-enabled within a year. They all recognized Java’s popularity and potential, and realised that offering a superior Java run-time environment will give them a competitive advantage.

But there are also bottlenecks. The most important trend is towards higher levels of software abstraction above the hardware. The more tightly a software is intertwined with the hardware, the bigger the headaches for developers and users. Programmers get more performance by writing to the metal, but the code is hard to maintain and even harder to convert. And code lives longer than anyone plans. (This is one of the reasons why we are going to face the famous year 2000 problem in another 3 years time.)

Developers who need maximum performance have at least one good reason for avoiding Java-it is slow. You can write a compiled program in C or C++ that runs easily 10 times faster than an interpreted Java program. On the other hand, there are also many applications, for which this isn’t important. Tools such as Visual Basic and Power Builder have become popular mainly because they’re fast enough. But when speed of the performance comes under the spot light, there’s no denying that interpreted Java byte code gets beaten by all the rest.



In the long run, it is difficult to think that any of these technical problems that deter today’s developers are likely to pose an insurmountable obstacle for Java, since both as a language and as a plat- form, Java is evolving at an unprecedented pace. We can speclate on Java’s course because it’s consistent with historical trends in computing.

In all fairness, some features of Java have no doubt been offered by other operating systems before, but not all of them together. Unix and NT no doubt offered some hardware abstraction, but they’re multi-platform not cross-platform as Java. Users of Unix and NT still have to replace or recompile all their software if they switch micro- processors, since not all software is compatible with all microprocessors. Also, these Operating Systems still chain you to an Operating System. Java is going to set you free.

From a developer’s point of view, the decision about adopting Java will depend on

one main question can it handle the jobs? It should be clear by now that Java is suitable for a wide range of applications and is fast gaining ground. Though it can't do everything, and the tools need to get a lot better, in all probability a lot of developers are going to turn to Java in future.

Finally, the inevitable big question is it worth placing all your bets entirely on Java? If you do not like taking risks it is better not to put all your eggs in one basket. It is a popular cliché among programmers that those who gamble correctly on an emerging software will be rewarded well, and those who gamble wrong end up with a dead code. Somehow, this is not hundred percent true in this case. Even if Java fails to conquer the world as a platform, you'll still end up with a code that runs on whatever platform that rules the world. For developers, these risks are minimal. For users, considering the new freedom Java could bring to change their operating systems and hardware without bothering to convert software, it is worth the trouble.

Chanuka Wattegama is an Electronics Engineer by profession. He is a regular contributor to the business and IT pages of local newspapers and he received the award for the Best Science Writer in the island in 1988.