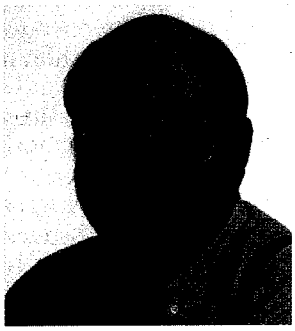
The Year of the Net Wars



BY ROGER SESSIONS

You didn't watch the Superbowl. You slept through the Olympics. You forgot about the presidential debates. But there is one melee nobody could have missed. 1996 was the year of the Net Wars.

To be sure, this war isn't settled and probably won't be for at least another year. However, the opening skirmishes were mean, and the first victims have already fallen.

As usual, we have Microsoft on one side. As usual, we have most of the rest of the world on the other. As usual, IBM is not entirely sure what the game is about, but they heard it's going to be fun and don't want to be left out.

It's tempting to dismiss the Net Wars as a Microsoft ploy to take over the world, but the truth is more complex. This war is really one of two competing visions of what the Internet is all about. Both visions are compelling. Both have serious weaknesses.

The Microsoft opposition is led by Netscape and SunSoft. Netscape produces the highly successful Web browser, Netscape Navigator. SunSoft produces Java, a new programming language that has taken at least the book publishing industry by unprecedented storm.

In the Microsoft vision, the Net is a logical extension of the operating system. Because everybody will be using the same operating system (guess which), the rest of the issues are unimportant. A Web browser is a relatively uninteresting application, one of many used to find documents and start up programs that live someplace on the Net.

In the Netscape/SunSoft vision, the operating system is merely a support mechanism for the Web browser. Since everyone will be using the same Web browser (guess which), the underlying operating system really doesn't matter. Programs will be written especially for the Web browser and will therefore run on any machine on which the Web browser runs.

IBM's strategy is simple: The enemy of my enemy is my friend. Therefore, whatever Netscape and SunSoft decide is fine with IBM.

Three main fronts exist on which this war is being fought. They are all interrelated, and it is difficult to look at any one in isolation. These fronts are:

- programming language
- scripting language
- Web browser.

Programming language

In this arena, it's Java, representing SunSoft/Netscape, versus ActiveX Controls, representing Microsoft.

Java is a new programming language. It is object-oriented, compact, and interpreted. It is derived from C++, but much easier to use.

Because Java is interpreted, a Java program can run on any machine that supports the Java interpreter. And because the Java interpreter is highly portable, it is available for just about every machine type. Java programs do not have to be recompiled; they are simply downloaded and run. There is absolutely no difference between a Java program for UNIX and a Java program for OS/2.

Java's "download and run" capability is an essential part of the SunSoft/Netscape "who cares about the operating system" philosophy. It is also an attractive concept to any software vendor that would like to support a single product that can be run on any machine.

There are two basic categories of Java interpreters. One category is an interpreter designed for running stand-alone Java applications. The other interpreter is designed for running mini-applications under the control of a Web browser. These Web-based mini-applications are called *applets*. The programming model for applets is slightly different from the one for applications, but with care, a programmer can write Java code that will run both as an applet or as an application.

Microsoft doesn't think developers want to start writing in a new language. They believe there are plenty of applications already written in C and C++, the standard languages for most software developers.

What Netscape/SunSoft call an applet, Microsoft calls a component. Microsoft believes that it already has a widely accepted model for developing components, and that model is OLE (Object Linking and Embedding). OLE describes how an application can be written as a component. Components have well-defined mechanisms for interacting with each other, including the ability to run as an embedded component within another component, such as a Web browser. OLE components are typically written in C++ and must support specific methods defined by OLE.

OLE doesn't support linking across the Internet. Microsoft has added extensions to include this support. An OLE component that supports these extensions is said to be an ActiveX Control. Microsoft believes that any

serious application is already written as an OLE component, and with minimal modification, can be turned into an ActiveX Control. Why, it asks, would anyone want to totally rewrite their application in a new, unproven language?

Of course, ActiveX components, unlike Java applets, are going to be operating-system-specific. You can't take code compiled for Windows and run it on a Macintosh. But Microsoft doesn't see this as an issue. Windows is the only real operating system, it reasons, so why would anyone want to run code anywhere else?

So the language front boils down to these issues.

ActiveX provides a logical upgrade path from OLE and offers the best support for traditional programming languages, such as C++. Any number of applications can serve as a host to an ActiveX Control. However, these applications can only be run on one operating system.

Java is a new language that allows the development of machine-independent applications. It is designed to be simple. It does away with many of the complexities of C++. A Java applet can be run on any number of operating systems. However, existing apps must be rewritten in a new language.

Scripting language

HTML (Hyper-Text Markup Language) is a well-accepted standard for describing the layout of documents. When you download a page to your Web browser, you are really downloading an HTML file. This file is then interpreted and displayed as a Web page by your browser. HTML is supported by both Netscape Navigator and Microsoft Explorer.

Web page designers have long wished to extend HTML with some rudimentary programming capabilities. These capabilities would allow processing that is now done on the server to be moved to the client, where it can be done much more efficiently.

Both Microsoft and Netscape/SunSoft have proposed HTML extensions called *Scripting Languages*. Scripting languages are mini-programs embedded directly into an HTML file. Needless to say, Microsoft and Netscape/SunSoft do not agree on what this scripting language should look like.

There is one little-noticed technical area in which Microsoft and Netscape/SunSoft have gone in different directions.

Microsoft thinks that any scripting language should be based on some well-established script-like technology. By an incredible coincidence, Microsoft happens to have just such a technology available. It is called Visual Basic. The scripting language it proposes is VBScript, for Visual Basic Script, a straightforward extension of Visual Basic. This path, Microsoft argues, allows the many developers familiar with Visual Basic to make an easy switch to HTML programming.

Of course, in the Netscape/SunSoft view, real programmers don't use Basic. Nobody, they argue, will want to use a derivative of—snicker, snicker—Basic when we have a modern programming language like Java available. Therefore, the scripting language should be one derived from Java: JavaScript.

Many of the arguments about JavaScript versus VBScript are old dogs in new costumes. Are object-oriented languages better that procedural languages? Which are easier to learn? Which are easier to maintain? In which style are most existing programs already written? You get the picture.

Web browser

The most visible front of the Net Wars is the battle over the Web browsers. Microsoft offers Internet Explorer. SunSoft/Netscape offer Netscape Navigator. Much ink has been spilled comparing these two. Which is faster? Which is better integrated with other Web tools? Which is more expensive? In reality, none of these issues matter much. Most articles are missing the basic point of the war.

In reality, Microsoft doesn't care about selling Explorer. It cares about getting rid of Netscape and eliminating its dangerous vision that the Web browser, rather than the operating system, is the top of the GUI heap. The Microsoft plan is to have Explorer beat Netscape, and then morph Explorer into the operating system.

Netscape/SunSoft has an initial lead in the browser field. They are using a three-prong strategy:

- get Navigator out to as many sites as possible, to as many machines as possible
- take advantage of the widespread interest in Java as a development language for creating applets
- encourage as many people as possible to develop applets.

Microsoft is running a close second, much too close for Netscape/SunSoft's comfort. Microsoft is using a four-prong strategy to win on the browser front:

- give Explorer away free
- support all of the capabilities of Navigator including Java applets and JavaScript
- extend these capabilities with ActiveX components and VBScript
- encourage Web sites to take advantage of the Explorer-specific capabilities.

There is one little-noticed technical area in which Microsoft and Netscape/SunSoft have gone in different directions. This area is one that is likely to have a major impact on the evolution of the Net, and yet is hardly ever discussed. This area is security.

Security is a significant issue when people are downloading software from the Net. How can one be sure that the software being downloaded is trustworthy?

Netscape/SunSoft build their security into Java and the Java interpreter. They've removed features

from the language that historically have been a favorite tool of hackers—these features, say Netscape/SunSoft, are unnecessary anyway due to the simplicity of Java. They have also restricted what a Java applet is allowed to do while running under Navigator.

The most significant restriction enforced on Java applets by Navigator is the ability to read and write local files. Netscape/SunSoft rightly believe that if arbitrary applets can be downloaded from who knows where and allowed to read and write local files, then the possibility for malicious netplay is unlimited.

On the other hand, this restriction makes it almost impossible to write a serious application. How many real applications do not need to store data in files?

Microsoft has made a different choice. In its security model, every ActiveX control is assigned a unique, secure ID code. Given a control, one can determine its ID code. This code cannot be forged. The system administrator registers the ID codes of trusted components within the operating system.

When a component is about to be downloaded from the Net, its ID code is checked against those the operating systems trusts. Only trusted components are allowed to be downloaded. Once you download a trusted component, you allow it to do whatever it wants, include read and write files. After all, you trust it.

The Microsoft security mechanism allows much more robust controls to be downloaded. Controls can do all of the things we normally associate with OLE components and real applications. However, a system administrator must decide in advance that a particular control is trustworthy.

The Netscape/SunSoft security mechanism allows Java applets to be downloaded from anyplace. The person writing a Web page can write any Java applets the page needs, and ensure they are downloaded as needed. The system administrator doesn't have to decided in advance which applets will be allowed to run. The browsee doesn't have to worry about what the applets will do, because Navigator won't allow them to do anything dangerous. However, the applets will be very limited—one might even say, crippled, in functionality.

The wild card

There is one wild card in this whole game that is worth considering. This card is thrown in by Oracle Corp., a company better known for its relational database products.

Oracle envisions a network-specific computer, nicknamed NC (Network Computer). This computer will be extremely cheap and very limited in functionality. In fact, about all it will be able to do is run a built-in Net browser. It won't have to worry about reading and writing to local disks, because it won't have any. Any data it needs will be downloaded across the Net.

This vision is highly compatible with the Netscape/SunSoft view of the world. For this machine, the browser *is* the operating system. Nothing could please Netscape/SunSoft more.

Oracle believes NCs, at a cost of a few hundred dollars, will become as ubiquitous as telephones. If this vision becomes reality, Netscape/SunSoft will have gained a powerful ally.

Final thoughts

No one knows how the Web will evolve over the coming year. Will it become a seamless extension of the operating system, or will it replace the operating system? Will it be used for running large applications or small applets? Will the new world of applications be written as minor extensions of OLE, or will they be written in a brand new programming language that barely existed a year ago?

All we know for sure is that we live in interesting times. The best thing to do is to sit back and dodge the bullets.

Books on these topics

The Java Programming Language by Arnold and Gosling (Addison-Wesley). A good introduction to the basic Java language. The problem with the book is a lack of information on the AWT (GUI) toolkit. Recommended.

The Java Tutorial by Campione and Walrath (Addison-Wesley). This book was originally written as a hypertext-linked HTML document. A lot of good information is here, but the attempt to maintain its hypertexted lineage is very annoying. It is also highly biased towards applet rather than application development. Not recommended.

Core Java by Cornell and Horstmann (SunSoft Press/Prentice Hall). A good, comprehensive book, with especially effective coverage of the AWT (GUI) toolkit. Biased towards applications rather than applets. Recommended.

Java in a Nutshell by Flanagan (O'Reilly and Associates). This small reference book is indispensable. You will use it many times each day to find some obscure method and look up some class hierarchy. Not good for learning Java, but great once you know it. This book never leaves my knapsack. Highly recommended.

The Java Application Programming Interface, Volumes 1 and 2, by Gosling and Yellin (Addison Wesley). Similar to *Java in a Nutshell* as far as library coverage, but much more extensive. Good for your bookshelf. Recommended.

The Java Language Specification by Gosling, Joy, and Steele (Addison-Wesley). This book contains the formal definition of the language. A necessity for people implementing the Java language and others who have absolutely no real life. Otherwise, not recommended.

JavaScript by Flanagan (O'Reilly and Associates). A good overview and reference book for JavaScript. This book is about the beta version, so things will change. Recommended.

Presenting ActiveX by Ernst (Sams Net). A high-level overview of the Microsoft strategy with ActiveX. Not much information for the hard-core techie, but there isn't much out about ActiveX. Recommended. OS/2

Roger Sessions teaches and consults on object technology. His books include the recently published Object Persistence—Beyond Object-Oriented Databases. *Roger also publishes the OS/2 Objects Home page (http:/www.fc.net/~roger/owatch.htm). He can be contacted at roger@fc.net.*