Choosing Your Web Design Tool

BY MICHAEL TEMPLEMAN

Here's a guide to the pros and cons of today's Internet publishing tools.

"Complexity is Conserved."—attributed to Larry Tesler
"TANSTAAFL (There Ain't No Such Thing As A Free Lunch)."—The Moon is a Harsh Mistress, *Robert A. Heinlein*

As a programmer you've probably found that in many cases, the easier or safer a development or design tool is, the less powerful it is. For example, doing a print layout with Adobe PageMaker is infinitely easier than creating the same layout directly by programming with Postscript. On the other hand, you can do many things directly in Postscript that are either hard or impossible to do in PageMaker. For every 10,000 people that can make an attractive layout in PageMaker, you'll probably find only one person able to do the same in Postscript.

You face similar tradeoffs when you evaluate tools for creating World Wide Web pages. Easy HTML development won't exploit the full power of HTML. Conversely, hand-coding HTML allows you to use the expressive power of the language much more readily than using a high-level tool such as Microsoft's Internet Assistant or Adobe's PageMill. And when frame-based layout arrives on the Web—through whatever mechanism—it will be enormously simpler to use a tool such as Microsoft's Internet Studio or PageMill than to hand-code it from the underlying language. When you factor in the ability to script using Visual Basic Script (VB Script) or JavaScript and extend the basic HTML elements through OCXs and plug-ins, then things will really go over the top.

The same will be true of security. Tight security, which appears to exist in Netscape Java, will go far to prevent malicious or incompetent applets from trashing your computer. On the other hand, security this tight prevents extremely useful actions and can dramatically limit functionality of applet-enhanced pages. While Big Nanny will prevent you from getting into too much trouble, she also won't let you ski, skydive, or rock climb.

But if you loosen up the security constraints, users and system administrators suddenly have to worry about certification, validity, and sources. Is that really an Amber plug-in from Adobe, or a malicious piece of code that will issue an FDISK on my hard drive?

In the meantime, here's a look at the most prominent members in the spectrum of tools for Web design:

• ***Java.*** While you can use Java to create standalone programs, at this point you're probably more interested in how to use Java as a programming language for the

Michael Templeman is president of MetaBridge Inc., a Seattle, Washington-based company that provides design and engineering services to online publishers. Contact Michael on the Internet at miket@metabridge.com.

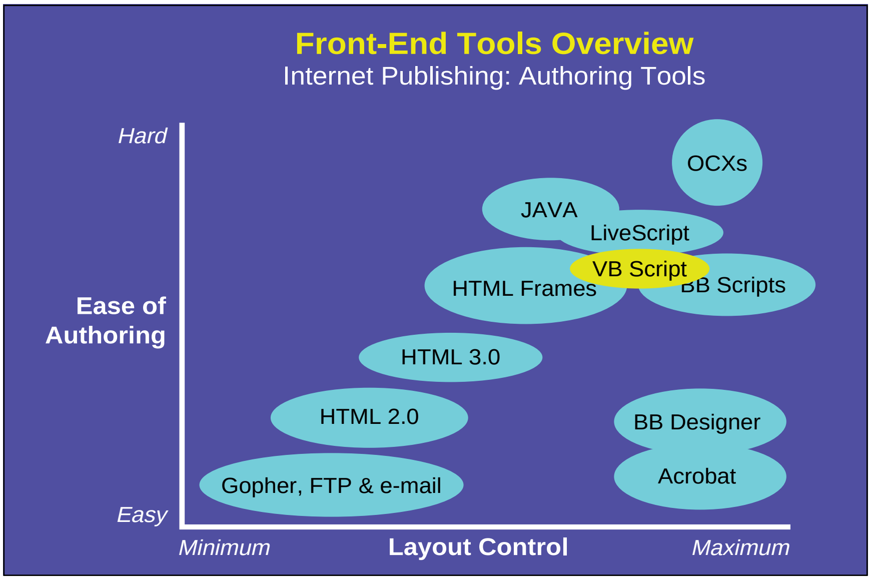

FIGURE 1 ***Spectrum of Web Development Tools.*** *This graph shows the relative tradeoffs in power versus complexity of the many tools you can use to create Web applications. Early HTML pages were extremely easy to create, which was central to their success. But soon every page started to look alike and the demand for more formatting and more dynamic pages escalated complexity. HTML 3.0 provides frames and tables. Because scripting languages embed code with HTML, you can quickly create small custom applications with JavaScript and VB Script. OCXs are difficult to write but provide much more power.*

Web. Using Java, programmers can build "applets," small pieces of code that run within an HTML page. Java applets are partially compiled into a bytecode and cannot be disassembled into the source.

These applets are roughly comparable to OCXs in terms of general functionality. Netscape characterizes Java applets as mini-applications: "When integrated into Web pages, Java applets allow expert graphics rendering, real-time interaction with users, live information updating, and instant interaction with servers over the network. Java applets are downloadable from any server and run safely on any platform. Java is designed to provide maximum security on public networks, with multiple safeguards against viruses, tampering, and other threats."

- ***JavaScript.*** JavaScript, on the other hand, is a simple language you can embed within HTML pages and use to glue HTML components together. JavaScript is roughly comparable to Microsoft's VB Script. This is how Netscape characterizes JavaScript: "Based on the Java language, JavaScript extends the programmatic capabilities of Netscape Navigator to a wide range of authors and is easy enough for anyone who can compose HTML. Use JavaScript to glue HTML, inline plug-ins, and Java applets to each other."
- ***HTML/HTTP.*** HTML is the language of the Web. HTTP is the language to communicate with Web servers. HTML began as a markup language with little formatting control, but is quickly morphing (or metastasizing) to include layout, formatting, and embedding of components. These added features as well as future features will make working with HTML pages really chaotic and complex. Enhancements to HTTP will have little effect upon page design, though webmasters should be aware of features such as keep-alive or byte-range downloading.
- ***Blackbird, PageMill, FrontPage, and Internet Assistant.*** These are tools that produce or will produce HTML and, in some cases, assist with the management of the server. These tools are easier to use and much cheaper than hand-coded HTML, but sometimes they cannot completely exploit HTML. Still, if you can't hand-code in HTML right now, it is hard to know what you are missing. Microsoft has delayed Blackbird's release as Internet Studio in order to change it from a proprietary format to an entirely HTML-based format (see Interactive Developer, "Microsoft's Blackbird Uses OLE to Compete with Java, Netscape," *VBPJ* December 1995). For more information, visit www.adobe.com/Apps/PageMill/.
- ***OLE/OCX.*** OCXs can communicate with one another through OLE, support several kinds of persistence, expose a standard object model, deliver specific platform features, and be used in a wide range of containers. Microsoft claims that Mac OCXs will be possible in the future, created from the Macintosh cross-compiler of Visual C++.

Microsoft is developing new interfaces and functionality for OCXs specifically targeted at the Internet, also called COM objects for the Internet. For more information on Internet OCXs, visit http://198.105.232.6/intdev/inttech/olecon.htm. This is part of the complete Microsoft package of tools, APIs, and technology code-named "Sweeper."

JAVA APPLETS ARE ROUGHLY COMPARABLE TO OCXs IN TERMS OF GENERAL FUNCTIONALITY.

- ***Inline plug-ins.*** Netscape Navigator supports inline plug-ins to support new data types and additional features. Plug-ins are comparable to OCXs, but they have a limited object model and no type library support as OCXs do, which limits the ability to use plug-ins as components. Like OCXs, plug-ins are compiled for specific platforms—Windows, Mac, or Unix—and are inherently insecure. In other words, you'd better know who provided your plug-in before you plug it in!

To find out more about plug-ins, visit http://home.netscape.com/comprod/products/navigator/version_2.0/plugins/index.html and check out some of the cool things that have been done. Two great plug-ins to experiment with are Adobe's Amber (http://www.adobe.com/Amber/) and Macromedia's Shockwave (http://www.macromedia.com/).

ADVANTAGES AND DRAWBACKS

- ***Java.*** Java works very well over the World Wide Web. Its built-in classes allow developers to easily access URLs over the Web. Because Java is a byte-code interpreted language, it is readily portable to most popular computer platforms. Its object-oriented model, built-in multithreading support, security, and C++ syntax make it a strong contender for use in Web pages that need the extra functionality or "zing" that creative programming can provide.

On the other hand, some of the advantages of Java are also limitations. The security model developed for the Netscape version of Java prevents programmers from reading or writing files, saving state, querying the underlying machine about capabilities, and connecting to any server other than the one providing the applet. Furthermore, while Java applets should be able to talk to other Java applets on the same HTML page, they will not be able to communicate with plug-ins or OCXs. Microsoft labels this kind of security "sand box"—you can play only in the sand box, and Nanny won't let you play outside.

Security features such as these are double-edged swords. While users can trust an applet much more than they can trust an OCX, they pay for this with fewer capabilities than they have come to expect from desktop applications. Another interesting aspect of the security model, the lack of any sort of persistence of a Java applet on the client machine, leads developers and content providers to provide persistence at their Web site. In other words, your data, profile, and customizations are stored at the content providers' site, not on your personal machine. Talk about locking up your customers. Will customers stand for this? We'll see. Certainly content providers would love to tie up online customers in such a manner.

Another issue for Java applets may be overall performance. While it is unfair to evaluate performance of a beta product, it is a pretty good bet that an interpreter is slower than compiled code. This doesn't matter in many applications, but when it does matter it is critically important. Java applets, unable to leverage high-performance compiled code, will hit the proverbial wall.

Even with these limitations, amazing things are being done with Java applets right now on the Web. Visit www.gamelon.com to see the different kinds of Java applets that are being done: animation, data visualization, client/server, games, and so on. Reasonable users should have their security concerns mollified by the extensive provisions within Java and the Java interpreter. Because HTML pages have been static up to now (reader-pull/server-push weirdness aside), anything is an improvement. Java doesn't need an infrastructure to "certify" an applet. You can do cool things with Java right now. You can circumvent some of the odder security limitations, such as the inability to connect to any server other than the one providing the applet, through clever server programming.

And because Java is cross platform, it will not require the Win32 API underneath. As a result, publishers seeking to reach the widest audience can depend upon Java and the supplied Netscape framework to be the lowest common denominator.

• ***JavaScript.*** JavaScript essentially serves the same function as VB Script: it provides a way to embed event handling and functions into an HTML page. You can write JavaScript to drive applets, plug-ins, and Navigator objects (document, URL, frame, button, checkbox, and so on). To see examples of JavaScript, link to http://home.netscape.com/comprod/products/navigator/version_2.0/script/interest.html with Netscape Navigator 2.0b3 or later. JavaScript does not support reading or writing to the local hard disk.

JavaScript has the advantage of security and language constructs similar to Java, but several differences between the object-based JavaScript and the object-oriented Java make it a new learning experience. You can begin to play with JavaScript right now, because Netscape Navigator 2.0b6 supports it. So at this time, JavaScript is the only game in town.

Right now, you develop JavaScript with a basic text editor. No IDE for you! Will Netscape or a third party develop one for JavaScript? Hopefully so, because even a poorer language will almost always succeed over a technically superior language if it offers a good development environment. As we proceed to table/frame/forms-enabled HTML, scripting, and applets, we face a geometric explosion of complexity. An IDE is one of the main tools by which developers manage that complexity.

• ***VB/VBA/VB Script.*** See a trend here? You should. VB Script is a subset of Visual Basic for Applications, which is a subset of Visual Basic. VB Script is almost exactly comparable to JavaScript, so it will be interesting to see how this competition plays out (see my other feature, "Collision Course: JavaScript and VB Script," in this issue).

Microsoft will provide VB Script for all flavors of Windows and Mac PowerPC, and is working with third parties to provide VB Script for Unix. Internet Explorer 3.0 will support VB Script as well as OCXs.

As I mentioned, an IDE is a critical benefit to developers working on Web pages enhanced with applets, OCXs, plug-ins, scripts, or objects. Hand-coding Web pages with all this complexity will probably go the way of hand-coded Postscript—beautiful but deadly slow to develop.

• ***Plug-ins and OCXs.*** The biggest advantage of plug-ins over OCXs is that they work right now within Netscape Navigator. That plug-ins cannot talk to other plug-ins, communicate with the container or browser much beyond reading and writing streams, expose methods and properties, automatically install, support sets of methods and properties (interfaces), and so forth is fairly unimportant right now.

What's important is that you can use plug-ins to do many tasks that Java applets cannot handle, such as feasible image processing, fast text processing, sound processing, video support, and exploiting the features of the underlying platform. Developers are rushing to build inline plug-ins for Netscape Navigator.

In a sense, plug-ins validate Microsoft's support for OCXs. Users need components that can deliver the performance, functionality, and quality that only code compiled and tailored for a given platform can provide. Both inline plug-ins and OCXs provide that, but OCXs go well beyond providing this basic value and are more useful in a scripting or programming environment than a simple plug-in. (Now if we could just get them <g>.)

JAVA'S BUILT-IN CLASSES ALLOW DEVELOPERS TO EASILY ACCESS URLs OVER THE WEB.

• ***HTML & frame-based layout.*** Right now, the only way to control layout in HTML as you do in PageMaker is to use tables and frames. Unfortunately, as cool as these two approaches are, neither really does the trick. Tables don't support any kind of z-order and, well, are still tables. Frames don't have z-order either, behave strangely, and are a real bear to program. In fact, both approaches remind me of the old "neat desktop" approach used, and later discarded, by the original Windows. What this means is that designers will continue to have a tough time uniquely branding HTML pages.

This situation won't last, though. Some company will fill this need, either by going around HTML entirely as Adobe has done with Acrobat and as Microsoft tried with Internet Studio, or extending HTML to provide yet more tags to control layout and developing authoring tools that use these tags. My bias is that extending HTML is more likely to succeed than skipping HTML altogether. A lot of software and people support HTML, and any custom front end will have a hard time providing the breadth and depth of support that an HTML solution gets for free.

THE CRYSTAL BALL

What fun, let's be futurists. The future will bring inconceivable synergisms and events, to be sure, but I won't let that stop me from making wild guesses. And it's such fun to predict winners and losers in this business.

It seems that Microsoft and Sun/Netscape offer comparable technologies, though they come at it from different perspectives. Netscape thinks the network is the platform and all those machines running Unix, System 7.5, or Windows are just a complexity that can be abstracted away. Microsoft thinks that Windows, specifically 32-bit Windows, is the *real* platform, that all those other platforms are minor players, and that you *cannot* abstract out the operating system without paying a penalty in functionality.

Of course, baldly stating the differences in such simple terms is an overstatement. Netscape will justifiably point to plug-ins as an excellent way to exploit the underlying platform. Microsoft will justifiably point to VB Script and OCX for Macs as cross-platform solutions. But reasonable people never have any fun prognosticating, so we'll leave it as simple as a political attack ad for the moment. I deserve some fun for wading through all the spec-sheets, support papers, samples, crashing programs, and vapor to write this article.

My gut feeling is that some sort of mix-and-match solution will appear on the Internet for commercial and high-end corporate World Wide Web sites. A lot of this will be driven by the first workable IDE, but my guess is that in the end, a "maximized" Web site will use Java applets, VB Script, *and* HTML frames to provide the biggest "punch" to the widest audience. If something like an inline plug-in is needed, OCXs will probably be the preferred technology over inline plug-ins. Economies of scale and greater component-ness will probably drive that decision.

It will be an interesting struggle to see if Java gets squeezed out between the scripting languages and OCXs, or if Java forces the OCX into a marginal existence as a plug-in replacement. Java as a scripting language would be very interesting, with its support for multithreading, inheritance, Web-centric framework, and security.

I don't know if you are confused yet, but I still am. There are still many things I'd like to know. Will Netscape support Java on Windows 3.1? Will Microsoft add security features to VB Script? Who will deliver the first IDE for building rich, scripted HTML titles that incorporate OCXs as well as content? How much of this technology is needed for commercial World Wide Web development? Corporate? Personal? Will OCXs automatically install over the wire? Will interpretive cross-platform technologies really work when building real-world solutions? When will all this stuff really work, and how will the providers make money? Who is the Betamax analog and who is the VHS?