Chapter 3: Choosing Technologies

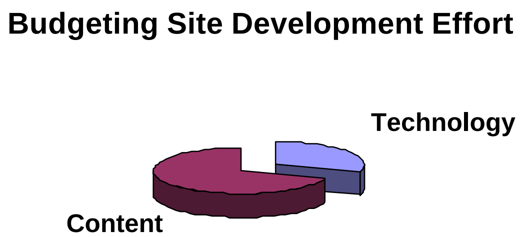
As you plan your community networking project, you face an extremely important, even critical, question: what technologies will you use for your project? Make the wrong choice of technologies, and you could find yourself lost in a technological morass.

Actually, you face choices to make in several areas:

- Technologies for preparing content – "authoring tools," image editors, etc.
- Technologies to help you manage your Web site.
- Technologies for serving content – Web server hardware and software.
- Technologies your users need to view the content you serve – so-called "plugins" and viewers.

Often, your choice in one area will affect another area. For instance, if you decide to serve online versions of written reports using Adobe Acrobat, you've implicitly made an assumption that your users have the Acrobat "Reader" installed alongside their Web browsers. Conversely, you will need Acrobat authoring tools (discussed later in this chapter) to prepare files for placing on the server. Similarly, if you decide to offer a 3D tour of your downtown using Shockwave Flash, your users will need the Flash plugin in order take the virtual walk, and you'll need software to prepare Shockwave content.

Along the same lines, if you decide that Microsoft FrontPage is an authoring tool of choice, you may want to choose a Microsoft Web server package, or install Microsoft FrontPage Server Extensions on your non-Microsoft server. This will make it easier for your content providers to publish content using your server.



In fact, you will probably find that many of the stakeholders in your community information project have their own opinions as to which technologies you ought to use. If you listen to every person with a technology opinion, you're likely to find yourself adopting a number of different technologies, some overlapping, and some incompatible. Your job is to select the right mix of technologies to get the job done, so that you can spend most of your efforts on content, not technology.

Think about the aggregate resources you have available to you for your community networking project. Add up the talents and time available from a mix of paid and volunteer workers, and you'll realize you only have so many talent-hours at your disposal. The question is how will you spend your scarce resources? In other words, if you've got 100 person-hours per week of effort to put into your site building project, will 70% go to content and 30% to technology – or will it be 30% content and 70% technology?

As the Internet industry has evolved, it has brought to market an ever-widening set of technologies, many of which are proprietary. The interest of the industry is not necessarily to improve the quality of the Web and the Internet; the industry wants to make money. In some ways, the proliferation of technologies is an attempt to sell newer and more complex tools – some of which will not necessarily enhance the quality of your Web site.

The bottom line is simple: your technology choices are every bit as important as the choices you make for the goal of your Web site, the partners who will work on the site, and how the site will be organized.

Of course, your initial choices need not lock you into an irrevocable technological straightjacket. In fact, it is almost axiomatic that you will eventually migrate from your initial suite of technologies to other, newer technologies. Nevertheless, if you don't make any technology choices up front, you'll find you're under pressure to adopt *every* technology – and then you're not going to be spending any time at all on content!

Technology as Building Blocks

The range of technologies available to you is mind-boggling. It may be useful to think of these various technologies as a set of building blocks you can employ in your site building.

The simplest of Web sites require only a very few fundamental elements:

- A Web server capable of serving static pages according to the rules of the Hypertext Transfer Protocol (HTTP).
- A series of HTML (Hypertext Markup Language) pages organized under the file system hierarchy belonging to that Web server. (If you're not familiar with HTML concepts, you may want to skip to Chapter 4 and learn the basics of HTML tags, and then revisit this discussion.)
- Inline image files to represent diagrams, charts, or photographs that enliven some or all of the HTML pages in the site.

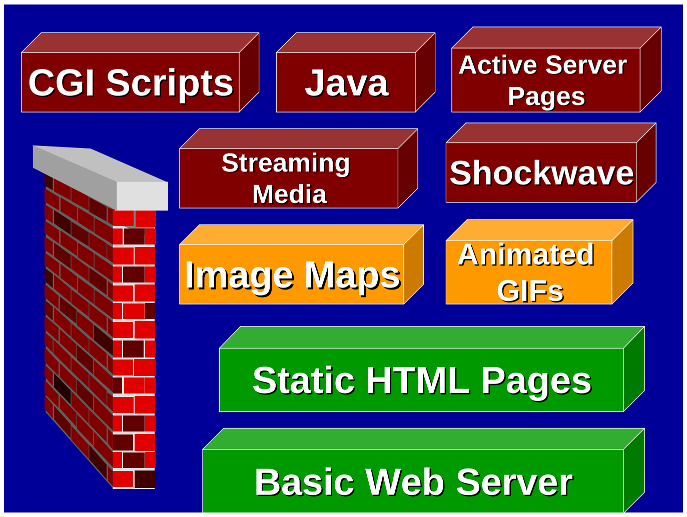
A basic Web server environment consists of server hardware, a server software package, and a permanent (full-time) direct Internet connection. That, in a nutshell, is all the technology you require. *You may require no additional technology to accomplish your Web publishing mission.* This basic environment involves some pretty sophisticated technology – for instance, the Web server will need to be connected to the greater Internet, which requires technologies such as Ethernet, leased lines, routers, and so forth.

Your HTML documents at the outset will probably be "static" pages stored in the file system of your server. By "static" HTML pages, we mean HTML files that are prepared by your content providers and placed by hand on the server for retrieval by your site's viewing audience. “Static” means that each individual file is prepared in its entirety off-line, and left on the server until it is replaced with a new, updated file manually.

The alternative to static HTML files is any one of a variety of mechanisms for dynamically-updated content; some of these choices are covered later in this chapter. For the vast majority of community information sites – and sites in general for that matter – static HTML files will make up most or all of the content on a given site. For every file on the server, the model is the same: a content provider uses an authoring tool or text editor to prepare the file, and then the content provider places the file in the appropriate place on the server.

Static HTML files can accomplish quite a lot:

- They can deliver formatted text.
- They can refer to inline images (drawings or photographs).
- They can link to other files or other Web sites.
- Thanks to the HTML `<mailto>` tag, a static HTML file can invoke a user’s e-mail package when the user clicks on the appropriate hyperlink. (That is true, of course, only if the user has previously configured his or her Web browser to know the correct e-mail package to invoke.)
- They can deliver to the user’s screen a form to be filled in.



But static HTML files are limited in what they can accomplish. For instance, an HTML form requires some sort of intelligence back at the server to handle the data the user types in. A static HTML file can’t observe what a user is typing to try to catch errors before they are made, as a Javascript program could. In general, static HTML can’t aspire to be as animated or as interactive as more complicated technologies can be.

So your most basic design choice is this: what technologies, if any, will you choose beyond your basic Web server and static HTML pages? You should not feel it necessary to choose any additional technologies. Given that your pages include rich graphics in the form of inline images, you need not apologize for your site if it lacks any more elaborate technologies. A quite compelling Web site can be built using the basic building blocks of static HTML and inline images.

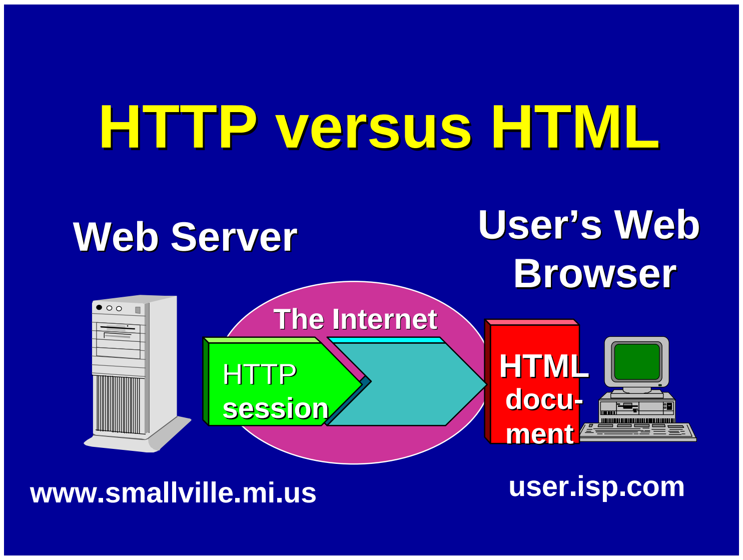
In general, the best guideline is to try to select the technology that is best suited to the kind of information you are presenting. Take care not to let elaborate technologies get in the way of delivering information in the most usable form to the greatest number of potential users.

Authoring, Server, and Client-Side Technologies

When evaluating what technologies you want to use, it's useful to think about Web server technologies by category:

- Authoring tools help your content providers prepare HTML documents and other pieces of content such as inline images. Examples include Microsoft FrontPage, the Netscape composer, Adobe PageMill, and HTML Assistant Pro. There are many other possible authoring tools to choose among; chapter 4 discusses authoring tools in more detail.
- Server technologies consist of the Web server software itself, plus any other tools necessary to accomplish your mission. You can augment your basic server setup with tools such as software to integrate your Web server with an SQL database. A simple server technology you'll want to have is a mechanism for allowing your content providers to place their HTML documents on your server.
- Client-side technologies are tools invoked in your user's environment. These can include "plugins" such as an audio player or a video player program (e.g. the Realaudio plugin). They also might include programs you deliver to the user written in Java, JavaScript, or under the ActiveX framework.

The choices you make in these three areas can affect your overall site development effort. For instance, most authoring tools provide some sort of mechanism for users to upload their finished HTML pages. These mechanisms require corresponding tools on the server to accept the content if the user is authorized to store the files. For example, if you choose the popular Frontpage editor from Microsoft, you will want to strongly consider installing Microsoft's "Frontpage Extensions" for your Web server. (Microsoft servers come with these extensions built-in.)



Animated GIFs

If you've spent any time at all surfing the Web, you've encountered animated GIFs: the "banner ads" you see so commonly on commercial sites – those elongated rectangles with dancing images advertising some product or service – are implemented using this simple technology. We'll discuss GIF files more in Chapter 6; for now, the important point is that the GIF (Graphics Interchange Format) file is a basic, universally recognized format for still images. The GIF format is designed to incorporate more than one image "frame" within a single image file, as well as control over how quickly each successive image will be displayed. In effect, an animated GIF is a tiny, self-contained cartoon.

There are three features of animated GIFs that make them attractive for use on the Web:

- Every Web browser knows how to display animated GIFs. Your user doesn't have to have a special plugin to view an animated GIF.
- Animated GIFs are relatively easy to build using widely-available shareware and commercial tools
- With a bit of preparation work, you can create animated GIFs that are small in file size, so that they download quickly.

There are Web sites that offer clip art libraries, where you can find a wide variety of ready-made animated GIFs. Such clip art might depict animations such as a letter going into a mailbox (as an icon for e-mail or a guestbook); a pulsating arrow (to draw attention to an important part of your page); a spinning globe – or just about any other symbolic message you might want to convey.

The main risk posed by animated GIFs isn't technological – it's a question of style. New webmasters are tempted for some reason to litter every page with a dozen animated GIFs. This makes for too much animation, making your page far less attractive and useful.

The message is simple: use animated GIFs sparingly, and they can enhance your site without requiring a major technology investment by you or your users.

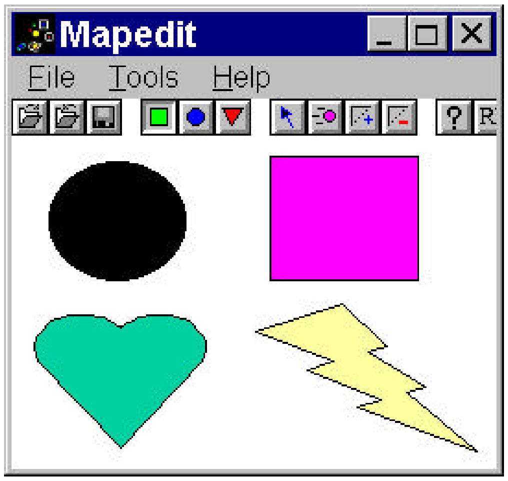


Image Maps

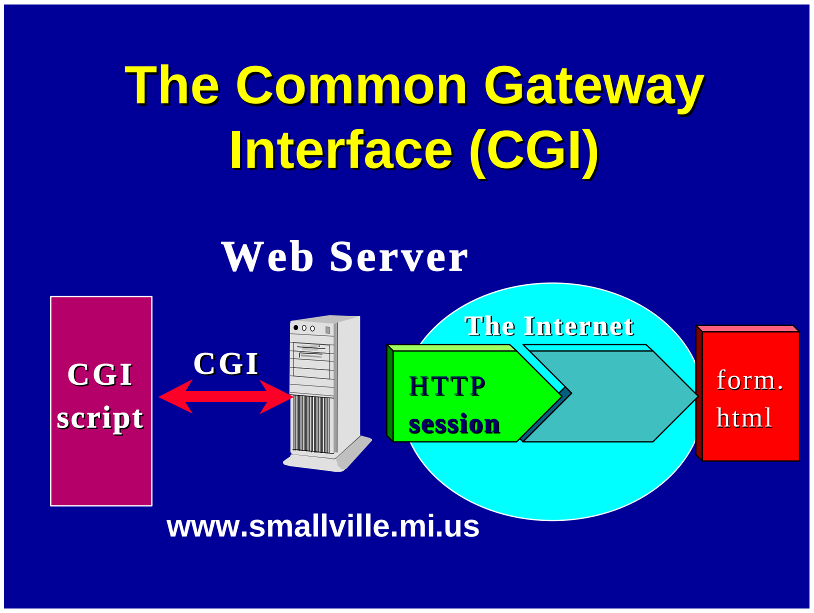
An image map is a mechanism that allows users to click on an element in an image, and in so doing cause the browser to load a particular page. For instance, the user might click on an icon of a heart and be taken to a page with information on heart disease.

An image map works quite simply: the page author first creates the image file, using a drawing utility or an image editor. Then, the author uses a tool to mark the coordinates of each of the icons within the image. That data is saved as part of the HTML for the page that refers to the image – and thus you have created what is known as a "client-side image map." We explore the process in greater detail in Chapter 6.

Image maps are relatively easy to set up and can be an attractive, effective mechanism for navigation. In a community information site, you might have an image map that consists of the main buildings of a town square – City Hall, the Post Office, the Parks & Recreation Department, and so forth. Although this is perhaps a somewhat trite metaphor, it's intuitive for users and can be a simple way to communicate in a friendly way the various major content areas of your site. Image maps are worth your consideration.

The Common Gateway Interface (CGI)

A technology that dates back to the first days of the Web in 1993, the Common Gateway Interface, or CGI, is the original mechanism developed for connecting a Web server with an external program, or script. CGI was developed as a set of rules that define the communication between the server and the external script.



In recent years, newer mechanisms have evolved that in many cases have replaced CGI. We discuss some of these mechanisms later in this chapter. Yet CGI remains a popular way to accomplish some simple kinds of tasks without the need to buy or install more sophisticated general interactivity tools.

For instance, suppose you want to implement a guest book for your site. Guest books are popular among webmasters because they allow a user to fill in a very simple form and quickly submit a comment or suggestion. You can implement guest books in a variety of ways depending on your server environment, but if you have a relatively simple server setup, you might choose a CGI script to accomplish the task.

Another very common application for a CGI script is the basic processing of forms. If you have a user fill in a form to perform a basic transaction – for instance, to reserve a baseball diamond at 7:00 pm on July 14 – a CGI script can take the data the user types and place it in a flat file or in a simple database. The script might even e-mail the form to the person who is responsible for reserving baseball diamonds.

If you choose CGI, you can either write a script yourself, or you can search for one on the Net. There are a number of CGI script archives that offer ready-to-run CGI scripts written in popular languages. (One of these, "Matt's Script Archive", was compiled by a young man who began his collection as a teenager when the Web became popular in 1993.) CGI scripts are written in the computer language chosen by the author, but certain languages are popular in certain server environments. For instance, the language PERL is popular among Unix server administrators, and VBScript is popular on Windows NT-based servers.

If you choose to keep your server environment basic, you may want to adapt or write a few CGI scripts to add some basic elements of interactivity or to support a few specific transactions.

The risk of using CGI is that it may not be the most efficient mechanism for supporting an ever-growing series of transactions. For instance, let's suppose you implement the baseball diamond reservation form in CGI. You may find this fill-in form so popular that residents of the community want to be able to reserve any recreation facility that way – and they want to perform many other kinds of transactions as well.

If you stick with CGI to implement more and more transaction forms, you may find that you have a mish-mash of scripts, when it would be better to have a different mechanism for interfacing with your database – such as Active Server Pages, discussed below.

Active Server Pages

Microsoft offers a popular Web server known as IIS (for Internet Information Server), and Microsoft has developed commercial tools that make it easy to integrate an IIS server with a database running under MS-Access, SQL, or even a non-Microsoft database product. This family of tools is known as Active Server Pages, or ASP. If you've ever encountered a Web site whose pages have file names that end in extensions of `.asp`, you've encountered an ASP-powered site. (Normally, static Web pages have names ending in `.htm` or `.html`.)

ASP applications can be developed using a Microsoft development environment called Interdev. Each version of Interdev comes closer to offering a drag-and-drop, or what-you-see-is-what-you-get, environment for building applications. (Note that third party vendors also offer tools to speed development ASP applications – just as they offer tools to speed development of Java-based applications.)

As you evaluate whether to implement ASP, consider these pros and cons:

In favor of ASP:

- Because of the popularity of IIS and Microsoft database products, ASP is already a popular environment, meaning tools and training materials are plentiful.
- The Interdev environment makes it relatively easy for someone with little programming experience to develop interactive Web applications.
- Productivity in developing new applications can be very high. Once you begin to put information in databases, it becomes easy to write many new applications, offering immediate, online access to the information your users desire.

In favor of caution with respect to ASP:

- Adopting ASP as a technology is more of an up-front investment than simpler schemes such as CGI. To get the benefits, someone on your team will need to understand all the pieces – IIS, your database package, your databases themselves, and ASP and Interdev.
- Some people feel that ASP locks webmasters too intimately into proprietary Microsoft tools.

The Rochester Hills Public Library demonstration site developed a sophisticated history application using ASP and an Access database. The Toolkit includes an interactive community calendar application based on ASP. That application is available as part of the Toolkit. See Chapter 10 for more information on the demonstration sites, and see Chapter 11 for details on the Toolkit software.

Cold Fusion

Cold Fusion is a proprietary Web-to-database programming language from Allaire Corporation. Cold Fusion is a somewhat simpler tool than ASP, but its purpose is similar: providing a way the author of a Web page can connect to a database. Whereas ASP pages are generated dynamically based on the specifics of the application, with Cold Fusion, a page author writes code in the Cold Fusion Markup Language. He or she then inserts these bits of programming code right in the midst of HTML tags. The resulting document is a blend of HTML and CFML, both of which goes onto the server. When a user clicks on a page that includes CFML, Cold Fusion steps into action.

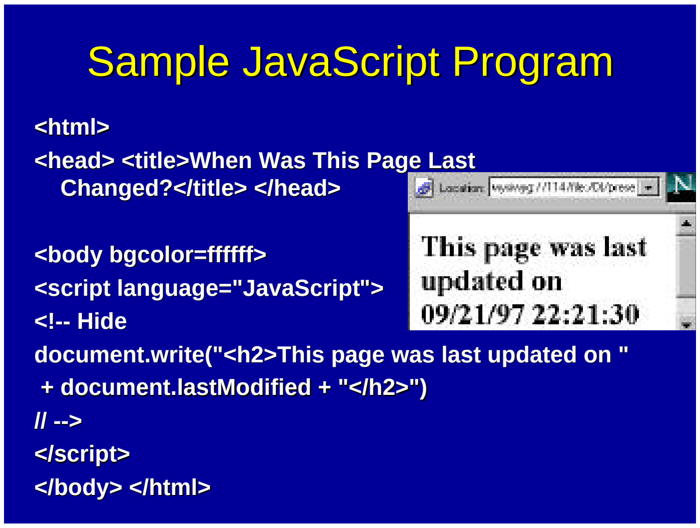
Cold Fusion probably has a somewhat easier learning curve than ASP, and may be less daunting for simple applications. However, Cold Fusion is older than ASP, and the recent developments in Interdev probably mean it's easier and faster to develop applications once you've mastered all the ASP concepts.

JavaScript

JavaScript is a language developed by Netscape. It was named after Java, the popular language developed by Sun, but in reality, JavaScript and Java are two very different critters. Let's consider JavaScript first, and Java later.

JavaScript is a programming language, but it requires no external "compiler" or language processor or program libraries: the code you write in JavaScript is inserted right in the middle of your HTML document. Web browsers understand JavaScript: they have a JavaScript "interpreter" built right into the browser.

JavaScript is useful as a relatively simple way to animate a page. One common example of JavaScript that you may have seen when surfing: often when you drag your mouse over the items in a menu bar, you'll see each menu item change as the cursor overlaps the item. Often a menu button will appear to "light up" as if you've shined a flashlight on it. This technique, known as a "rollover," is frequently implemented in JavaScript. With rollovers, what's going on under the covers is actually quite simple: JavaScript has the concept of "events"– such as the event of your mouse rolling over an object. The language has a method for the programmer to say when the mouse rolls over an object, a particular action should be taken. In the case of menu buttons that light up, the action is simply the substitution of a separately-prepared GIF image that's illuminated differently.



Sample JavaScript Program

```
<html>
<head> <title>When Was This Page Last
   Changed?</title> </head>

<body bgcolor=ffffff>
<script language="JavaScript">
<!-- Hide
document.write("<h2>This page was last updated on "
 + document.lastModified + "</h2>")
// -->
</script>
</body> </html>
```

This page was last updated on 09/21/97 22:21:30

JavaScript is useful for other kinds of simple client-side interactivity. For instance, JavaScript can be used to provide intelligent pop-up menus or other navigational aids. JavaScript is an accessible language, and it requires no special technology on the server side. There's relatively little risk if some of your content providers decide to incorporate JavaScript into their pages, but do consider a couple of points:

- Due to the unfortunate war between Netscape and Microsoft, the implementation of JavaScript in Internet Explorer may not behave the same as Netscape's implementation. You'll need to test pages that use JavaScript using both browsers.

- Like any other technology, it's easy to get too wrapped up in trying to enliven a page, whittling away on JavaScript animations instead of writing the content.

JavaScript is a relatively easy-to-learn programming language that can endow your Web pages with useful interactive capabilities.

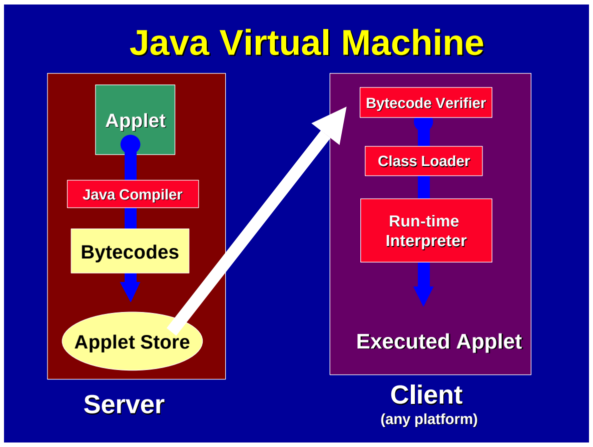
You'll find lots of resources about JavaScript on the Web. Sites such as `webreference.com` offer reference and instructional material about the language. Other sites offer archives of JavaScript code you can use. And because the JavaScript code appears inside HTML, you can invoke the "View Source" function in your Web browser and inspect anyone's JavaScript code for ideas on how to do similar tricks. (Note: it is a violation of copyright to "borrow" someone's JavaScript code without their permission. It is permissible to read their code to learn JavaScript concepts, but not to lift other's programs if they have not given permission to do so. Some programmers will include terms of reuse in the JavaScript code's comments.)

Another way to produce JavaScript code is to use an authoring tool such as Macromedia's Dreamweaver. For instance, Dreamweaver will let you use mouse drawing and drag-and-drop to design a menu bar. It will then create the necessary JavaScript code to enliven that menu bar. You can build pages with elaborate “mouseovers” without having to learn how to write a single line of JavaScript.

Java

Java is Sun Microsystems’ solution for programming in the Web environment (and elsewhere). Java is a serious programming language that requires a considerably higher level of programming skill than JavaScript.

Java programs can run on the server, but we most often encounter Java “applets” that are downloaded and run on the user’s desktop. Just as Web browsers have built-in JavaScript interpreters, they also have built-in Java “virtual machines.” An applet is written by a programmer, “compiled,” (run through a language compiler), and placed on a server for later use. A Web page author can refer to an applet via a simple HTML tag (appropriately enough, the <applet> tag), which will cause the applet to be downloaded and interpreted on the user’s computer.



When is Java the right tool to use? Java can be useful when you want to have some rather elaborate control over the user’s screen, doing complex animations or interactions.

The downside of Java is its complexity. Despite the fact that Java books dominate the Internet section in bookstores as well as the headlines in computer trade magazines, Java probably isn’t as widely used as Sun would have us believe.

A Java applet that makes your splash screen an animated marvel may frustrate users with slow modems or slow PCs.

Another problem is that Java implementation has varied over the different versions of browsers since its announcement, and of course Internet Explorer handles Java differently than Netscape does.

Also, Java can be slow to load. Your users who have older PCs dialing in over slow modems will appreciate it if your initial screen (your “splash screen”) does not force them to wait for a Java applet to load every time they visit.

For some applications, you can do things in Java that are hard or impossible to do in any other technology. If you have such applications, use Java. Otherwise, you may want to spend your effort elsewhere.

ActiveX

ActiveX is Microsoft's answer to Java. Where Java is a language, ActiveX is a way of accomplishing the applet idea – but with ActiveX, developers can program in any language. An ActiveX "control" is a little program that's analogous to an applet. It can be downloaded as referenced in a Web page, and it executes on the user's desktop to perform a particular task.

In Microsoft's Internet Explorer, ActiveX performs its task well and is a useful tool. Unfortunately, ActiveX is a Microsoft tool, and it's not implemented in the Netscape browser. As of this writing, the Netscape browser still has about one-half of the market, so programming in ActiveX would disenfranchise about half of your users. Until Netscape loses all its market share, or until all popular browsers run ActiveX, a community information site probably will want to steer clear of ActiveX.

Shockwave and Flash

Macromedia is a company that helped people build interactive CD-ROMs before the Web was born. Macromedia's tool known as Director was – and is – a popular authoring tool for building CD-ROM "worlds." For instance, one could implement a 3D virtual library that a user could "walk" through using the mouse.

When the Web became popular, Macromedia invented Shockwave, a way to play Director-produced content over the Web. Shockwave was effective, but could be rather slow for the user.

Therefore, Macromedia invented Flash. Flash is an efficient way to download the essence of a 2D or 3D environment as a series of vector-based objects. What that means in a nutshell is that your user can interact with a rich environment much faster than previously. The kinds of interactions you can offer with Flash make the typical Web experience look tame by comparison.

As with all the technologies we've examined, you can make a case for using Shockwave for certain applications. A virtual walk through the downtown of your community would be a perfect application for Flash. Whether the time to learn the technology and perfect the tour is worth the effort is another matter.

Besides the ability to create interactive 3-D environments, Shockwave also incorporates streaming media capability – our next topic.

Streaming Media

Although much of the information about a community can be conveyed well in text and still images, some content exists in other forms. For instance, one of the best ways to tell a story about your community is to record the voices of residents (particularly long-time residents) and build an oral history archive.

Until streaming media was invented, there was no good way to deliver audio content over the Internet – unless users had very fast connections and very large hard disks.

Streaming audio is a wonderful way for a community Web site to communicate oral history recordings.

Streaming media allows a user to begin playing an audio or video file as it downloads. Streaming media makes it possible to deliver high-quality audio to users who are dialing in at speeds as low as 28.8 kilobits / second.

Streaming media can be used to deliver both recorded content and real-time events. For recorded content, the audio (or video) stream is captured during the original recording, and encoded into the streaming format at some time prior to placing it on the server for delivery on user demand. For real-time delivery, content is encoded in the streaming format in real time during the speech (or other performance) and sent out over the Internet instantaneously by the streaming server.

The pioneering company in the field of streaming media is Real Networks, Inc., once known as Progressive Networks. Real has an overwhelming share of the market with their RealAudio and RealVideo products. Microsoft is rising in share with its Netshow family.

Is streaming media right for your community information site? It could well be. Working with audio is not difficult on any fairly recent computer, and the server tools needed for streaming are free to nonprofits and fairly easy to set up. You will want to think very carefully before tackling video over the Web, as editing is much more complicated, and, despite compression and faster modems, video over the Web remains a slow-scan, somewhat blurry experience. Chapter 7 explains how to work with digital audio editing, and Chapter 8 explains more details about streaming media.

Adobe Acrobat

Adobe Acrobat seeks to make it easy to move the printed page into the online realm. That may sound very similar to the goal of HTML, the language of the Web, but in fact Acrobat is quite different than HTML:

HTML is great for on-screen presentation, and generally terrible for printing. Adobe Acrobat PDF lets you deliver documents intended for printing and offline viewing.

- With HTML, you prepare documents for the online realm from the outset.
- With Acrobat, you convert documents prepared for the printed page into Adobe's Portable Document Format, or PDF, so that those documents can be distributed online – by e-mail, on CD-ROM, or via the Web. Users can either view PDF documents online, or print them for offline viewing.

PDF documents retain the structural aspects we normally associate with print documents; a defined page size is assumed for each document – say, 8.5 by 11 inches – and the document is formatted, or “laid out” using that specific assumption. A PDF file can be printed on a variety of printers, and the appearance will be similar no matter what printer is chosen – subject to the limitations of the printers themselves. Thus, anyone who downloads a PDF file via the Web and prints it will find it retains the presentation structure of the original. For instance, two people can download a document, print it out at their respective locations, and discuss the document over the telephone, referring to page numbers in order to locate pages to talk about. Try doing that with a long HTML document!

By contrast, HTML is poorly suited for printing. Page numbers and formatting will vary according to browser settings, which browser a user has chosen, and variations in printers. Users who print out Web pages find that they waste a lot of paper trying to capture all the information they want. Also, because Web sites tend to be organized into numerous separate HTML pages, users find they have to click on the Print icon many times to capture a single “document” – once for each page in the document.

In short, HTML is great for delivering content to online users. Although HTML isn’t so good for content that ultimately will be printed, PDF is great in that role. On the other hand, although PDF files can be viewed online using the Acrobat Reader, they’re often not too easily read on screen.

So when should you use Acrobat?

- When you have content that users will find more readable on paper, consider Acrobat. For instance, a 50 page report from your Planning Commission outlining your city’s ten-year Development Plan may be better delivered in Acrobat form than as HTML.
- When you have content that multiple people will need to discuss and comment on, consider Acrobat. For instance, a proposed resolution before the County Commission is probably more useful in printed form than as HTML pages: commissioners will want to refer to page numbers during deliberation, and paper is more convenient than computer terminals during a live meeting.
- When you have content that comes to you as a large document formatted for print distribution, consider Acrobat. Converting a large document to HTML is often time consuming, even given automated translation tools in programs such as Microsoft Word.

You should also keep in mind limitations of Acrobat:

- Although Adobe has put a lot of engineering effort into the Acrobat Reader and the Portable Document Format itself, PDF files can be very difficult to read on-screen. For instance, a newsletter may be laid out carefully in multi-column format for a great layout on the printed page, but that layout will make for cumbersome scrolling if users try to read it on-screen.
- Although recent browsers have Acrobat Reader functionality built-in, many users may have older browsers, or special-purpose browsers such as WebTV, and thus they will be unable to handle Acrobat files.

- Users with visual impairments may use "talker" technology that may not be able to interpret and read PDF files effectively.

In sum, PDF is often best used for delivering documents that were originally prepared for paper distribution, and offering them online as a more convenient form of distribution. PDF probably isn't the right choice for delivery of content being prepared initially for your community Web site. It's a mistake to bury basic information in PDF. It may be tempting to take the lovely brochure that lists City Hall services and slap it online in PDF, but you'll serve your audience far better if such basic content is offered in HTML. There's no point running the risk that a basic piece of data such as a phone number or name of a contact person can't be accessed because a given user doesn't have the Acrobat Reader.

By contrast, if you want to offer a 300 page printed report from the Planning Commission, you may find PDF a fast, convenient, and efficient way to make that document available for your users to download and print. And your users who lack the Acrobat Reader may be willing to pay the cost of downloading and setting up the Reader so they can print that long document at high quality for offline reading.

Adobe's Acrobat family includes these pieces:

- The Acrobat "PDF Writer," which is a sort of virtual print driver. The PDF Writer allows you to print to a file instead to a physical printer. The resulting PDF file can be distributed online for your readers to access. You install the PDF Writer once, then you can "print" a PDF file from any popular application, such as Microsoft Word or Corel Draw or Word Perfect. The PDF Writer is available as part of Adobe's Acrobat authoring package and is also bundled with some other products.
- The Acrobat "Reader" which your users must install as a plugin to their Web browser if they are going to fetch and read PDF files from your Web site. The Reader is available from `adobe.com` and other sites at no charge.
- The Acrobat "Distiller," which converts files from Adobe's Postscript language to PDF. This is useful if you have existing Postscript files to convert.
- Acrobat "Capture" – a high-end software package for scanning in paper content and managing large projects to convert paper archives to PDF.

Keep in mind that nothing precludes your offering the same content in both PDF and HTML forms. Depending on your original source document format, this may be quite easy to do. For instance, you might "print" a Microsoft Word document in PDF format using the PDF Writer, and "Save" the same document in HTML. (You could even go one step further, and offer the document in Word format, if your audience may wish to edit or annotate the document.) Multiple format choices may be a little more work for the webmaster, but choices can be liberating for users.

VRML and Other 3-D Environments

VRML, or the Virtual Reality Modeling Language, was developed relatively early in the life of the Web, starting in 1994. (The term is often pronounces "vermal.") Spearheaded by companies such as Silicon Graphics, VRML has always enjoyed the enthusiastic support of its advocates. Like many Web technologies, VRML probably has not lived up to the hype that came with its introduction.

VRML is a markup language analogous to HTML. Its goal is to provide a standard language for 3-D interactive worlds. A number of authoring tools make it possible to create VRML worlds without having to learn to write VRML code. The goal of VRML was to become an open standard with VRML browsing capability built into popular browsers, so that a VRML world could be experienced by anyone with a current Web browser. In practice, users end up choosing among an array of VRML plugins, after which they must download and install the plugin.

A VRML world might mirror a real world – the shelves and hallways of a library, or the sidewalks, statues, and trees of a town square.

Although VRML has attracted a great deal of attention and has pockets of strong advocacy, as a general proposition it is probably less often used than proprietary 3D environments such as Shockwave.

VRML and Shockwave are well-suited to animated, interactive environments – much as you'd see with a video game such as Doom. For many applications, such as a virtual walk-through of a town square, you may prefer photo-realistic scenes over metaphorical, cartoon-like renderings of the street scene. Products that offer photo-realistic 3D environments include Apple's Quicktime VR and Ipix.com's immersive Photobubble technology. The user places the mouse within the window of a still photograph, and is able to change perspective, panning left and right, zooming in, and following hyperlinks.

These tools require an investment on the authoring side – special cameras or lenses and special software to stitch separate photographs into a virtual mosaic – as well as requiring users to install corresponding viewers work with the user's browser.

Dynamic HTML (DHTML)

The term "Dynamic HTML" refers to a family of technologies whose goal is to give the Web author more control over the appearance of Web pages on users' screens. In theory, HTML tags do not imply any particular rendering on screen; rendering is the province of the browser and the preferences set by the user. But as the Web has evolved, authors have insisted on the kind of control of presentation they enjoy when laying out for the printed page, or for CD-ROMs, or for video.

There are two main components to Dynamic HTML:

- Cascading Style Sheets (CSS): Cascading Style Sheets are "style sheets" in the sense that they provide a framework for defining how text and graphics are to appear on a Web page. They are "Cascading" in the sense that a set of presentation rules can be defined in one file, and applied to part or all of an entire Web site. The rules "cascade" across the entire site. With CSS, you can control all of the aspects of layout you'd expect to be

able to control with any word processor: typefaces, font colors, background colors, background images, page margins, etc. And, just as word processors allow you to define these preferences in a generic way – for instance, level one headings are to be rendered as 16 point Arial Bold – you can define these preferences one time, in one place, and apply them across all of your Web pages.
- Object Positioning Extensions: Standard HTML gives very little control over how text and graphics are rendered on screen. Dynamic HTML extensions try to give authors precise control over screen layout. Alas, Microsoft and Netscape introduced radically different approaches to the problem with the 4.0 releases of Internet Explorer and Netscape Communicator.

DHTML gives you as Web author a great deal of power, but because of incompatibilities in the Microsoft and Netscape approaches to DHTML, you may wish to hold off on DHTML until the World Wide Web Consortium has prevailed in its attempt to promulgate a common standard accepted by all players, including the major browser vendors.

XML

Finally, we consider XML – the Extensible Markup Language. In Chapter 4 we will examine the language of the Web, the HyperText Markup Language, or HTML, in some detail. As we will see, HTML is theoretically concerned with "logical" markup of documents, but in practice HTML authors tend to worry about how a page will appear on screen, or presentation.

XML is an attempt to provide Web authors with the ability to create their own markup to meet their own needs. Suppose for instance that you are putting up letters from Civil War soldiers to and from their families. You might want to keep track of certain topics or concepts within these letters – such as references to battles, to injuries, to names of officers, and so forth. With XML, you are free to make up your own set of tags to define any of these concepts. For instance, you might create tags such as `<battle>` and `<injury>` and `<officer>`. An example use of one of your tags might be:

> We had not seen such fury on the battlefield since
> `<battle>` Bull Run `</battle>`.

Once you've "marked up" a document showing all these conceptual elements within a document, the power of XML comes into play. An indexing tool could examine all of your Civil War documents and build an index of all the references to battles. Users doing historical research could examine, for instance, different ways that officers and soldiers write about battles.

In practice, you probably would not want to create your own set of tags for each application. Instead, you would look around to see who has done similar work, and see if they have created a set of tags that meet your needs. Various communities of interest, such as the Text Encoding Initiative, have found it best to work together to devise a common set of tags to meet needs common to the group.

XML is a powerful methodology based on two decades of experience with SGML, the Standardized General Markup Language. SGML has been widely used in humanities research, by the military, and as an internal tool in the publishing industry. XML attempts to minimize some of the thorny problems associated with SGML while extending its power to the masses of authors and users of the World Wide Web.

Historically, SGML and XML have been most popular for "marking up" large collections of unstructured text. Some people describe SGML, and now XML, as object oriented tools enabling re-use of text. Today, XML and related standards are also seen as ways that cooperating Web sites can exchange information in standardized ways, for instance for electronic commerce applications.

There is a great deal of promise for XML, and there are some pioneering applications online. Microsoft and Netscape have begun putting XML technology into their browsers. SGML adherents will advise you that you should begin your new Web publishing project with XML at its core. However, as of this writing, it is probably safe to say that far fewer than 1% of Web publishing efforts exploit XML at all. It will require up-front work to understand XML concepts and choose tools and procedures that exploit its capabilities. (However, XML adherents will tell you that by adopting XML at the beginning of your project, you will create a much more effective collection of online information.)

Should you use XML for your community information network? As with all technology choices, you will want to consider this question carefully before committing. You will probably want to see whether similar CI projects have adopted XML, and how well the technology choice has met project needs.

There is a great deal of information about XML on the Web and in bookstores. A very good starting point is the XML FAQ, written by Peter Flynn and available at:

`http://www.ucc.ie/xml`

An excellent place to see the power of SGML methodologies for historical applications is the Electronic Text Center at the University of Virginia:

`http://etext.lib.virginia.edu`