CHAPTER 28

Integrating JavaScript and HTML

by Rick Darnell

IN THIS CHAPTER

- Getting to Know JavaScript *544*
- Creating Scripts *547*
- Using JavaScript in Your Web Page *557*
- Controlling Browser Behavior *561*
- JavaScript Resources on the Internet *561*

JavaScript is a scripting extension to HTML that extends your ability to respond to user events without the need for client/server communication or CGI scripting.

Before scripting languages, material such as HTML forms was typically submitted to the server for all processing, whether it meant checking a ZIP code or putting information in a database. The process bogged down each time information was passed back and forth between client and server due to inherently slow communication lines. JavaScript eliminates much of the client/server communication by shifting responses to user events to the client side. Because network transmission is not required, the process goes much faster.

Getting to Know JavaScript

JavaScript is more closely related to a programming language than to HTML tags. However, JavaScript cannot exist outside of HTML. To function, it must be included as part of a page.

What Is JavaScript?

JavaScript was developed by Netscape in conjunction with Sun's Java. You may also know it as LiveScript—the first name it had before the collaboration with Sun.

JAVASCRIPT ISN'T JAVA

Java is an object-oriented programming language used to create standalone applications and applets, special mini-applications for Web pages. Java is compiled into machine-independent byte codes, which are in turn interpreted by the Java virtual machine on the host computer. Writing Java programs is easiest when you have some background in programming languages, such as C or C++.

JavaScript shares some of the same syntax and structure as Java, but provides a much smaller and simpler language for people with HTML or CGI experience. It is interpreted along with the rest of the page at load time. JavaScript only resides within an HTML document and provides for greater levels of interactivity than basic HTML commands.

For example, JavaScript enables the HTML author to respond to user-initiated events, such as mouse clicks and form activity, without the need for client/server interaction. The result provides quicker operation for the end user, and less load on the server.

Although similarities exist between Java and JavaScript, the languages are different and are intended for different uses. A simple form-handling routine that would require a significant amount of coding in Java represents a basic task for JavaScript, but creating a browser such as HotJava in JavaScript is impossible.

Java applets occupy a defined space on the screen, much as an image or other embedded item. Though a Java applet can communicate with another applet on the same page, communication with a page's HTML elements requires a substantial amount of code. For more information about Java, see the next chapter, "Integrating Java Applets and HTML."

JavaScript does not represent a watered-down version of Java for programming beginners. Although related to Java, it provides a solution for client-side scripting in an era when users with high-powered machines get bogged down by client/server communication.

Although many ways exist to control the browser from within a Java applet, simple tasks such as computing a form or controlling frame content can become complicated affairs. JavaScript bridges the gap by enabling HTML authors to implement basic HTML functionality and interactivity without hours and hours of writing code.

A narrower focus and application for JavaScript means there is a much smaller set of objects, methods, and properties to work with, and they are all focused towards dealing with HTML content. For example, JavaScript does not have the capability to control network connections or download files.

Why Should I Learn JavaScript?

Although based on programming, JavaScript is simple enough to be within easy reach of anyone who feels comfortable with HTML coding. It greatly expands the capabilities of typical HTML pages, without a great deal of hassle.

Take the following example:

```
<SCRIPT LANGUAGE="javascript">
document.writeln("This page last changed on "+document.lastModified());
</SCRIPT>
```

When the page loads, some basic information about itself is also included, such as the time and date the page was saved. Without any further communication with the server, JavaScript accesses the date and displays it for the user. You don't need to remember to update a line of HTML or include a link to a CGI script—the process is automated with the JavaScript line.

With JavaScript, you can also effectively manage multiple frames. You can control the content in other frames by loading them with new URLs or managing form input. For more information on creating and working with HTML frames, see Chapter 18, "Creating Sophisticated Layouts with Frames and Layers."

One major power of JavaScript is revealed in its capability to handle forms and their elements. Using JavaScript, it's possible to validate and check information on a form before it is sent to the server, saving valuable network bandwidth and processing time on the server. Client-side form processing also localizes validation of potentially destructive content, making it much harder for end users to send incompatible data that could cause damage to the server.

Additional characteristics represent form elements in the shape of events and event handlers. For `INPUT TYPE=` tags such as `BUTTON` and `TEXT`, the page author can check for mouse clicks, changed text, focus, and even change the content of form elements. Submission of forms and other information is also controlled by substituting custom actions for the standard submit button formats.

How to Use JavaScript Now

Most HTML editors that handle nonstandard HTML tags will allow creation and editing of JavaScript sections to your pages. Several resources available for learning the syntax and idiosyncrasies of JavaScript appear at the end of this chapter.

TIP

One of the most useful sites is the JavaScript online documentation at `http://home.netscape.com/eng/mozilla/Gold/handbook/javascript/index.html`, which is also available as a downloadable file.

For users to take advantage of your JavaScript-empowered pages, they need to use a compatible browser. Currently, the list is limited to the two most popular selections—Netscape Navigator (Version 2.0 and later) and Microsoft Internet Explorer (Version 3.01 and later). Other lesser-used choices, such as NCSA Mosaic and Sun HotJava, can also be expected to follow suit, although no timetable is available for JavaScript implementation.

WARNING

The JavaScript API is not entirely stabilized yet. As Netscape continues its own development and collaboration with Sun and Microsoft, the implementation and workability of some features may change. You should always try your script on more than one browser and platform combination to ensure that your solutions function as planned.

NOTE

JavaScript is JavaScript, whether it appears on Internet Explorer or Navigator, right? Wrong. JavaScript is essentially Netscape's invention, and they call the shots on its evolution. That would leave Microsoft stuck in an eternal game of catch-up, except that Microsoft developed its own version of JavaScript called JScript.

The two implementations are very close—both include objects such as *window* and *document*, both support user events, and their basic syntax is the same. However, Netscape supports several objects not included by Microsoft, and Microsoft includes an expanded set of user events not supported by Netscape.

One of the most significant differences is in case-sensitivity. Microsoft is much more forgiving in its use of capital letters for objects and methods. Therefore, if you write a script that should work for both Microsoft and Netscape, it may crash on Netscape because of the difference in capitalization.

If you're developing for both browsers, start by testing your work on Netscape. When it's stable, move it to the Microsoft platform and check it there. Netscape is driving the train, so you'll want to stay current with their work, and then make sure it's compatible with other implementations.

As do many sites that provide a host of compatibilities, some of which are mutually exclusive, it shows good manners to offer your page in a generic version if crucial content is not usable with a non-JavaScript browser. You can also offer a link to sites where a compatible browser is available for downloading.

Creating Scripts

Creating scripts is really quite simple, although you must use proper syntax to ensure success. Although a knowledge of object-oriented programming is useful to anyone creating functions with JavaScript, it is not a necessity.

New Language, New Terminology

Objects, methods, properties, classes—an object-oriented world has taken off, and a whole new batch of terminology has cropped up to go with it. This section provides a quick primer on the basic terms that are used in conjunction with an object-oriented language such as JavaScript.

TIP

JavaScript is case-sensitive. For example, if your variable is called `Skunk`, you can also have `SKUNK`, `skunk`, and `skunK`, and each one will have its own unique smell.

Object

Objects are at the heart of any object-oriented programming language, and JavaScript is no different. An *object* is a software model, typically used to represent a real-world object along with a set of behaviors or circumstances. In JavaScript, built-in objects can also represent the structure, action, and state of an HTML page.

In object-oriented terminology, the actions are called *methods* and the states are called *properties*. Both of these terms are covered later in this section.

To build an object, you need to know something about it. Consider a squirrel as an example. A squirrel has several physical properties, including sex, age, size, color. It also has properties relating to its activity, such as running, jumping, eating peanuts, or tormenting dogs. Its methods relate to changes in behavior or state, such as run away, stop and look, or twitch tail and chatter.

This example may seem all well and good, but how do you represent this idea as an object in JavaScript? The basic object creation is a two-step process, beginning with a defining function that outlines the object, followed by creating an instance of the object. Using some of the properties listed in the preceding example, you make a JavaScript squirrel. (See Listing 28.1.)

Listing 28.1. A JavaScript object definition for a squirrel.

```
function squirrel(color) {
    this.color = color;
    this.running = false;
    this.tormentingDog = false;
}

var groundSquirrel = new squirrel("brown")
```

The first part of the script with the `function` tag outlines the initial state for any given squirrel. It accepts one parameter, called `color`, which becomes a property, and adds two more properties called `running` and `tormentingDog`, both set to false by default. If you don't understand all the syntax and notation, don't worry. We'll get into those details just a little bit later in this chapter.

By itself, the function does nothing—it has to be invoked and assigned to a variable. This is what happens in the next step, where a variable called `groundSquirrel` is created and given the color brown. The following code shows how the object and its properties are represented:

```
groundSquirrel.color // "brown"
groundSquirrel.running // false
groundSquirrel.tormentingDog // false
```

Now, to implement the object as part of an HTML page (as shown in Figure 28.1), include the object definition between the `<HEAD>` tags. (See Listing 28.2.)

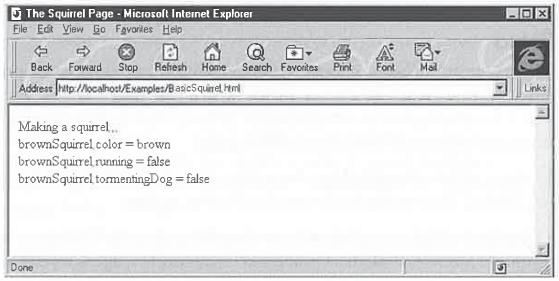


FIGURE 28.1.
The squirrel page creates a simple object and displays its properties.

TIP

The two slashes (//) are used to include comments in your JavaScript code. Anything after them is ignored by the interpreter.

Listing 28.2. Use the JavaScript definition of a squirrel in an HTML document similar to this one.

```
<HTML>
<HEAD>
<TITLE>The Squirrel Page</TITLE>
<SCRIPT language="javascript">
<!--
function squirrel(color) {
    this.color = color;
    this.running = false;
    this.tormentingDog = false;
}

// -->
</SCRIPT>
</HEAD>
<BODY>
Making a squirrel...
<BR>
<SCRIPT LANGUAGE="javascript">
var brownSquirrel = new squirrel("brown");
document.writeln("brownSquirrel.color = "+brownSquirrel.color);
document.writeln("<BR>brownSquirrel.running = "+brownSquirrel.running);
document.writeln("<BR>brownSquirrel.tormentingDog = "
                 +brownSquirrel.tormentingDog);
</SCRIPT>
</BODY>
</HTML>
```


Class

A *class* represents the definition for a type of object. Although classes are in Java and not in JavaScript, it is helpful to understand classes because many discussions about either language may refer to them.

Simply stated, a class relates to an object as a pattern for a stuffed squirrel relates to the actual toy. The pattern contains all the information about the stuffed squirrel, but you can't play with it. To play with the stuffed squirrel, you need to create an instance of it. In object-oriented terminology, this process is called *instantiation*.

Classes can also have inheritance, which means they take on the behavior of other classes. A gray squirrel and a red squirrel both have basic squirrel characteristics, but different specific traits and behaviors. They are considered subclasses of the squirrel class.

Although a JavaScript function has a definition similar to a class, it can operate without instantiation.

Property

Properties are the individual states of an object, typically represented as variables. In the squirrel example, `color`, `running`, and `tormentingDog` all represent properties of squirrel. An object's properties can include any of the valid JavaScript variable types.

WHICH TYPE IS WHICH?

A variable's type is the kind of value it holds. Several basic variable types are offered by JavaScript, including string, Boolean, integer, and floating-point decimal.

JavaScript utilizes loose casting, which means a variable can assume different types at will, as in the following example:

```
squirrel.color = "pink"
...statements...
squirrel.color = 30
```

Both color values are valid. In Java, this would cause an error because it incorporates tight casting. After a variable is assigned a type in Java, it can't be changed.

Loose casting can make life easier when working with JavaScript. When building strings, for example, you can add a string to an integer, and the result will be a string:

```
value = 3;
theResult = value + "is the number." //Results in "3 is the number."
```

The downside is that sometimes you can easily forget what a variable thinks it is. It's a good idea to try to keep variables to their original type unless absolutely necessary.

Object properties are accessed using the object's name, followed by a period and the name of the property:

```
squirrel.color
```

Assigning a new value to the property will change it:

```
squirrel.color = "pink"
```

Function

A JavaScript *function* is a collection of statements that are invoked by using the name of the function and a list of arguments, if used. As a general rule, if you use a set of statements more than once as part of a page, it will probably be easier to include them as a function. Also, any activity used as part of an event handler should be defined as a function for ease of use.

Functions normally appear in the `HEAD` portion of the HTML document to ensure that they are loaded and interpreted before the user has a chance to interact with them.

The syntax to define a function is

```
function functionName ([arg1] [,arg2] [,...]) {
...statements...
}
```


An example of a function that automatically generates a link to an anchor called `top` at the top of the current page could look like this:

```
function makeTopLink (topLinkText) {
    var topURL = "#top";
    document.writeln(topLinkText.link(topURL));
}
```

This function accepts a text string as its one argument and generates a hypertext link, similar to using the HTML `<A HREF>` tags.

```
makeTopLink("Return to the top.");
makeTopLink("top");
```

Method

If properties represent the current conditions of the object, methods serve as the knobs and levers that make it perform. Consider the squirrel example again. Defining a squirrel seemed easy enough, but what about making it do something? First, the methods need to be defined as JavaScript functions.

The first method for the squirrel makes him run and quit tormenting the dog:

```
function runAway() {
    this.running = true;
    this.tormentingDog = false;
    document.writeln("The squirrel is running away.");
}
```

The second method makes the squirrel stop moving and tease the dog:

```
function twitchTailChatter () {
    this.tormentingDog = true;
    this.running = false;
    document.writeln("The squirrel is being annoying.");
}
```

One more method would help you see what happens to the squirrel as his state changes:

```
function showState() {
    document.writeln("<HR><BR>The state of the squirrel is:<UL>")
    document.writeln("<LI>Color: "+this.color+"</LI>");
    document.writeln("<LI>Running: "+this.running+"</LI>");
    document.writeln("<LI>Tormenting dog: "+this.tormentingDog+"</LI>");
    document.writeln("</UL><HR>");
}
```

TIP

You can include HTML tags in text written to the browser screen using JavaScript's `write` and `writeln` methods. These methods are interpreted like any other HTML text, so formatting can occur for generated content.

Now that you have three methods defined, you need to make them a part of the object. This step amounts to including the method names as part of the object definition:

```
function squirrel(color) {
    this.color = color;
    this.running = false;
    this.tormentingDog = false;
    this.runAway = runAway;
    this.twitchTailChatter = twitchTailChatter;
    this.showState = showState;
}
```

Finally, the last step is to include the whole package as part of an HTML document, such as Listing 28.3, and see whether it works. (See Figure 28.2.)

Listing 28.3. Using the JavaScript definition of a squirrel and its behavior requires an HTML document similar to this one.

```
<HTML>
<HEAD>
<TITLE>The Squirrel Page</TITLE>
<SCRIPT language="javascript">
<!--
function runAway() {
    this.running = true;
    this.tormentingDog = false;
    document.writeln("The squirrel is running away.");
}

function twitchTailChatter () {
    this.tormentingDog = true;
    this.running = false;
    document.writeln("The squirrel is being annoying.");
}

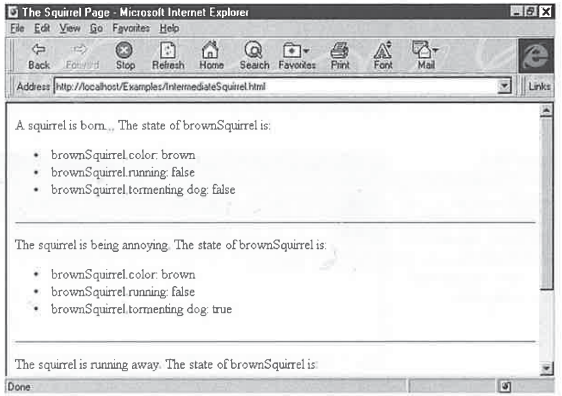
function showState() {
    document.writeln("The state of "+this.name+" is:<UL>")
    document.writeln("<LI>"+this.name+".color: "+this.color+"</LI>");
    document.writeln("<LI>"+this.name+".running: "+this.running+"</LI>");
    document.writeln("<LI>"+this.name+".tormenting dog: "+
                     this.tormentingDog+"</LI>");
    document.writeln("</UL><HR>");
}

function squirrel(color,squirrelName) {
    this.name = squirrelName;
    this.color = color;
    this.running = false;
    this.tormentingDog = false;
    this.runAway = runAway;
    this.twitchTailChatter = twitchTailChatter;
    this.showState = showState;
    document.writeln("A squirrel is born...");
}
```



```
// -->
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="javascript">
var brownSquirrel = new squirrel("brown","brownSquirrel");
brownSquirrel.showState();
brownSquirrel.twitchTailChatter();
brownSquirrel.showState();
brownSquirrel.runAway();
brownSquirrel.showState();
</SCRIPT>
</BODY>
</HTML>
```

FIGURE 28.2.
The browser screen displays the activity of the JavaScript object as each method is executed.



The <SCRIPT> Tag

As seen in the earlier examples in this chapter, JavaScript requires its own special tag to mark its beginning and end. The basic form of the tag appears in the following code:

```
<SCRIPT [LANGUAGE="JavaScript"]>
...statments...
</SCRIPT>
```

A problem can develop when users view a page embedded with JavaScript statements with a noncompatible browser.

Note that Figures 28.3 and 28.4 both use the same HTML document (shown in Listing 28.4), which includes JavaScript statements; however, Figure 28.3 is viewed with Internet Explorer 3.0 and Figure 28.4 is viewed with NCSA Mosaic.

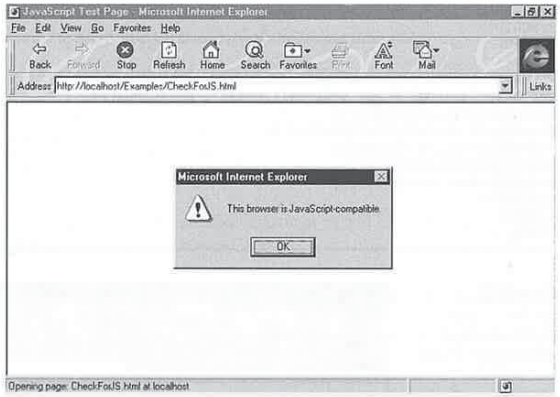


FIGURE 28.3.
JavaScript-compatible Internet Explorer 3.0 displays this HTML document and processes the JavaScript commands contained in it.

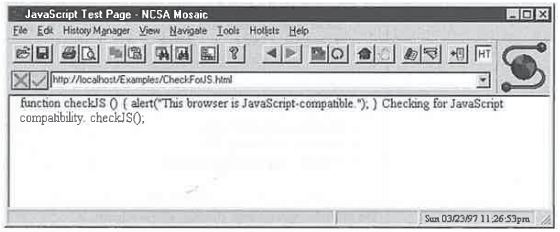


FIGURE 28.4.
The NCSA Mosaic browser is not yet JavaScript compatible. The SCRIPT tags are ignored and the commands are processed as any other text.

Listing 28.4. The contents of the HTML document used in Figures 28.3 and 28.4.

```
<HTML>
<HEAD>
<TITLE>JavaScript Test Page</TITLE>
<SCRIPT>
function checkJS () {
    alert("This browser is JavaScript-compatible.");
}
</SCRIPT>
</HEAD>
<BODY>
Checking for JavaScript compatibility.
<SCRIPT>
checkJS();
</SCRIPT>
</BODY>
</HTML>
```


A noncompatible browser ignores the <SCRIPT> tags and displays the JavaScript commands as any other text. For a document including any length of JavaScript, the result produces a screen full of commands and characters otherwise unintelligible to the user.

Hiding Scripts from Incompatible Browsers

To prevent an older or noncompatible browser from incorrectly processing your JavaScript code, you must use HTML comment tags correctly.

To hide your JavaScript, you must nest a set of HTML comment tags inside the <SCRIPT> tags. A JavaScript comment tag (two forward slashes) must appear just before the closing comment tag to prevent it from being processed as another line of JavaScript and causing a syntax error. (See Figure 28.5.)

```
<SCRIPT LANGUAGE="JavaScript">
<!--  Note: An opening HTML comment tag.
...statements...
// Note: A JavaScript comment tag (two forward-slashes),
// followed by a closing HTML comment tag.
// -->
</SCRIPT>
```

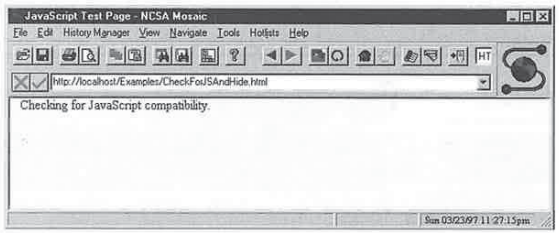


FIGURE 28.5.
NCSA Mosaic with the additional HTML comment tags displays the non-JavaScript content and passes over the part it can't interpret.

With proper placement and usage, JavaScript can add vital functionality to your HTML pages without interfering with the capability of noncompatible browsers to interpret the document. Remember, however, that if your page depends on JavaScript for including crucial information or operability, it shows common courtesy to warn users and, if possible, supply a generic version of the document.

Placing Scripts on the Page

The <SCRIPT> tag can appear in either the HEAD or BODY section, although its placement will determine when and how the script is used.

If placed in the HEAD portion of the document, the script is interpreted before the page is completely downloaded. (See Listing 28.5 and Figure 28.6). This order works especially well for documents that depend on functions. The script is loaded and ready before the user has a chance to interact with any event that actually invokes the function. (See Figure 28.7.)

Listing 28.5. Placing functions in the HEAD portion of the document ensures that they are interpreted and ready for use before the rest of the document can access them.

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="javascript">
function printMessage(msgStr) {
    document.writeln("<HR>");
    document.writeln(msgStr);
    document.writeln("<HR>");
}
alert("The function is loaded.")
</SCRIPT>
</HEAD>
<BODY>
Welcome to the body of an HTML page.
<SCRIPT LANGUAGE="javascript">
printMessage("I just called a function from the body.")
</SCRIPT>
</BODY>
</HTML>
```

FIGURE 28.6.
The initial screen displayed by Listing 28.5. Note that the alert box with our message has appeared, but the text contained in the BODY portion of the document has not.

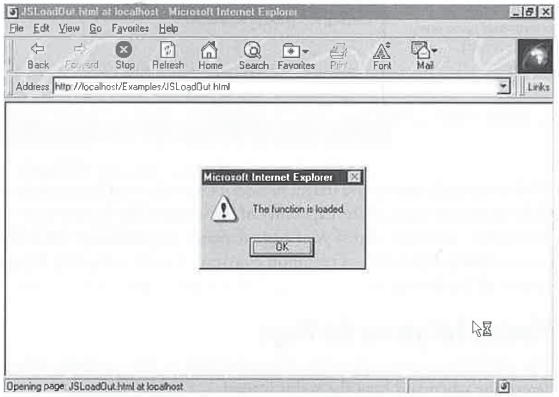
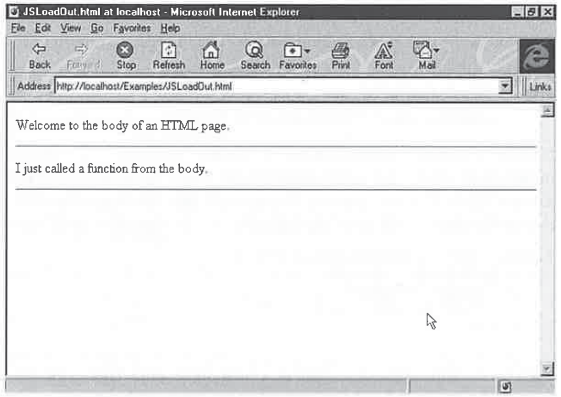


FIGURE 28.7.
The rest of the display generated by Listing 28.5 shows the HTML text in the body of the page and executes the function that was defined in the head.



Using JavaScript in Your Web Page

Many uses exist for JavaScript, and more continue to appear all the time as developers experiment with the possibilities opened with interactive HTML.

This section presents a few uses and examples to get you started.

SCROLLING TICKER DISPLAYS

A JavaScript ticker, which scrolls a message across the status bar, is absent from this list of applications. With Internet Explorer's <MARQUEE> tag and the availability of multifeatured Java ticker applets, the JavaScript ticker no longer provides an efficient use of browser capability.

Validating Forms

Using CGI scripts to validate forms wastes precious user time and server time to conduct a process that is now easier and faster on the client's computer. The time required for client/server communication is reduced, along with the lengthy development necessary for CGI scripts.

With its capability to interact with form elements, JavaScript seems ideally suited to validate information directly on the HTML page. This setup localizes the process and takes advantage of the under-utilized client machine. Checking information on the client side also makes it much harder for users to send incompatible or damaging data to the server.

Several methods exist to implement form validation, and most include adding a JavaScript function as the action of a Submit button. The HTML definition of the Submit button could look like this:

```
<INPUT TYPE="BUTTON" NAME="SUBMIT" VALUE="SUBMIT"
       onClick="checkInformation(this.form)">
```

`checkInformation` is a function that provides verification for the form to ensure that the information meets CGI script expectations. If not, it should return to the document without submitting the form contents to the server. It can also return focus to the offending items. If everything passes inspection, the function can also use the `submit` method.

```
function checkInformation(thisForm) {
    ...validation statements ...;
    if (validationPassed) {
        thisForm.submit(); }
    return;
}
```

Each form element becomes part of a `form` object with JavaScript. By using the name of the form as the name of the object, you can access each of the elements. If a name is not used, you can also use the `forms` array. The first form on the page is `forms[0]`, the next is `forms[1]`, and so on.

For an example, look at the following form definition:

```
<FORM NAME="validation">Enter your user name and identification in the boxes.<BR>
Your name: <INPUT TYPE="text" NAME="userName" VALUE=""><BR>
User ID: <INPUT TYPE="text" NAME="userID" WIDTH="9" VALUE=""><BR>
<INPUT TYPE="button" NAME="button" VALUE="Submit" onClick="checkID(this.form)">
</FORM>
```

Each element in this form is represented in JavaScript, as follows:

```
document.validation.userName
document.validation.userID
document.validation.button
```

The last element, a button, includes an event handler that calls the function `checkID` with the current form as the argument. Note that you don't need the name of the form in the call because the contents are passed as an argument in `this.form`. In the function shown in Listing 28.6, the form is referred to as the name of the argument, `formID`.

Listing 28.6. A function that checks to make sure the length of two form elements are correct before submitting the form.

```
function checkID(formID) {
    val validUser = true;
    val validID = true;
    if (formID.userName.length != 10) {
        validUser = false;
        formID.userName.value = "Invalid Entry"; }
    if (formID.userID.length != 9) {
        validID = false;
        formID.userName.value = "Error"; }
    if (validUser && validID) {
        formID.submit(); }
    else {
        alert("Please try again."); }
}
```


To understand the function, you work through it section by section. First, two Boolean variables are initialized. These flags indicate whether or not the validation has been passed when it comes time to check at the end of the function.

Next, the length of a form element named `userName` is checked. If the value doesn't equal (represented in JavaScript by `!=`) 10, the valid flag is set to false, and the form element receives a new value, which is reflected immediately on the page.

The same process is repeated for the next form element, `userID`. At the end, if both flags are true (logical `and` is represented by `&&`), the form is submitted using the `submit` method. If either or both of the flags are false, an alert screen appears.

Random Numbers

JavaScript includes a method of the `Math` object that generates random numbers, but in its current form, it only works on UNIX machines. Another way of generating somewhat-random numbers exists using a user-defined function instead of the built-in method. It is referred to as a calculated random number, and can reveal its biases and true nonrandom nature if used repeatedly over a short period of time.

To ensure compatibility for a script across platforms, any script depending on random numbers shouldn't depend exclusively on the `random` method, and, instead, should rely on a generated number created by a function similar to the following random number function:

```
function UnixMachine() {
    if (navigator.appVersion.lastIndexOf('Unix') != -1)
        return true
    else
        return false
}

function randomNumber() {
    if UnixMachine()
        num = Math.random()
    else
        num = Math.abs(Math.sin(Date.getTime()));
    return num;
}
```

If the client machine has a UNIX base, `randomNumber` will use the built-in function. Otherwise, it generates a number between 0 and 1 by generating a sine based on the time value.

JAVASCRIPT TIME

Time in JavaScript is measured as the number of milliseconds elapsed since midnight on January 1, 1970, and is accessed by using the `Date` object or an instance of it. The `Date` object is dynamic, ever changing with the time. An instance of the object returns a static value, depending on the current value of `Date` or the date parameter passed to it.

The time is based on the client machine, not the server. One idiosyncrasy occurs in JavaScript's representation of time elements. The `getMonth` and `setMonth` methods both return a value from 0 (January) to 11 (December). When using these two methods, make sure to convert to the 1–12 system the rest of the world recognizes, as in the following example:

```
var birthday1 = new Date(96,1,11);
document.writeln(birthday1.getMonth()); //Returns a 1 (February)
var birthday2 = new Date("January 11, 1996 06:00:00");
document.writeln(birthday2.getMonth()); //Returns a 0 (January)
```

If you need a constant stream of random numbers, a sine wave pattern will become evident. In this case, it becomes necessary to show some variation in the process by adding more variation into the calculation. You can do this by substituting a different computation (`cos`, `log`) at various intervals of time.

Status Bar Messages

With event handlers and the `window.status` property, JavaScript enables your browser to display custom messages in the status bar that respond to user actions. One of the most popular implementations is a descriptive line for hyperlinks:

```
More information is available from
<A HREF="http://www.microsoft.com"
onMouseOver="window.status='The Microsoft Home Page';return true">
Microsoft Internet Explorer</A>.
```

One problem with the `status` property is that it becomes the default message until a browser-generated message overrides it, or `status` is set to a different value. In the preceding example, `The Microsoft Home Page` remains in the status bar until another message preempts it.

Working around this problem requires the use of a timer. After the mouse is passed over the hyperlink, the message displays, but only for a short time, after which the status bar is reset to a blank display. This setup requires two functions—one to write the message and set the timer, and one to erase the message.

```
timerLength = 1.5;
function writeStatus(str) {
    timeoutVal = timerLength * 1000;
    setTimeout("clearStatus()",timeoutVal);
    window.status = str;

}
function clearStatus() {
    window.status = "";
}
```


This method of generating status bar messages requires more lines of code, but results in a cleaner operation for custom hyperlink messages. The message appears for the number of seconds assigned to `timerLength`, after which the `clearStatus` function is called to write a null string to the display. To invoke the new method, use the following example:

```
<a href="http://www.microsoft.com/"
onMouseOver="writeStatus('Microsoft Home Page'); return true;">
Microsoft</A>
```

Another possibility for this method of generating status bar displays includes making a copy of the old value of `window.status` and restoring it when the timer expires.

Controlling Browser Behavior

One of the important and powerful capabilities of JavaScript is controlling various aspects of browser behavior and appearance. This feature comes in handy for implementing demonstrations and tours by adding the capability to spawn new browser windows with controllable levels of functionality.

The command syntax to create a new browser window is

```
windowVar = window.open("URL", "windowName" [, "windowFeatures"])
```

`windowVar` represents the name of a variable that holds the information about the new window.

`URL` refers to an address for the contents of the new window, and can be blank.

`windowName` represents how the window will be referred to in frame and window references.

`windowFeatures` provides a list of the individual features of the browser that should be included or excluded. If blank, all features are enabled. If only some features appear, any unmentioned features are disabled.

To include a feature, use the syntax *`windowFeature`*`=yes` or *`windowFeature`*`=1`. Conversely, to disable a feature, use *`windowFeature`*`=no` or *`windowFeature`*`=0`.

The features include `toolbar` for the row of buttons at the top of the screen, `status` for the status bar at the bottom, `scrollbars` for the buttons and slides to control the part of the document viewed, `resizable` for user control over the size of the browser, and `width` and `height` in pixels for the initial size.

To open a plain window with hotlink-only navigation, use this code:

```
//Note: Setting one feature automatically sets all non-mentioned features to false.
window.open("URL", "windowName", "toolbar=no")
```

JavaScript Resources on the Internet

JavaScript is used and talked about on the Internet frequently, making the Internet one of the first stops for information. You can find up-to-the-minute information on current implementations, bugs, workarounds, and new and creative uses.

Netscape

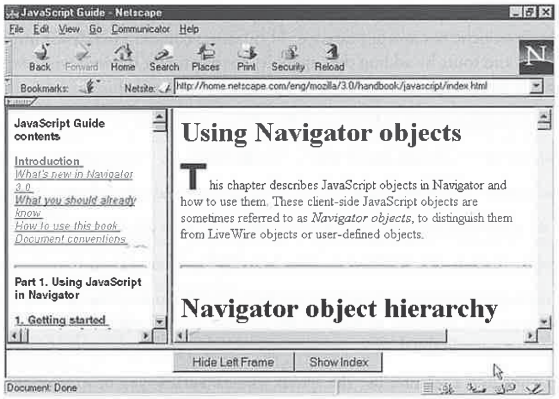
One of the first stops you make should be the home of the people who developed and implemented JavaScript. The Netscape site provides a place to look for new developments and documentation about JavaScript features.

The complete JavaScript online documentation (shown in Figure 28.8) also appears here. The online manual is also available in a ZIP file that you can download.

You can access Netscape at `http://home.netscape.com/`.

The manual is located at `http://home.netscape.com/eng/mozilla/3.0/handbook/javascript/index.html`.

FIGURE 28.8.
The Netscape site has an online manual on JavaScript, including objects, methods, and event handlers. It is also available for download.



Gamelan

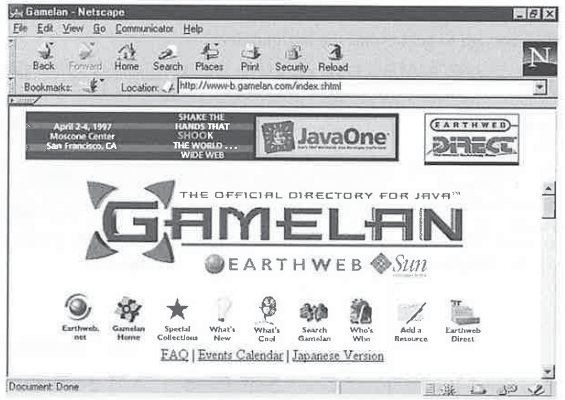
This site provides one of the best places for examples of what other people are doing with JavaScript on the Web. Though Gamelan is more geared towards Java, it also includes one of the largest listings of JavaScript sites found anywhere on the Web. By perusing the examples here (as shown in Figure 28.9), you can gain insight from others who have gone before you.

You can access Gamelan at `http://www.gamelan.com/`.

A Beginner's Guide to JavaScript

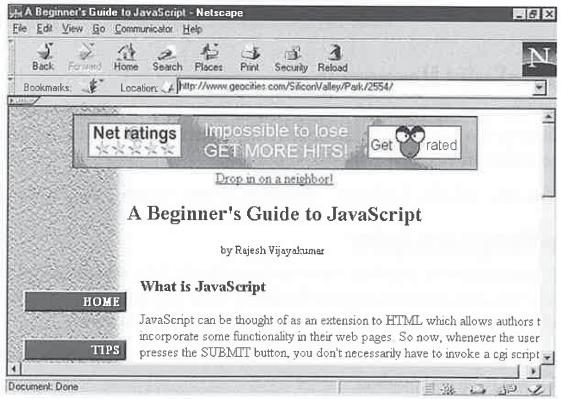
The JavaScript Index (shown in Figure 28.10) has a collection of real-life JavaScript examples and experiments, including a list of Web pages that illustrate some of JavaScript's features and capabilities. It includes tips and code snippets that are available for use by the author in other Web pages.

FIGURE 28.9.
The Gamelan site offers one of the premier sites for JavaScript resources on the World Wide Web.



You can access the Beginner's Guide at `http://www.geocities.com/SiliconValley/Park/2554/`.

FIGURE 28.10.
The JavaScript Index has a solid source of examples and successful experiments from developers who are making JavaScript one of the fastest growing languages on the Internet.



Voodoo JavaScript Tutorial

The Voodoo JavaScript Tutorial (shown in Figure 28.11) contains a set of lessons covering the various aspects of including JavaScript on your Web pages. This project continues to evolve with new lessons added periodically, so you may want to check back periodically to see what's new.

You won't find much in the way of advanced material here, but what you find gives you more than enough to get beyond the beginner level.

You can access the site at `http://rummelplatz.uni-mannheim.de/~skoch/js/index.htm`.

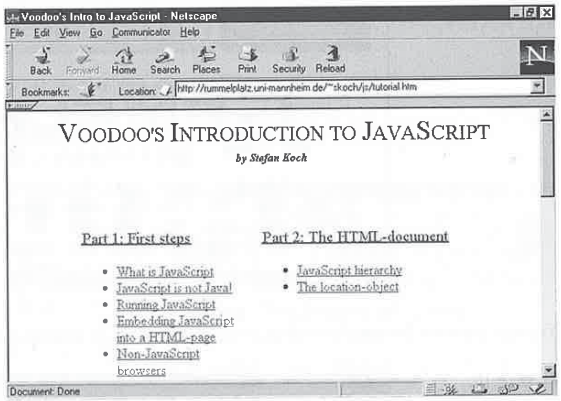


FIGURE 28.11.
The Voodoo JavaScript tutorial provides a good place to pick up the basics of working with JavaScript, in addition to other HTML features such as frames.

JavaScript Newsgroup

This group is frequented by many of the JavaScript gurus and provides an excellent source of information, tips, and workarounds. A high level of activity occurs and the threads move quickly, so make sure to check every day, if possible.

You can reach the JavaScript Newsgroup at `news://comp.lang.javascript`.

netscape.navigator

It never hurts to have a direct line monitored by the folks who developed JavaScript at Netscape, and netscape.navigator remains the closest thing to that line. JavaScript topics are definitely in the minority in this group, but you can find them if you look.

Note the different news server. The title implies it's secure, but it seems to be readily available for browsing and posting.

It's address is `news://secnews.netscape.com`.

Summary

JavaScript adds new functionality and interactivity to HTML pages that in the past you could only attain through learning CGI scripting languages such as Perl. By switching the bulk of interactive behavior to the client side, it has also improved the perceived speed of World Wide Web sites as seen by the user.

Although it does have its nuances and idiosyncrasies, taking the time to learn JavaScript will pay off in supporting Web pages with dramatically improved features and functions that users will want to revisit again and again.

CHAPTER 29

Integrating Java Applets and HTML

by Rick Darnell

IN THIS CHAPTER

- Getting to Know Java *568*
- How Applets and Applications Are Different *571*
- Creating Java Applets *574*
- Using an Applet on a Web Page *578*
- Applets for Fun and Utility *582*
- Applet Sources on the Web *588*

Java is one of the hottest topics on the World Wide Web, and for good reason. It offers expanded portability for Web content, including sound and animation, without the use of plug-ins or other helper applications, and independent of host hardware. In this sense, Java has helped promote a change in the way page developers think about content on the World Wide Web, similar to the way the World Wide Web changed the way people think about the Internet.

Java holds a great deal of promise for the World Wide Web, and computers in general, because it provides a solution to the problem of incompatible platforms. Internets and intranets are no longer expected to include similar or directly compatible machines (all UNIX, all Macintosh, or all PC). Because it has a neutral architecture, the same application written in Java can be used by anyone on the network, without concern for what kind of machine the developer used.

This is the same as HTML—a Web page will work on a PC just as well as UNIX. But, Java is a programming language capable of much more than a Web page. Although it's virtually impossible to build a spreadsheet program with HTML, it's very possible with Java.

For standalone applications, Java's object-oriented structure provides an easy method to upgrade. The class for the upgrade or extension of the application is downloaded into the appropriate class library; then you can run the updated features.

With its modeling capabilities, Java represents a good choice for implementing advanced Web capabilities and content, such as virtual reality sites or Web crawlers powered by intelligent agents.

Getting to Know Java

Java is an object-oriented programming language developed by Sun Microsystems, Inc. Although not initially conceived as a way to expand the interactivity and capability of Web pages, it didn't take long for people to see how the platform-independent nature of Java was an ideal fit with the nature of the Internet.

In the past, when an author developed a page with special content beyond the constraints of HTML, an important decision had to be made—either use helper applications or shift the necessary processing to the server. The first solution meant that some content would be inaccessible to some users if they didn't have the helper application or if a helper was unavailable for their system. The second solution meant excluding some content because inherently slow modem lines made animation and sounds unworkable over normal network connections.

Enter Java from Sun. By utilizing a key feature of Java—platform independence—Java applets can implement sound, animation, and other user interactivity, regardless of platform. You still have to have a compatible browser for your machine, however. Java is currently supported by at least three browsers—HotJava from Sun (which is written in Java, also), Netscape Navigator 2.0 or later, and Microsoft Internet Explorer 3.0 or later.

What Is Java?

According to Sun Microsystems, "Java is a simple, robust, secure, object-oriented, platform-independent, dynamic programming environment."

At first, all of this Java talk can sound like a lot of voodoo. After you strip away the hype, however, it's easy to see how Java works effectively for implementing simple solutions to potentially complicated challenges in a distributed environment.

Platform independence is probably one of Java's most important features. One compiled piece of Java code can run on any platform with a Java compiler. Currently, the list of platforms includes Windows 95, Solaris, and Macintosh, but that list should grow significantly in the near future. By its very nature, Java does not contain any "implementation-specific" syntax. This format means a byte is an 8-bit integer and a float is a 32-bit IEEE 754 floating-point number, no matter where the applet runs.

Java was designed with C and C++ programmers in mind. C and C++, however, had many hard-to-understand, rarely used features that Java developers felt caused more pain than benefit. Important functions that come standard to every program, such as memory allocation and pointer arithmetic, are automatically handled by objects in the Java system without needing any acknowledgment from the programmer.

WHAT DOES *Object-Oriented* MEAN?

Object-oriented, probably one of the most overused and confusing terms in computer lingo, really has a simple and easy-to-understand meaning. It facilitates creating clean and portable code by breaking software down into coherent units.

In other words, object-oriented programming focuses on the ways of interacting with data, rather than the programming language. For example, if you're going to mow the lawn, are you going to be concerned about starting the lawnmower or about the type of socket used to install the spark plug? In object-oriented programming, the focus is on the lawnmower.

Objects become the basic building blocks of the application. Because of their modularity, an object can change without requiring major revision of the other program elements.

One benefit of the object-oriented code is the dynamic nature of the resulting programs. A programmer can use inherited interfaces—a set of methods without instance variables or implementation—to update a class library and not worry about affecting the capability of the rest of the program to interact with it.

In addition, Java is relatively small. Because of the need to run on a wide variety of platforms, Java applications tend to be smaller than the multimegabyte applications that predominate the marketplace. The overhead to run a Java program includes 40KB for the basic interpreter and classes, plus an additional 175KB for the standard libraries and threading.

Java programs must be inherently reliable because any piece of Java byte must be capable of running on any platform. For this reason, a great deal of emphasis is placed on checking for bugs and problems early in the development process, beginning with basic language implementation.

The Java compiler checks for a wide variety of errors beyond basic syntax, including type casting and method invocations. If you make a mistake or mistype, chances are good that your mistake will be flagged by the compiler, which is a far better place than by the interpreter at runtime.

Java and Security

Running in a distributed environment, such as an intranet or the World Wide Web, requires safeguards for client computers; a potentially hostile piece of code can do a great deal of damage by erasing files, formatting disks, and creating other types of damage. Given the way applets are implemented—automatic load and run—you need to ensure the integrity of any piece of code distributed to a broad and uncontrolled audience. Java uses three security procedures to make the end user safe from malicious attacks:

- Byte code verification
- Memory layout control
- File access restrictions

Byte Code Verification

After a piece of Java code is loaded into memory, it enters the interpreter, where it is checked for language compliance before the first statement is executed. This process ensures against corruption or changes to the compiled code between compile time and runtime.

Memory Layout

Next, the memory layout is determined for each of the classes, preventing would-be hackers from forging access by deducing anything about the structure of a class or the machine it's running on. Memory allocation is different for each class, depending on its structure and the host machine.

File Access Restrictions

After that, the interpreter security continues to monitor the activity of the applet to make sure it doesn't access the host file system, except as specifically allowed by the client or user. You can extend some implementations of this specific feature to include no file access.

Although no system can guarantee 100 percent security, Java goes a long way to ensure the protection of client systems from its applets.

How to Use Java Now

The simplest way to implement Java with your HTML Web pages is through embedding applets. A wide variety of applets are already available for inclusion, including the animation applet included with the Java Development Kit, and a plethora of "ticker" display applets.

WATCH WHICH BROWSER

Although spreading quickly, Java applets do not come with all browsers. Netscape Navigator, Microsoft Internet Explorer, and Sun HotJava support Java applets. NCSA Mosaic added Java compatibility to its wish list for future upgrades to its product, but no word has been given on when that might happen.

If your browser is not Java-compatible, the applet section of the HTML page is ignored.

How Applets and Applications Are Different

Java applications work similarly to standalone programs, such as your browser or word processor. They don't require a third-party intermediary, such as HotJava or the applet viewer. Applets require a Java-compatible browser or the applet viewer for viewing. They operate similarly to other objects imbedded in HTML documents, such as Shockwave or RealAudio files, which require assistance to run.

NOTE

The HotJava browser developed by Sun is a Java application that was written and implemented entirely in the Java language.

Applets run on a host system; this fact makes them especially suspect and leads to several key security restrictions. Applets have limited capability to interact with the host platform. An applet cannot write files or send files to a printer on the local system. In addition, it cannot read local files or run local applications. Although no system is 100 percent secure, Java goes to great lengths to ensure the integrity of applets generated under its banner.

Java is not bulletproof, however. As quick as it was proclaimed "secure," a dedicated group of programmers went to work to find security holes. And they found them. Through cooperative efforts between Sun, Netscape, Microsoft, and others, these are being corrected, but it's still a dangerous world. There are reports of "black Java" applets that are hostile enough to format system drives and pass secure information across the Internet.

As discussed in the introduction, the compiled byte code is checked extensively for illegal operations and is verified again on the host system before the applet is run. Although these security features limit the scope and capabilities of an applet, they also help ensure against "Trojan horse" viruses and other shenanigans by less-than-scrupulous programmers.

With all of the security features built in, you don't want to implement word processors, spreadsheets, or other interactive applications in Java applets. If you require these programs, consider building a full Java application, which does not contain the security restrictions of an applet.

JAVA APPLETS AND JAVASCRIPT

It has been said a million times, but if you have just started using Java, it bears repeating. Java isn't JavaScript. JavaScript isn't Java.

Java, in applet or application form, is a compiled language with classes and inheritance. HTML pages can include a reference to Java applets, which are then downloaded and run when a compatible browser finds the tag.

JavaScript is an object-based, client-side scripting language developed by Netscape, but it does not include classes or inheritance. JavaScript exists on the HTML page and is interpreted by a compatible browser along with the rest of the page.

Although they share some common syntax and terminology, the two work differently and have different uses. Confusing Java and JavaScript only leads to a steeper learning curve.

Java Applet Viewer

During applet development and testing, sometimes it's easier to bypass the unnecessary overhead of a browser. If your browser doesn't support applets, you still need a way to view the applets. At this point, the Java Applet Viewer (shown in Figure 29.1) comes in handy.

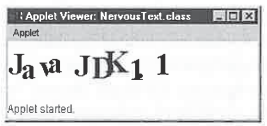


FIGURE 29.1.
The Java Applet Viewer enables the programmer to view embedded Java applets, such as NervousText, without using a browser. Only the embedded applet is displayed; the rest of the HTML around the applet is not even acknowledged.

Using the Applet Viewer

The Applet Viewer searches the HTML document for the `<APPLET>` tag. Listing 29.1 is an example of an HTML document that contains an `<APPLET>` tag.

For more information on the `<APPLET>` tag, see the "Using an Applet on a Web Page" section, later in this chapter.

Listing 29.1. A simple HTML document containing an <APPLET> tag.

```
<HTML>
<HEAD>
<TITLE>The animation applet</TITLE>
</HEAD>
<BODY>
<APPLET CODE="Animator.class" WIDTH=460 HEIGHT=160>
<PARAM NAME=imagesource VALUE="images/beans">
<PARAM NAME=endimage VALUE=10>
<PARAM NAME=pause VALUE=200>
</APPLET>
</BODY>
</HTML>
```

Using the information contained within the tag, the Applet Viewer opens a window on the user's system, and then runs the applet. Other HTML information on the page is ignored; only the applets appear.

The Java Applet Viewer is distributed with the Java Development Kit and is found in the same directory as the Java compiler and interpreter. To run the Applet Viewer, use the following steps:

1. Create a document that references your applet with the appropriate tags and parameters. (See Listing 29.1 for an example.)
2. From a command line prompt, type `appletviewer [path/]filename.html`.

   If the Applet Viewer launches from the same directory as the HTML document, you don't need the pathname. Otherwise, the path is relative to your current location in the directory structure. The extension `.htm` is also valid for the viewer.
3. Any applets found in the HTML document are loaded and run, with each applet in its own instance of the Applet Viewer.
4. Although you cannot change the initial parameters contained within the HTML page from the Applet Viewer, you can start the applet from the beginning by choosing Applet¦Restart from the menu. To load it again from memory, select Applet¦Reload.
5. Leave the applet by choosing Applet¦Quit.

TIP

The Applet Viewer Reload function will not work if the application was launched from the same directory as the HTML document and classes. For applets, create a subdirectory from your class directory called `HTML` and place all of your classes and HTML files in it. Call the Applet Viewer from the parent directory by using `appletviewer html\filename.html`. This way, you can make changes to the applet, compile it, and use the Reload function to see your changes.

Creating Java Applets

Creating Java applets is easier if you already have a background in programming. With Java's tight structure, the basic format of an applet is fairly straightforward. You walk through an example here.

TIP

You can access online tutorials and documentation for Java and object-oriented programming from the Sun site, `http://java.sun.com/`.

Applet ABCs

At its simplest, an applet consists of two parts—the class declaration and a paint method. The following snippet contains a breakdown of the common elements for any applet:

```
Import java.applet.Applet;
import java.awt.Graphics;

public class MyApplet extends Applet {
    public void paint (Graphics g) {
       your statements here;
    }
}
```

The first line includes a copy of the `Graphics` class from Java's Abstract Windowing Toolkit (AWT), which contains the methods needed for putting graphics, including text, lines, and dots, on the browser screen. This line may also be represented as `import java.awt.Graphics` if more than the `Graphics` class will be used.

Second, the actual applet is declared. It is `public`, meaning it is available to any other class, and it is a subclass of Java's Applet class, which provides the behavior necessary for interaction with the host browser.

The third section defines a method called `paint`, which the Java interpreter looks for to put the information on the screen. It is public to the class, and `void` indicates it does not return a value when it is completed. Its one parameter is an instance of the `Graphics` class imported on the first line of the program, which is referred to as `g`. This reference could just as easily be called `bob` or `hammer`, but `g` is the commonly used convention.

Displaying with `paint`

Now that the applet is defined, you need to make it do something. For the `paint` method, include the following line:

```
g.drawString("Hava a nice day.",50,25);
```

After compiling the code and inserting it into an HTML document (as shown in the "Using an Applet on a Web Page" section, later in this chapter), you get something that looks like the message shown in Figure 29.2.

COMPILING AN APPLET

To convert your source code into a usable class, type `javac MyApplet.java` at the command prompt. If any errors are reported, check your spelling and syntax and try again.

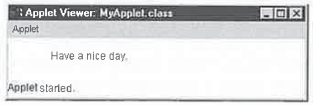


FIGURE 29.2.
MyApplet displays a simple message on the screen.

Of course, applets can do much more. The text can look better if some other AWT classes are included. First, you need the classes that control the font and display color:

```
import java.awt.Font;
import java.awt.Color;
```

After the class declaration, create a variable to hold a new setting for the text:

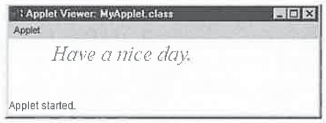
```
Font f = new Font("TimesRoman",Font.ITALIC,24);
```

After the `paint` method declaration, use the `Graphics.set` methods to set the display before writing to the screen:

```
g.setFont(f);
g.setColor(Color.red);
```

With this extra bit of effort, the applet looks like the one in Figure 29.3.

FIGURE 29.3.
MyApplet displays in a larger font in red after some minor revisions to the code.



Again, this example is limited. The addition of a parameter to control the string would make it more useful to the HTML author. After the class declaration, declare the message as a variable:

```
String message;
```

A new method is also required to initialize the value of `message`.

APPLET ACTIVITIES

In addition to `paint`, four major activities exist in the life of an applet. If any are omitted, default versions are provided in the `Applet` class. This setup is called *inheritance*. Providing new methods in the applet is called *overriding*.

The first activity is *initialization*, accomplished with the `init` method: `public void init() {...}`. This activity occurs once, immediately after the applet is loaded. Initialization includes creating objects, setting graphics, or defining parameters. It can only happen once in the applet's life.

The second activity is *starting*, accomplished with the `start` method: `public void start() {...}`. After initialization, activity begins. This activity can also happen if a user activity stopped the applet. Starting can happen many times in the life of an applet. The `paint` method is invoked somewhere in this method.

The next activity is *stopping*, accomplished with the `stop` method: `public void stop() {...}`. This activity can be an important method to include because, by default, the applet continues running and using system resources, even after the user has left the page with the applet. Like `start`, stopping can occur many times in the course of execution.

The last activity is *destroying*, accomplished with the `destroy` method: `public void destroy() {...}`. Destroying occurs when an applet throws out its own garbage after completing execution—when the applet is no longer needed or the user exits the browser. Java provides adequate coverage in this department, so you don't need to override this method unless you want to return specific resources to the system.

To initialize the message parameter, the `init` method for the applet must be overridden:

```
public void init() {
    this.message = getParameter("message");
    if (this.message == null) {
        this.message = "Your message here."; }
    this.message = "A note from Java: " + this.message;
}
```

This method retrieves the value of the parameter in the HTML document. If a parameter named `message` is not found, the value is null and `message` is set to the default string.

TIP

Java is case sensitive for all of its variables, even when passed back and forth as parameters. Remember, a `Rose` by another name is not a `rose`.

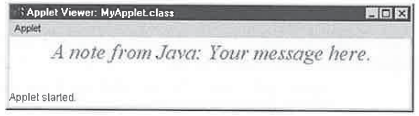
Next you need to update the `paint` method so that it uses the string defined in `init` rather than the literal string in the `drawString` method:

```
g.drawString(this.message);
```

Using the Applet Viewer again generates the results in Figure 29.4.

To place your own message in the applet, add a `<PARAM>` tag to the HTML source containing the applet. For more information, see "Passing Parameters to Applets," later in this chapter. The complete listing for MyApplet appears in Listing 29.2.

FIGURE 29.4.
The default message generated by MyApplet, after checking for a `message` *parameter and finding none.*



Listing 29.2. A simple applet for displaying text onscreen. Note the use of the parameter in the `init` method.

```
import java.awt.Graphics;
import java.awt.Font;
import java.awt.Color;

public class MyApplet extends java.applet.Applet {
    Font f = new Font("TimesRoman",Font.ITALIC,24);
    String message;

    public void init() {
        this.message = getParameter("message");
        if (this.message == null) {
```

continues

Listing 29.2. continued

```
            this.message = "Your message here."; }
        this.message = "A note from Java: " + this.message;
    }

    public void paint(Graphics g) {
        g.setFont(f);
        g.setColor(Color.red);
        g.drawString(this.message,50,25);
    }
}
```

TIP

Listing 29.3 is a sample HTML file that can be used as the basis for inserting or testing applets. Saved in a generic form, it is a very reusable piece of code.

Listing 29.3. A sample of an HTML document that can display MyApplet.

```
<HTML>
<HEAD>
<TITLE>The MyApplet</TITLE>
</HEAD>
<BODY>
<HR>
<APPLET CODE="MyApplet.class" WIDTH=400 HEIGHT=50>
<PARAM NAME=message VALUE="Here I am.">
</APPLET>
<HR>
</BODY>
</HTML>
```

Using an Applet on a Web Page

Using applets on a Web page involves a two-part process. First, you must make sure your classes and related files, such as images and audio clips, appear in a directory accessible to the HTML page. One common location is in a `classes` subdirectory of the HTML documents.

Second, the `<APPLET>` tag that refers to the class must be inserted in the Web page, along with any parameters the applet needs to function.

All About the <APPLET> Tag

The `<APPLET>` tag is used to insert the applet on a page, and it takes the following syntax:

```
<APPLET CODE="appletName.class" [CODEBASE="pathToClass"]
WIDTH=xxx HEIGHT=xxx [ALIGN= ]>
```



```
[<PARAMETER name=parameterName value=parameterValue>]
</APPLET>
```

The required line of code identifies the name of the applet, `CODE`, and the size it will appear on the page.

The optional parameter `CODEBASE` indicates a relative path to the class if it is not stored in the same directory as the HTML file. `ALIGN` works much like the parameter in the `<IMG SRC>` tag by controlling the positioning of HTML text adjacent to the applet's space.

Passing Parameters to Applets

Parameters are used to pass information to an applet about its environment and how it should behave in the current HTML document. Some applets have one method of running and don't accept any parameters. Most, however, contain some user-definable parameters that can be changed.

The `<PARAM>` tag enables you to pass information to the applet. The syntax is this:

```
<PARAM NAME=paramName VALUE=paramValue>
```

The parameter of the name is case sensitive and must exactly match the parameter name in the applet. The value of a parameter is a different matter. All parameters passed from an HTML page to an applet are passed as strings, no matter what it is or how it's formatted on the page. Any conversion to other types (integer, Boolean, date) must happen within the applet itself.

For example, `<PARAM NAME="speed" value=100>` and `<PARAM NAME="speed" value="100">` both pass the string `"100"` to the applet. This cuts a lot of guesswork between the browser and the applet because neither one has to guess what kind of value is being sent. Everything is converted to a string, and the programmer can take care of casting the value to a new type after it enters the applet.

Controlling Applets with Scripts

Controlling a Java applet with a scripting language such as JavaScript is a fairly easy matter, but it does require some knowledge of the applet you're working with. You have to know which methods, properties, and variables in the applet are public. Only the public items in an applet are accessible to JavaScript.

TIP

Two public methods are common to all applets and you can always use them—`start()` and `stop()`. These methods provide a handy way to control an applet when it is active and running.

With this information in hand, start with the <APPLET> tag. It helps to give a name to your applet to make script references to the applet easier to read. The following snippet of code shows the basic constructor for an HTML applet tag that sets the stage for JavaScript control of a Java applet. The tag is identical to the tags you used in previous chapters to add applets, except that a new attribute is included for a name:

```
<APPLET CODE="UnderConstruction.class" NAME="AppletConstruction"
WIDTH=60 HEIGHT=60>
</APPLET>
```

Assigning a name to your applet isn't absolutely necessary because JavaScript creates an array of applets when the page is loaded. However, doing so makes for a much more readable page.

To use a method of the applet from JavaScript, use the following syntax:

```
document.appletName.methodOrProperty
```

TIP

Beginning with version 3.0, Netscape Navigator uses an `applets` array to reference all the applets on a page. The `applets` array is used according to the following syntax:

```
document.applets[index].methodOrProperty
document.applets[appletName].methodOrProperty
```

These two methods also identify the applet you want to control, but the method using the applet's name without the `applets` array is the easiest to read and requires the least amount of typing.

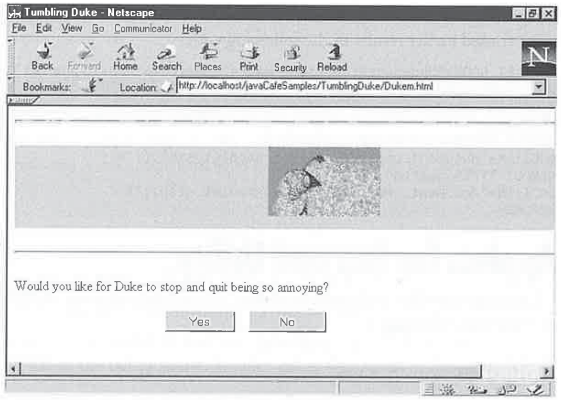
Like other arrays, one of the properties of `applets` is `length`, which returns how many applets are in the document.

The JavaScript `applets` array is currently available only in Netscape Navigator 3.0 or later, and not Microsoft Internet Explorer. This doesn't leave Internet Explorer completely out in the cold–JavaScript can still reference an applet in Explorer using the applet's name.

One of the easy methods of controlling applet behavior is starting and stopping its execution. You start and stop an applet using the `start()` and `stop()` methods that are common to every applet. Use a form and two buttons to add the functions to your Web page. (See Figure 29.5.) The following code snippet is a basic example of the HTML code needed to add the buttons, with the name of the applet substituted for *appletName*:

```
<FORM>
<INPUT TYPE="button" VALUE="Start" onClick="document.appletName.start()">
<INPUT TYPE="button" VALUE="Stop" onClick="document.appletName.stop()">
</FORM>
```


FIGURE 29.5.
One of the simplest methods of controlling an applet is to use buttons that start and stop it.



You can also call other methods, depending on their visibility to the world outside the applet. JavaScript can call any Java method or variable by using a `public` declaration.

TIP

Any variable or method within the applet that doesn't include a specific declaration of scope is `protected` by default. If you don't see the `public` declaration, it's not.

The syntax to call applet methods from JavaScript is simple and can be integrated with browser events, such as the button code snippet just shown. The basic syntax for calling an applet method from Java is

```
document.appletName.methodName(arg1,...,argx)
```

To call the `stop()` method from the `underConstructionApplet` applet within an HTML page, the syntax is as follows (assuming that the applet is the first one listed on the page):

```
document.underConstructionApplet.stop();
```

Here's how you do it with Navigator (again, assuming that the applet is the first one listed on the page):

```
document.applets[0].stop();
```

Integrating the `start()` and `stop()` methods for this applet with the applet tag and button code snippet used earlier results in the following code:

```
<APPLET CODE="UnderConstruction" NAME="underConstructionApplet"
WIDTH=60 HEIGHT=60>
</APPLET>
<FORM>
<INPUT TYPE="button" VALUE="Start"
onClick="document.underConstructionApplet.start()">
<INPUT TYPE="button" VALUE="Stop"
onClick="document.underConstructionApplet.stop()">
</FORM>
```

Applets for Fun and Utility

This section provides a selection of applets available on this book's CD-ROM that you can use on your own Web pages.

Animator

Probably one of the most frequently used applets is the Java Animator Applet (shown in Figure 29.6), which comes with the Java Development Kit and provides a quick-and-easy way to add animation to your Java-powered page. You can find it at `http://localhost/javaCafeSamples/Animator/example1.html`.

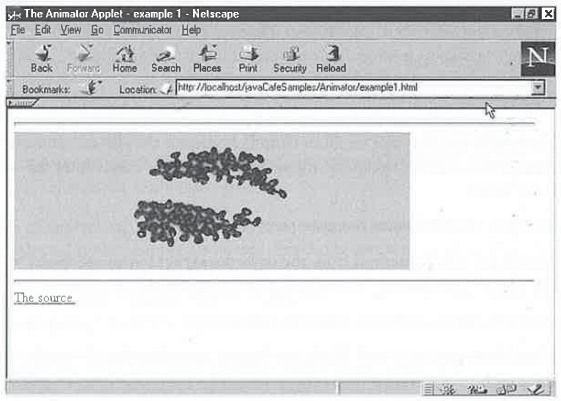


FIGURE 29.6.
The Animator Applet is used to display a series of images with an option for frame-specific sounds and soundtracks.

Animator also supports synchronized sound with the animation, but the sound must use the `Sun.AU` format. No other sound formats are supported yet.

Implementing the applet requires a set of GIF or JPG files containing the images that form the animation.

TIP

Try to keep the size of the images as small as possible. Each image has to be downloaded to the client machine, adding significantly to the time required for the applet to load and run.

A wide variety of parameters exist that control the operation of Animator. Here's the breakdown and syntax (with the explanation following each line of code):

```
<APPLET CODE="Animator.class" WIDTH=number HEIGHT=number>
```

Width should be at least the width in pixels of the widest frame, whereas height should reflect the size of the tallest frame. Smaller values will result in the image being clipped.

```
<PARAM NAME=IMAGESOURCE VALUE="pathInfo">
```

Points to the directory that contains the animation frames. The default directory is the same as the HTML document. By default, the files are named `T1.gif`, `T2.gif`, and so on.

```
<PARAM NAME=STARTUP VALUE="filename">
```

An image that is displayed while the applet loads and prepares to run.

```
<PARAM NAME=BACKGROUND VALUE="filename">
```

An image file for use as a background for the animation.

```
<PARAM NAME=BACKGROUNDCOLOR VALUE="color,color,color">
```

The color for the animation background, represented as an RGB value with a number from 0 to 255 for each of the settings.

```
<PARAM NAME=STARTIMAGE VALUE=number>
```

The first frame in the animation, by default `1`.

```
<PARAM NAME=ENDIMAGE VALUE=number>
```

The last frame in the animation.

```
<PARAM NAME="NAMEPATTERN" VALUE="dir/prefix%N.suffix">
```

A pattern to use for generating names based on `STARTIMAGE`, `ENDIMAGE`, or `IMAGES`.

```
<PARAM NAME="PAUSE" VALUE=number>
```

Number of milliseconds to pause between images default—can be overridden by `PAUSES`.

`<PARAM NAME="PAUSES" VALUE="number¦number¦...">`

Millisecond delay per frame, with each value separated by a vertical bar. Blank uses a default PAUSE value.

`<PARAM NAME="REPEAT" VALUE=true>`

If true, the animation will continue as a loop.

`<PARAM NAME="POSITIONS" VALUE="x@y¦x@y...">`

Screen positions (X@Y) for each frame, represented in pixels and separated by vertical bars. A blank value will use the preceding frame's position.

`<PARAM NAME="IMAGES" VALUE="number¦number¦...">`

Used to define an explicit order for frames, which becomes useful if your frames are out of order or if you want to reverse the sequence (such as "1¦2¦3¦2¦1").

`<PARAM NAME="SOUNDSOURCE" VALUE="aDirectory">`

Indicates the directory with the audio files. The default is the same directory as the class.

`<PARAM NAME="SOUNDTRACK" VALUE="aFile">`

An audio file to play throughout the animation as background music.

`<PARAM NAME="SOUNDS" VALUE="aFile.au¦¦¦¦bFile.au">`

Plays audio files keyed to individual frames.

`<PARAM NAME="HREF" VALUE="aURL">`

The URL of the page to visit when user clicks the animation (if not set, a mouse click pauses/resumes the animation).

Calendar Applet

The Calendar Applet (`http://localhost/javaCafeSamples/Calendar/Calendar.html`) displays a calendar for the current month with a button for each day and a scrolling message across the top. The month is changed via a pair of buttons at the bottom of the applet. (See Figure 29.7.) As a button for a specific day is pushed, the applet displays any events on that day.

```
<APPLET CODE=Calendar.class WIDTH=300 HEIGHT=275>
</APPLET>
```

The events are set using HTML parameters.

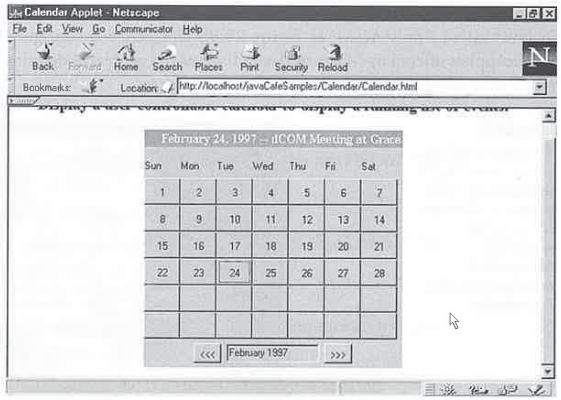
```
<param name=EVENTnum value="mm/dd/yy:Message">
```

First is the parameter name. Each EVENT is numbered sequentially, so the first is EVENT1, the second is EVENT2, and so on. The value of the parameter is in two parts, separated by a colon. The first part identifies the date value for the button in MM/DD/YY format. This is followed by a colon and the text of any string you want to display:

```
<param name=EVENT1 value="02/14/97:February 14, 1997 - Happy Valentine's Day">
```

In this example, EVENT1 corresponds to 2/14/97. Clicking the button for that day results in "February 14, 1997—Happy Valentine's Day" scrolling across the top of the calendar.

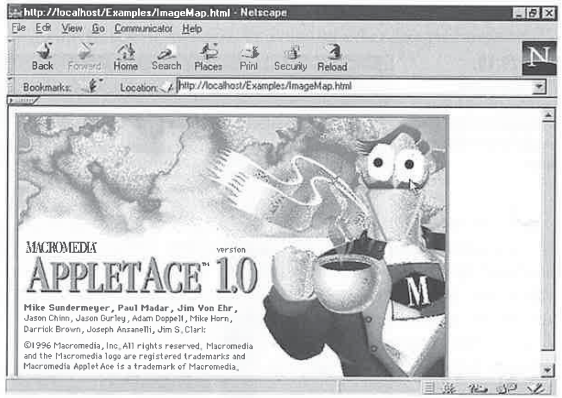
FIGURE 29.7.
The Calendar Applet.



Macromedia ImageMap PowerApplet

The Macromedia ImageMap PowerApplet (`http://www.macromedia.com`) is an example of some of the creative things you can do with Java for client-side image maps. The applet enables you to define hot areas in a GIF image that act as hyperlinks to other Web pages. In addition to the basic hyperlink activity, it can also provide simple animation by replacing portions of the original image with another image, display pop-up text, and produce other effects. (See Figure 29.8.)

FIGURE 29.8.
The ImageMap applet is using the splash screen from AppletAce. Placing the mouse over the Ace's eyes makes them "pop out," instead of their normal appearance.



To make working with ImageMap easier, Macromedia has also developed a handy utility (written in Java) called AppletAce. (See Figure 29.9.) AppletAce is configured to each of the four PowerApplets offered by Macromedia and makes the process of defining hot areas and actions much easier than working directly with parameter tags.

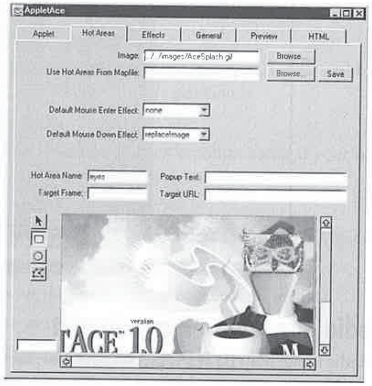


FIGURE 29.9.
AppletAce provides an easy-to-use interface for generating the HTML <APPLET> *and* <PARAM> *tags needed for customizing Macromedia's ImageMap and other PowerApplets.*

Ticker

The Ticker applet (`http://www.sbk-ks.de/~twe/e/ticker.html`) provides one example of the many "ticker tape" applets available. It is one of the improved versions, which has made the extra effort to reduce flicker and provide additional control over the text. (See Figure 29.10.)

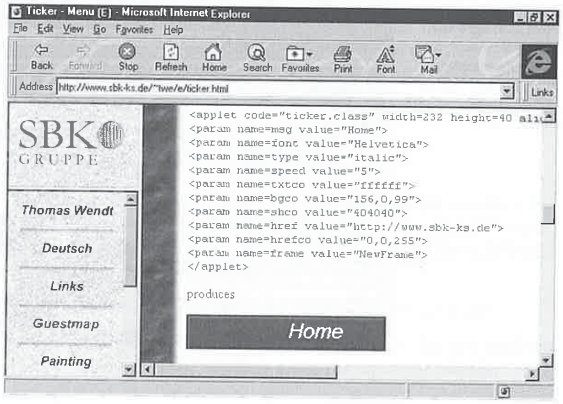


FIGURE 29.10.
The Ticker applet provides a flexible way to display scrolling messages on the browser screen.

The Ticker applet takes a variety of parameters, listed here with an explanation of each following the parameters:

`<PARAM NAME=msg VALUE="string">`

The message to display.

`<PARAM NAME=speed VALUE=number>`

The animation speed, expressed as the number of pixels per 100 milliseconds. The default is `10`.

`<PARAM NAME=txtco VALUE="r,g,b">`

The color of the message, expressed as an RGB value with numbers from 0 to 255. If omitted, the default is black.

`<PARAM NAME=bgco VALUE="r,g,b">`

The color of the background. If omitted, the default appears as light gray.

`<PARAM NAME=shco VALUE="r,g,b">`

The color of the message shadow. If omitted, no shadow appears.

`<PARAM NAME=href VALUE="URL">`

The ticker can also serve as a hyperlink if the user clicks the ticker. A relative or complete URL is legal.

`<PARAM NAME=hrefco VALUE="r,g,b">`

The color of the URL frame. If omitted, the default is blue.

```
<PARAM NAME=start VALUE="yy, mm, dd">
<PARAM NAME=exp VALUE="yy, mm, dd">
```

Dates to start and stop displaying the applet. If the page is viewed outside of these dates, as determined by the host machine, the ticker will not display its message. It will still occupy space on the screen, however. You can use either date parameter by itself.

`<PARAM NAME=exfill VALUE="r,g,b">`

If the local date falls outside of the start and stop parameters, the box becomes filled with this color.

J-Track

J-Track (`http://liftoff.msfc.nasa.gov/home/mission/jtrack/welcome.html`) is an interesting little piece of work that tracks several prominent satellites and the space shuttle when in flight. (See Figure 29.11.) After the applet is loaded and initialized, it displays the position and path of each satellite. The tracking information is updated in real-time, so if you just sit and watch, you can see the position of each target change. When the space shuttle is orbiting the earth doing repairs on the Hubbel telescope or docking with the space shuttle, you can track the progress from your desktop and be the first one on your block to know when to look out your window and wave.

Applet Sources on the Web

Many sources exist on the Web for applets that you can use. Make sure to check the licensing on the applet. Just because an applet appears on a page doesn't mean you can freely use it.

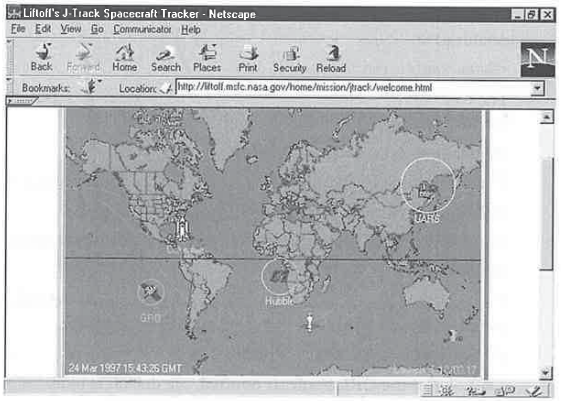


FIGURE 29.11.
J-track uses real-time data from NASA to track the position of several satellites as they orbit the earth, including the Hubbel telescope, Mir space station, and the space shuttle.

JavaSoft

JavaSoft, a subsidiary of Sun Microsystems, handles the Java products. Go to its Web site (`http://java.sun.com/`) first when looking for information, documentation, updates, downloads, and other feedback.

Originally part of the Sun Web site, JavaSoft received its own space to handle the dramatic increase in attention Java has received since its release.

The Java Applet Rating Service (JARS) World

JARS (`http://www.jars.com/`) is an independent organization devoted to promoting excellence in Java applet programming. Developers submit their work to the folks at JARS, who evaluate the applet, give it a rating, and then post the results on their site along with links to the developer's home page and applet example.

The number of applets rated through JARS has continued to grow, so it's a good place to look for some of the best in applet programming.

Gamelan

The Gamelan site (`http://www.gamelan.com/`) shows you what the rest of the world is doing with Java. Links appear here to some of the best applets to date for the viewing, and you can download some for use on your pages. It also includes a page devoted to JavaScript for links devoted to pages utilizing Java's cousin.

Some of the innovative productions found here include animators, tickers, network utilities, and a "Learn to Dance" applet.

alt.lang.java

Although not technically a source for applets, the `alt.lang.java` newsgroup provides a great source of information about Java and its uses. Following the threads can also lead to Java applets and applications, where you can learn from people already making the most of this new language.

Summary

With Java, you can embed customized applications within your Web page that expand the capabilities of HTML beyond its traditional static nature. Applets are available for enabling on-line shopping, database searching, player-to-player gaming over the Internet, and gathering guestbook information (without using CGI scripts).

Anything you do with a traditional application—word processing, spreadsheets, database access, and so on—you can do with Java. And with Java, you approach a write-once, run-anywhere capability. You don't have to provide a separate applet for Macintosh machines, another for PCs, and a third for UNIX. One applet will fit all three.

Java was, in its early days, primarily used for animations, tickers, and other multimedia-type content. Although these take advantage of some of Java's capabilities, you should also consider using Java for database applications, such as user registration and guestbook applications, on-line shopping, and computer-to-computer communication over network lines.

Java has been embraced by the major developers of hardware, including Intel, IBM, and Macintosh, and additional support from software developers, including Microsoft, Netscape, Sun, Macromedia, and Symantec. You can expect Java's use to continue spreading as more and more hardware and software manufacturers pledge support to the language and concepts. Even if you never have the chance to delve deeply into the intricacies of building an applet or application from scratch, an understanding of the basics will help you take full advantage of Java's powerful capabilities.

CHAPTER 30

Integrating ActiveX and VBScript

by Paul Lomax

IN THIS CHAPTER

- The Importance of ActiveX and VBScript 592
- A New Era for Web Pages 592
- Using VBScript with HTML Documents 593
- Adding ActiveX Controls to HTML Documents 605
- The Microsoft HTML Layout Control 615
- Setting Properties of Controls in an HTML Layout 617

In this chapter, you learn about the importance of the latest Microsoft ActiveX and VBScript technologies and how you can use them to create stunning Web applications. You learn how to add VBScript to your Web pages to interface and manipulate HTML intrinsic controls, ActiveX controls, and even the browser itself. You learn how to incorporate ActiveX controls and HTML layout controls to give your Web page the look and feel of a Windows application.

The Importance of ActiveX and VBScript

When Microsoft discovered the Internet in late 1996, the world was to change forever. Too dramatic? Maybe, but at least the world of computing, which is not known for staying put in one place for more than a few seconds, was about to undergo a revolution. Still too dramatic? Not really. This revolution was to be more a renaissance than an advent.

The renaissance is really in the form of centralized processing power. Over the past few years, corporations have been busy upgrading the specification of client machines, decentralizing processing, placing more and more work on the workstation, taking the load from the server and distributing it throughout the organization. However, the upsurge in the popularity of the intranet—with good reason—is, in part, from putting that processing back onto the server through the use of small, self-contained, componentized applications that run on the server, querying databases, and passing no more than a text-based HTML page to the client. But, of course, this is only part of the picture. If this were taken to its logical conclusion, we'd end up with Network Computer and Super Server, which as we all know is not what Bill Gates is aiming for.

So what exactly is Microsoft doing to computing?

Over several frantic months, Microsoft engineers adapted and modified their tried and tested technologies and added Internet capabilities to virtually anything that moved. In fact, more than adding Internet capability, the previous situation of a Web-enabled desktop application is actually reversed; it's now more common to ask whether a product can run as a standalone on the desktop as well as on the Web. (I exaggerate a little, but you get the point.) This change has been brought about by the progressive development of a technology that is now called *ActiveX*.

ActiveX is a way of distributing processing power once again. In fact, such is the flexibility of the technology that the decision about where a control (or script, for that matter) should execute is where it should be—in the hands of the developer. The developer decides—sometimes from experience, sometimes from trial and error—where the application should run. So what does this mean for the Web and, in particular, Web page design?

A New Era for Web Pages

Windows applications have for several years taken advantage of component object model (COM) objects, which are reusable software building blocks that are glued together by developers using languages such as Visual Basic to create complete applications.

COM has recently been given a facelift. Distributed COM (DCOM) enables developers to create an application, some part of which can be on a completely different machine. An extension of this idea is ActiveX. So rather than being a brand new untried technology, ActiveX is the logical progression of the very thing that Windows itself is based upon.

From an HTML developer's point of view, the best thing about ActiveX is that it can be glued into an HTML Web page.

ActiveX controls come in all shapes and sizes; at first glance, some might not even appear to be ActiveX controls at all. Consider, for example, the Forms2 controls that Microsoft includes with the full install of Microsoft Internet Explorer (MSIE). The text box, combo box, list box, and so on are all ActiveX controls. Then there are the third-party controls, everything from database controls to Scratch 'n' Sniff controls. (Okay, so I made that one up.) There are also ActiveX data controls to make the retrieval and display of data from the server that much easier, and ActiveX controls that operate on the server itself, even to the point of controlling the Web server.

Many ActiveX controls are self-contained, but the vast majority require some user interaction; they need some input to process and then output. This interaction is made possible by the glue that goes between the user interface (the browser) and the control. The glue that holds the HTML page and ActiveX control together is VBScript. Although it is a fully functional language in its own right and can be used to write all sorts of applications, its most popular use at the moment is to control the ActiveX controls.

Using VBScript with HTML Documents

When you add VBScript to your Web page, you create an interactive application. Like all programs, VBScript is used to provide an automated solution to a problem. For example, your problem might be figuring out how to communicate with the user quickly and without sending data back and forth between the server and the browser. The solution to this problem could be to write a script that displays a customized message box in response to a particular event, like this:

```
<SCRIPT LANGUAGE="vbscript">
    Sub cmdButton1_OnClick
        Alert "Thank you for clicking this button"
    End Sub
</SCRIPT>
```

Now let's look at the elements you use to create interactive scripted HTML documents.

The <SCRIPT> Tag

New for scripting is the `<SCRIPT>` tag, which does use a closing tag, `</SCRIPT>`.

The `<SCRIPT>` tag has one main element, `LANGUAGE=`. The current values for this are `JavaScript`, `VBScript`, or `VBS`. In Microsoft Internet Explorer Version 3.*x* and later, the default scripting language is JScript; if you don't use the `LANGUAGE=` element, your script will be compiled as

JavaScript; and if you've written in VBScript, all you get for your troubles is an error. To compile as VBScript, therefore, you must start your script with this line:

```
<SCRIPT LANGUAGE="vbscript">
```

TIP

Remember that, as with all HTML elements, the `<SCRIPT>` tag and the script it contains are not case sensitive; `vbscript` is the same as `VBScript` and `vBscRiPt`.

The `<SCRIPT>` tag tells script-enabled browsers to get ready to compile a script from the code that follows, but what if the browser (Netscape Navigator, for example) doesn't understand VBScript? Because you've used the `LANGUAGE=` element, the complete script block is ignored.

But what about browsers that aren't script-enabled at all? Well-behaved browsers will ignore the `<SCRIPT>` tag, but they won't ignore the script it contains and will treat the script like text and show it on the page. It's prudent, therefore, to always enclose your scripts within HTML comment markers, like this:

```
<SCRIPT LANGUAGE="vbscript">
  <!--
    your script goes here
  -->
</SCRIPT>
```

The comment markers are ignored by the scripting engine, so don't worry about your script being overlooked. This way, your Web page won't look like hieroglyphics to someone with an older browser.

You can have as many `<SCRIPT>` tags within one HTML page as you like, or need, or you can consolidate all your subroutines and functions for the page into one script tag block. Furthermore, it doesn't matter where on the page you place your script block, in the `<HEAD>` or `<BODY>`; wherever it is, it will be compiled.

So, you've got your script block, but when is your script going to execute? That depends on what you want the script to do. Here are the three options:

- A script that automatically executes as the page is downloading into the browser
- A script that executes as the result of an action taken by the user
- A script that executes because it has been called by another script

Allowing your script to run automatically as the page is loading into the browser entails a degree of risk, if only because you have lost control over its execution. However, sometimes you need to have a freely running script, such as when you need to print some variable text into the HTML page at a particular place.

CAUTION

Always ensure that scripts that will run as the page is downloading do not reference controls that have not yet been downloaded. In other words, a script that references a control must be placed after the control's definition on the page unless the script is held within an event handler or a subroutine.

To create a script that executes automatically, use the `<SCRIPT>` tag without any event handler or subroutine definition, like this:

```
<SCRIPT LANGUAGE="vbscript">
  <!--
    Document.Write "<CENTER><H3>Hey this is Cool!<H3></CENTER>"
  -->
</SCRIPT>
```

What's an event handler? An event is something that the user causes to happen or that a change in a control generates. A click on a button is an event; typing text into a text box is an event; the page downloading into the browser is an event. The code you write that executes in response to an event is called an *event handler.*

Unfortunately (or fortunately), you can't create your own events; you have to use the events available for the particular control or object. Here's how you define an event handler for an HTML button's `Click` event:

```
<SCRIPT LANGUAGE="vbscript">
  <!--
    Sub myButton_OnClick
        Alert "Hello World"
    End Sub
  -->
</SCRIPT>
```

`myButton` is the name of the control. (See the next section, "The `<INPUT>` Tag," for more information about how to name controls.) `_OnClick` is the event name. Note that the event handler is finished with the `End Sub` statement.

The final way to add scripting is by writing a subroutine, of which there are two types. A subroutine is a standalone piece of code that is called by some other code within your script. There are two types of subroutines: the function, which returns a value to the calling code; and the sub, which, when it's finished, simply hands back execution to the calling code without passing back a value. Here's how you define both:

```
<SCRIPT LANGUAGE="vbscript">
  <!--

    Sub myButton_OnClick
        Call mySubRoutine()
    End Sub
```

```
    Sub mySubRoutine()
        Alert timesTwo(4)
    End Sub

    Function timesTwo(someValue)
        timesTwo = someValue * 2
    End Function

  -->
</SCRIPT>
```

In the preceding code snippet, you can see all three types of scripted routines in action. First, the `OnClick` event handler for the button called `myButton` calls the sub `mySubRoutine`, which passes the value `4` to the `timesTwo` function. The `timesTwo` function multiplies the incoming value (whatever it is) by two and returns the result. In this case, the result is shown in an alert box, which you'll learn more about later in this chapter.

As you've seen, apart from writing scripts that execute on download to the browser, it is most likely that a script will at least start executing as the result of some user action—an event. The next section offers a look at the easiest way to create a dynamic, interactive, scripted user interface using normal HTML form elements.

The <INPUT> Tag

The `<INPUT>` tag will be familiar to anyone who has authored an HTML form. It defines the user interface items that you place within the form, such as the text box or option button.

A list of HTML form controls known as *elements objects* or *intrinsic controls* can be found later in the chapter in the section called "Referencing the `Elements` Objects."

VBScript can be used to interface with these HTML controls, just as easily as it can with ActiveX controls. In fact, you can gain a large advantage simply by adding scripting to your current HTML forms.

To attach script to an HTML form control, you simply need to give the control a name using the `NAME=` element of the `<INPUT>` tag. Be careful, however, to give your controls different names because the scripting engine must have a unique name to reference a control.

There are two methods of attaching a script to an HTML control. The first, and the easiest to read and maintain, is to create an event handler within a `<SCRIPT>` block. Listing 30.1 shows how to place the contents of one text box into another text box at the click of a button.

Listing 30.1. A simple script attached to the click event of a button.

```
<HTML>
 <HEAD>
  <SCRIPT LANGUAGE="vbscript">
  <!--
   Sub cmdButton1_OnClick
       frmForm1.txtText2.Value = frmForm1.txtText1.Value
       Alert "Hello World"
       Status "VBScript in Action"
   End Sub
  -->
  </SCRIPT>
 </HEAD>
 <BODY BGCOLOR="white">
  <CENTER>
   <FORM NAME="frmForm1">
    <INPUT TYPE="text" NAME="txtText1">
    <INPUT TYPE="text" NAME="txtText2">
    <INPUT TYPE="button" NAME="cmdButton1">
   </FORM>
  </CENTER>
</HTML>
```

When the user clicks the `cmdButton1` button (which, by the way, is not the usual `Submit` that you're probably used to using on an HTML form), the `cmdButton1_onClick` event is fired. In the simple script attached to this event, the `Value` property of the text box `txtText1` is assigned to the `Value` property of text box `txtText2`, thus displaying the contents of `txtText1` in `txtText2`. Just for good measure, the message `Hello World` then pops up on the screen in a small alert box, and the message `"VBScript in Action"` is displayed in the status bar along the bottom of the browser.

The other method of attaching code to a control's event is somewhat trickier to write, read, and maintain. Basically, you write the complete event handler within the control's HTML definition. Listing 30.2 is the same example as the one used in Listing 30.1, except the event handler is written inline.

Listing 30.2. A script written inline with the control's definition.

```
<HTML>
<HEAD>
</HEAD>
<BODY BGCOLOR="white">
<CENTER>
    <FORM NAME="frmForm1">
        <INPUT TYPE=text NAME="txtText1">
        <INPUT TYPE=text NAME="txtText2">
        <INPUT LANGUAGE="VBScript" TYPE=button ONCLICK="frmForm1.txtText2.value
➥=frmForm1.txtText1.value
Alert &quot;Hello World&quot;
 Status &quot;VBScript in Action&quot;"
         NAME="cmdButton1">
    </FORM>
</CENTER>
</HTML>
</BODY>
```

As you can see, the `cmdButton1`'s HTML `<INPUT>` definition is now creaking under the weight of compact, yet illegible coding. The `LANGUAGE=` tag is there, as is the `ONCLICK` event, only this time as an `ONCLICK=""`. The complete event handler must be written within quotation marks, which means that any quotation marks used within the event handler must be implicit; in other words, you have to use the `"` HTML code.

More than one inline event handler can be placed in a control's HTML definition to handle different events.

Later in the chapter, you'll see the ActiveX Control Pad in use. This tool enables you to add ActiveX controls to an HTML document with ease and to attach scripts to the control's events. You can also use the ActiveX control pad's Script Wizard to add scripts to HTML form controls, and by default it will generate the inline type of event handler seen in Listing 30.2.

Using VBScript with the HTML Object Model

VBScript is an event-driven language. Unlike scripts that run automatically as the page is downloading into the browser, VBScript programs are launched by an event—for example, when a user clicks a button or changes the value of a text box. Which events you can use to start a script executing depends on which events have been built into the objects that make up the environment in which you are working—in this case, the browser.

NOTE

Until Netscape launches a browser that supports both ActiveX and Active scripting (VBScript and JScript), when I mention the browser for VBScript, I mean Microsoft Internet Explorer 3.*x* (MSIE).

It follows, therefore, that to use VBScript successfully, you must understand the objects that make up the browser—the HTML object model.

A browser application that supports Active scripting must adhere to the HTML object model specification, a standard that allows for the consistent operation of client-side scripting languages such as VBScript.

The HTML object model defines several main objects and the properties, events, and methods associated with those objects. Because these objects expose their properties, events, and methods via a programmable interface, it is possible for scripting languages such as VBScript to interact with the object model by reading and setting an object's properties, responding to an object's events, and calling an object's methods.

The HTML object model is arranged in a hierarchy of objects, with the main browser window at the top and the form elements (such as text boxes) at the bottom. When you reference an HTML object in your script, you must follow the hierarchy; otherwise, you will generate an error.

TIP

The most common error that comes up when you are trying to reference an HTML object's property or event is `Property or Method Not Supported by this Object`. This means that you haven't followed the logical path of the hierarchy and thus have called a property or method that is not recognized by the stated object. You will usually find the solution by starting at the top of the hierarchy and working your way down toward the object you're trying to reference. You will come across an object that either doesn't appear in your reference or isn't being referenced properly.

The Object Hierarchy

To create a reference in your script to a particular HTML object, you must build the reference from the top of the hierarchy downward, using dot notation to separate the objects. Here is a list of the objects:

Object	*Description*
`Window`	The `Window` object is at the top of the tree in the object hierarchy; all other objects are child objects of the `Window` object. In most instances you don't have to specifically use the `Window.` reference in your script because it is implicit in all HTML object script references. Another way to use the `Window` object in your script is through the `Top.` reference.
`Frame`	The `Frame` object is actually an array of `Frame` objects. One way to reference them is by their ordinal numbers in the window, in which case the first frame is referenced as `Frames(0)`, the second frame is referenced as `Frames(1)`, and so on. The other way to reference a `Frame` object is by its name, assuming that you have used the `Name=` parameter when defining the frame in your `Frameset` document.
`Document`	The `Document` is a child object of either a `Frame` or a `Window`. Each `Frame` or `Window` object can contain only one `Document` object. As you might have guessed, the `Document` object represents the actual HTML document itself. You can use the `Document.Write` method to create dynamic HTML pages writing text to the document as it downloads to the browser, even creating completely new HTML documents at the browser from VBScript. After the document has been downloaded into the browser and displayed to the user, its content is fixed.

NOTE

Forthcoming developments from Microsoft will allow you to change the HTML page using VBScript even after it has been downloaded into the browser, giving almost unlimited flexibility and dynamism to HTML.

`Form` Like the `Frame` object, the `Form` object is an array of `Form` objects. A `Form` object can be referenced by its ordinal number in the array, starting with `0` or by its name. You can use the `Form` object to programmatically set the `Method` and `Action` parameters of the form and to submit the form data to the server. If the form and script reside on the same HTML page, you can reference a `Form` object without using a `Window.Document.` prefix; however, if the script and form are on separate documents (for instance, in two separate frames), you must use `Document.` as a prefix to the form reference. See "Scripting with the HTML Object Model" and "Working with Frames," later in this chapter, for more information.

NOTE

For the sake of clarity and understanding of the HTML object model, the little-used `Location`, `Navigator`, and `History` objects have been left out of this discussion.

`Element` The `Element` object is an array of form elements that includes `Text`, `Hidden`, `Password`, `Radio`, `Checkbox`, `Button`, `Submit`, and `Reset`. There is also one other main form element, `Select`, which creates a drop-down or selectable list of options. Although this is not defined using the `<INPUT>` tag as the other HTML form controls are, for all intents and purposes, the `Select` control can be treated like all the other form controls. Again, `Element` objects can be referenced by their names or by their ordinal numbers in the `Elements` array. You must note, though, that all the elements in a single form are placed in the same array; therefore, if you want to reference them by number, you need to code some method of distinguishing which type of element you are currently referencing.

Scripting with the HTML Object Model

To illustrate how you can use the HTML object model in your scripts, here are a few examples of each object. All the following examples assume that the object and the script are within the same window or frame; a separate section, "Working with Frames," has been devoted to the particularly tricky subject of creating scripts that have the object and the script located in different frames.

Referencing the `Window` Object

To display a quick message box to the user, you can call the `Window` object's `Alert` method:

```
<SCRIPT LANGAUGE="vbscript">
   Sub cmdMyButton_OnClick
       Alert "You clicked me?"
   End Sub
</SCRIPT>
```

Note that the `Window` object is implicit and does not need to be referenced. This means that the code

```
Alert "You clicked me?"
```

is the same as

```
Window.Alert "You clicked me?"
```

Referencing the `Document` Object

The most common use of the `Document` object is to create dynamic content via its `Write` method. To use this, simply add a script that will execute automatically on download within your HTML, and the resulting HTML text will appear in the completed document in the browser. Here's a quick example:

```
...
<BODY BGCOLOR="white">
<CENTER>
<H2>Welcome to my Dynamic Website</H2>
</CENTER>
<P>
Today is;
<SCRIPT LANGUAGE="vbscript">
    Document.Write WeekdayName(Now()) & "" & Now()
</SCRIPT>
<P>
...
```

Again, because the use of the `Window` object is implicit, you can simply start your reference to the `Write` method with the `Document` object.

Referencing the `Form` Object

The `Form` object is a child of the `Document` object, and you should therefore reference the `Document` object prior to referencing the `Form` object. There are many occasions when you will need to reference the `Form` object, most of which involve reading the value of text boxes and other elements of a form. However, you can both read and set two properties of the `Form` object itself. The `Action` property is the URL of the server-side script or CGI program that will receive the form's data; the `Method` property determines whether the form data should be sent to the server in a `GET` or `POST` HTML transaction.

The following example shows how you can dynamically change the `Action` property of a form:

```
<SCRIPT LANGUAGE="vbscript">
Sub cmdSubmit_OnClick
   If x = True then
       Document.frmForm1.Action = "http://www.justanycom.com/anasp.asp"
   End If
End Sub
</SCRIPT>
...
<FORM NAME="frmForm1" METHOD="POST">
...
</FORM>
```

Now here's the same code, but this time the reference to the form uses its ordinal position rather than its name:

```
<SCRIPT LANGUAGE="vbscript">
Sub cmdSubmit_OnClick
   If x = True then
       Document.Forms(0).Action = "http://www.justanycom.com/anasp.asp"
   End If
End Sub
</SCRIPT>
...
<FORM METHOD="POST">
...
</FORM>
```

You can also use the `Form` object to programmatically submit the form data, perhaps based on the outcome of a data validation routine, as the following example shows:

```
<SCRIPT LANGUAGE="vbscript">
   Sub cmdSubmit_OnClick
      If isDataValid = True Then
          Document.frmForm1.Submit
      Else
          Alert "Data not submitted"
      End If
   End Sub

   Function isDataValid()
       If IsDate(Document.frmForm1.InvDate.Value) Then
           isDataValid = True
       Else
           Alert "Value must be a date"
           isDataValid = False
       End If
   End Function
</SCRIPT>
<FORM NAME="frmForm1" ACTION="http://www.www.net/www.asp" METHOD="POST">
  <INPUT TYPE="text" NAME="InvDate">
  <INPUT TYPE="button" NAME="cmdSubmit" VALUE="SUBMIT">
</FORM>
```

In the preceding example, an HTML button has been used rather than the normal HTML Submit. The code attached to the button's `OnClick` event calls a function, which checks that the data input in the `InvDate` text box is a valid date; if it is, the function returns `True` and the form data is submitted using the Form object's `Submit` method.

Referencing the `Elements` Object

`Elements` objects are the intrinsic HTML controls that you place on an HTML form to solicit data from the user of the page. The object itself is an array and is a child of the `Form` object. Therefore, to reference an individual control, you use this syntax:

```
Document.form.control
```

With so many cool ActiveX form controls to choose from, you might wonder why you should even bother with HTML controls. There are a couple of good reasons. First, you might already have spent time developing your HTML forms, and you might want to leverage this investment by simply adding scripts to your existing forms. Second, not everyone will have loaded the full install of MSIE; those who didn't would not have the ActiveX Forms 2.0 controls, and therefore you would have to make them available for download from your Web site—which will take time and bandwidth. At least you can be sure that everyone can use intrinsic HTML controls. Here's a list of the intrinsic HTML controls (`Elements` controls). The following intrinsic HTML controls are all defined using the `<INPUT>` tag:

- Button: This Windows-style command button is similar to the popular Submit button. You can add scripts to the Button's `Click` event.
- Checkbox: Use this to allow the user to make selections when more than one item can be selected.
- Hidden: No control is displayed on the page. Use this to "hold" data to be submitted to the server within a form.
- Password: This is similar to the text control, except characters typed into the control appear as asterisks.
- Radio: This is an option button, which is used when only one of a range of choices is allowed.
- Reset: This clears all form controls.
- Submit: This button control has one purpose—to submit the form data to the server.
- Text: This is a good old-fashioned text box control for entering data. It's important to remember that all data passed from a textbox to a script is in the form of a string, even if the data is numeric.

The Select control, which creates a selectable list, is not defined with the `<INPUT>` tag, but it can be treated like the rest of the intrinsic controls when creating scripts.

Let's first look at an event that all the text-based HTML controls support: the `OnChange` event. It's fired when the user moves away from a control after changing its value from the last time the `OnChange` event was fired for the particular control. You can use this event to validate the data as the user is completing the form, like this:

```
<SCRIPT LANGUAGE="vbscript">
   Sub txtQuantity_OnChange
       If Not IsNumeric(Document.frmForm1.txtQuantity.Value) Then
           Alert "Quantity must be numeric"
       End If
   End Sub
</SCRIPT>
```

The preceding code also demonstrates how to read the value of a text-based control using its `Value` property. You can also set the `Value` property, automatically filling in parts of a form:

```
<SCRIPT LANGUAGE="vbscript">
   Sub Window_OnLoad
       Document.frmForm1.txtTodaysDate.Value = Now()
   End Sub
</SCRIPT>
```

In the preceding example, the code executes after the page has completed downloading into the browser by being attached to the `Window`'s `OnLoad` event. The code assigns the current date and time to the text box's `Value` property using the VBScript `Now()` function.

USING OBJECT VARIABLES

You might have noticed by now that when you reference individual controls on an HTML form, the code line can become quite long and unwieldy; furthermore, every dot that separates the individual objects in the hierarchy represents a function call deep inside the scripting engine. This becomes even more of a problem when you start adding frames into the equation. For example, this line of code references a value in a text box in the frame adjacent to the one containing the script:

```
dateValue = Top.frRightFrame.Document.frmForm1.txtDateVal.Value
```

See what I mean? If you had to write similar code for many fields on a form, not only would it take a lot of coding time and space, but it is also difficult to read. A solution is to use object variables. An object variable is simply a pointer to a particular object that you can use in place of the object. You create object variables using the VBScript `SET` command, like this:

```
Dim myForm
Set myForm = Top.frRightFrame.Document.frmForm1
```

Now, every time you use the variable, it is the same as using the complete hierarchical path to the object:

```
dateValue = myForm.txtDateVal.Value
```

Working with Frames

HTML frames are a popular method of creating easy-to-use Web applications. They can also be a boon for the script writer, enabling you to place all your scripts within one of the frame documents. However, care must be taken when you reference HTML objects in other frames.

The key to referencing the controls in one frame from a script in another frame is to always remember to start your reference from the topmost object in the HTML object hierarchy and work your way down the hierarchy until you arrive at the particular control.

For example, if you have a `FRAMESET` document like

```
<FRAMESET COLS=50%,50%>
  <FRAME NAME="frLeftFrame" SRC="scripts.htm">
  <FRAME NAME="frRightFrame" SRC="adoc.htm">
</FRAMESET>
```

the document `adoc.htm` contains a form like this:

```
<FORM NAME="frmAnyForm" METHOD=POST ACTION="something.asp">
Quantity <INPUT TYPE="text" NAME="txtQuantity">
<BR>
EMail Address <INPUT TYPE="text" NAME="txtEMail">
</FORM>
```

For a script within the left frame (`scripts.htm`) to reference the controls on the `frmAnyForm`, you must begin your reference with the `Window` object, then reference the right-hand frame, followed by the `Document`, `Form`, and finally the control itself:

```
Window.frRightFrame.Document.frmAnyForm.txtQuantity.Value
```

Just as frames can be nested (a frameset within a frameset), so too can `Frame` objects. Suppose that you had a further frameset within the right frame—the two frames called `frTopFrame` and `frBottomFrame`. If the form resided in the bottom right frame, you would reference the control like this:

```
Window.frRightFrame.frBottomFrame.Document.frmAnyForm.txtQuantity.Value
```

A sound understanding of the HTML object model will enable you to update your HTML Web pages with Active scripting; you will be able to control the browser, the document, the forms, and the intrinsic controls. With very little effort, you'll be able to turn once flat, lifeless HTML pages into dynamic applications that take advantage of the speed that client-side scripting can bring. However, to really bring your pages up-to-date with the very latest Active content, you need to go one step further and add ActiveX controls.

Adding ActiveX Controls to HTML Documents

The scripting techniques you've seen so far have revolved around the built-in functionality of the Web browser. To improve on this functionality, you must add new and exciting controls to your Web pages. Microsoft has leveraged years of development to bring you ActiveX controls that can add the sort of functionality normally associated only with a Windows desktop program.

What Is an ActiveX Control?

An ActiveX control is a program, or an executable file that exposes certain methods, properties, and events through a programmable interface. ActiveX controls have been around for quite a while, but have only recently been referred to as ActiveX controls. They adhere to the component object model (COM) standard; this standard has been amended recently to allow slimmer controls to be created for ease of download across the Internet.

An ActiveX control is built in such a way that other programs that are also built to the COM interface standard can interface with it. In this way, VBScript can read and set its properties, handle its events, and call its methods.

Here are some of the many different types of ActiveX controls:

- User-interface controls are the text boxes and other forms controls and buttons that you probably are used to using in Windows desktop applications. Other user-interface controls include menu bars, image controls, and hotspot controls.
- Server-side components are ActiveX controls that have no graphical user interface (GUI). These controls are called into action by a server-side script to perform a particular function (such as interfacing a database). They execute and return a result to the calling script. These components used to be known as remote automation servers or OLE servers.
- Client-side components are a special type of controls that, like the server-side components, have no visible interface for the user; they are simply add-on programs that you as a script writer can use to enhance the range of functionality in your script. An example of this type of ActiveX control is the Timer, which fires an event at specified periods.

It's impossible to know how many ActiveX controls are available from third-party software vendors, but it runs into many thousands, and more are coming to market every day. It is also possible for you to create your own controls using the Control Creation Edition (CCE) of Visual Basic 5.

Built-In ActiveX Controls

The full install of MSIE installs a range of commonly used ActiveX forms controls. The built-in ActiveX controls include these:

- The *checkbox* control allows the user to select a number of options.
- The *combobox* is a basically a single control, which is a combination of a text box and a drop-down list.
- The *commandbutton*—just click it!
- The *hotspot* control enables you to create programmable graphical areas.
- The *image* control lets you create a graphical area on the HTML page that is much more flexible than the HTML `<IMG>` tag.

- The *label* control is a text area that cannot be amended by the user; it is entirely under the programmer's control.
- The *listbox* control presents a list of values.
- The *optionbutton* control enables the user to select one of a number of options.
- Both vertical and horizontal *scrollbars* are available from a single control.
- The *spinbutton* control is used to increment and decrement a specific user-definable value.
- The *tabstrip* control enables you to include the now familiar notebook metaphor in a Web page.
- The *textbox* control is the ubiquitous data entry element.
- The *toggle button* control is a dual on and off button that the user can click to switch on and click again to switch off.

Third-Party ActiveX Controls

If you have a particular requirement that can't been fulfilled using the Forms 2.0 controls, you have to either create your own control using VB5 CCE or Visual C++, or look around the marketplace for an off-the-shelf solution. The best place to commence your search is the Microsoft ActiveX Gallery within the Microsoft Site Builder Network, from where you can download both fully functional and trial versions of some of the very best ActiveX controls.

Using the ActiveX Control Pad

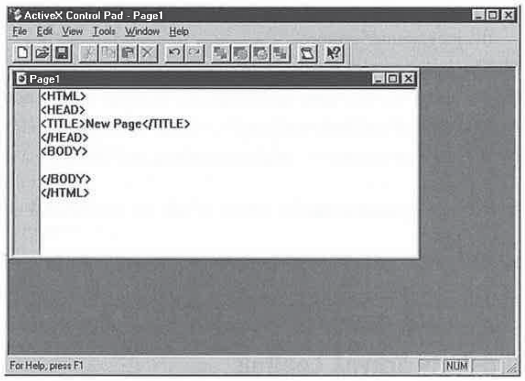
To include an ActiveX control on a Web page, you must refer to it using the control's unique 128-bit ID. I don't know about you, but I'm not too hot at remembering 128-bit IDs like 978C9E23-D4B0-11CE-BF2D-00AA003F40D0. Luckily, Microsoft's developers can't remember these IDs either, and they quickly realized that a tool was needed for us all to be able to add ActiveX controls to a Web page. Thus the ActiveX Control Pad, shown in Figure 30.1, was born.

The ActiveX Control Pad can be downloaded for free from Microsoft's Web site, and it is also included on the CD-ROM that accompanies this book.

As shown in Figure 30.1, an HTML template is opened automatically when you launch the program. You can close this if you wish and open your own HTML page using the File|Open command.

To illustrate how easily you can create dynamic Web pages using the ActiveX Control Pad, the following example takes you through adding a couple of ActiveX controls to a new HTML page, then shows you how to attach some VBScript using the Script Wizard that is part of the ActiveX Control Pad. You'll be creating a page that contains a Label control and a ComboBox control. The ComboBox will contain a list of font sizes from 8–24; when the user of the page makes a selection from the ComboBox, the font size of the label's caption changes to that selected by the user.

FIGURE 30.1.
The ActiveX Control Pad.



First, launch the ActiveX Control Pad. Before you start with any active content, you need to make these changes to the HTML template:

1. Change the title to something like "Test ActiveX Page."
2. Add a `BGCOLOR="white"` parameter to the `<BODY>` tag.
3. Add a `<CENTER>` tag under the `<BODY>` tag.

Now you're ready to start adding the ActiveX controls. The Label control is the first to be added, as follows:

1. Place your cursor under the `<CENTER>` tag.
2. Select Insert ActiveX Control from the Edit menu.
3. From the list of available controls, select the Microsoft Forms 2.0 Label and click OK.

A Design window and Properties window for this Label control will now be displayed. You can use the Design window to drag the size of the label and the Properties window to set the properties you need for this control.

Setting a Control's Properties

Change these properties for this control:

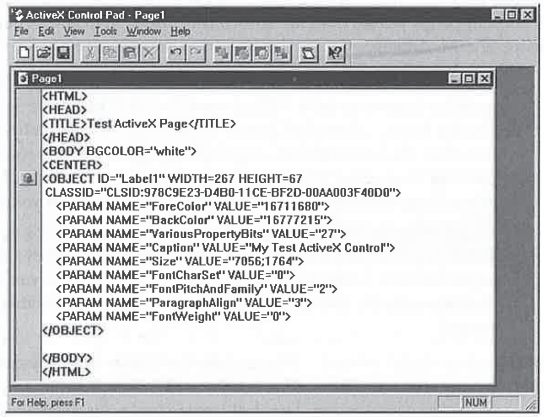
1. `BackColor`: This is the background color of the label itself. Click on the property name; the property value will be repeated in a drop-down list at the top of the Properties window. To the right of the down arrow button is a button with three dots (an ellipsis) on it. Click this button to display the Color dialog. Select the White color block and click OK. Click the Apply button.

A special selection dialog is displayed whenever you click a button with an ellipsis.

2. `Caption`: This is the text that is displayed on the label. Select the `Caption` property, type the required caption, `"My Test ActiveX Control"`, in the property setting box at the top of the Property window, and click Apply.
3. `ForeColor`: This is the color of the caption on the label, and you set it just like you did `Backcolor`: Make it a shade of blue.
4. `Height`: This property determines how tall the label will be; set it to `100`.
5. `TextAlign`: You can choose to align the caption to the Left, Center, or Right of the label. For this example, set it to Center by selecting the `TextAlign` property, then selecting Center from the list of options.
6. `Width`: This property determines the how wide the label is; set it to `300` pixels.

Your label is now customized for this example. To add the code to the HTML page that will define this label, close the Label Design window by clicking the X button in the top right corner. The `<OBJECT>` tag and the required parameters are now generated automatically by the Control Pad and pasted into the page at the point where your cursor was placed. Your page should now resemble Figure 30.2.

FIGURE 30.2.
The HTML definition for the ActiveX label.



You now need to add your second control, the ComboBox control, by following these steps:

1. Place a `<P>` tag after the `</OBJECT>` tag to create a little room on the page between the controls. Now press the Enter key to move the cursor to the next line.
2. Select Insert ActiveX Control from the Edit menu.

3. From the list of available controls, select Microsoft Forms 2.0 ComboBox and click OK.
4. A Design window and Properties window for this ComboBox are now shown. You don't need to change any of the properties for this control, so simply close the control design window to generate the HTML definition code for the ComboBox.

With the controls added, you now need to attach some scripting to the page and the ActiveX controls to achieve the desired results.

Attaching Scripts to ActiveX Controls: The Script Wizard

The ActiveX Control Pad includes a Script Wizard that simplifies the writing of most scripts. However, you must bear in mind that because of the nature of wizards, not every possible scenario can be catered for, and sometimes you might need to manually attach code directly onto the HTML page. This is particularly true of scripts that you need to execute on download.

For this example you need to first populate the ComboBox with even values from 8 to 24. You need to add code to the ComboBox's `Click` event so that when the user selects a font size from the list with the mouse, the font size is translated to the Label control's caption.

Launch the Script Wizard by either clicking the toolbar button that looks like an ancient scroll or selecting Script Wizard from the Tools menu. You need to note a couple of things about the Script Wizards defaults:

- Ensure that the Script Wizard's default is VBScript. The Script Wizard has the capability to generate both VBScript and JScript. Change the default language by selecting Script... from the Options menu, which you access from the Tools menu; then select the desired default scripting language. Some very strange folk might choose to default their Script Wizard to JScript (though I can't think why!); however, because we're talking VBScript here, it might benefit you to default yours to VBScript.
- Second, when the Script Wizard launches, use the Code View option. The List View option is quite honestly less than useless. To see what I mean, after you've written the scripts shown in Listing 30.3, try to switch to List view and you'll be told that because a custom script has been written, the List view cannot work. But aren't all scripts custom?

Okay, back to the job in hand... You need the ComboBox to be populated only once. You can only reference the control after it has been created, which means that you have to wait for the page to complete downloading. The ideal event to use in this situation is the Window's `OnLoad` event. This is fired after the page has been displayed and the script has been parsed through the scripting engine. Follow these steps to add your code to the `OnLoad` event:

1. In the events pane of the Script Wizard, click the plus sign (+) to the left of the `Window` object. This displays the two `Window` object events, `Onload` and `OnUnLoad`.
2. Select the `OnLoad` event.

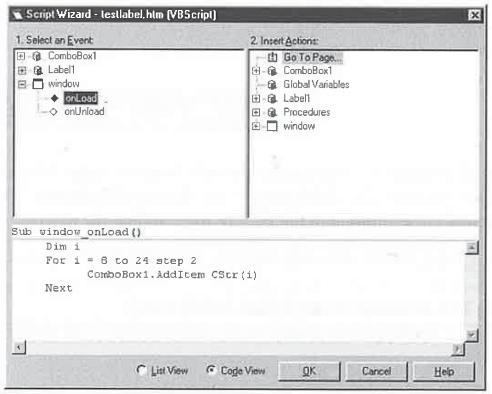
3. Type the following code in the code window:

```
Dim i
For i = 8 to 24 step 2
      ComboBox1.AddItem CStr(i)
Next
```

The first line of this code declares a local variable, `i`. The next line starts a loop that will execute eight times. `8 to 24 Step 2` specifies that on the first iteration, the counter `i` will have a value of `8`, and it will increase by 2 on each subsequent iteration until it has a value of `24`, after which time the loop will terminate. On each iteration, the counter `i` will be converted to a string value using the `CStr()` function, and then added to the ComboBox list with the `AddItem` method.

Your Script Wizard window should now resemble Figure 30.3.

FIGURE 30.3.
The Window `OnLoad` *event handler.*



The next stage is to create an event handler for the ComboBox control's `Click` event.

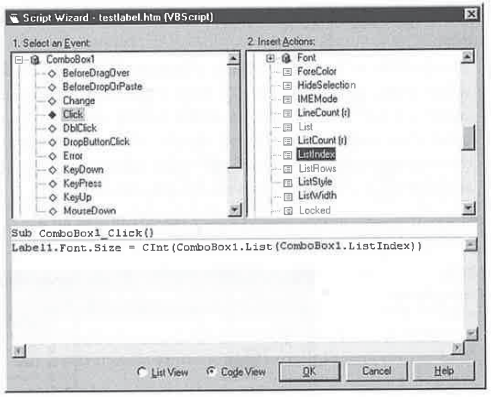
1. In the events pane of the Script Wizard, click the plus sign to the left of the `ComboBox1` object. This displays all the ComboBox events.
2. Select the `Click` event.
3. Type the following code in the code window:

```
Label1.Font.Size = CInt(ComboBox1.List(ComboBox1.ListIndex))
```

`ComboBox1.ListIndex` returns the index number of the item selected by the user. This is then used to return the text from the list that relates to this index. Because the value returned will be a string value and font sizes are numbers, you have to convert the value to an integer using the `CInt` function. This number is then assigned to the `Size` property of the label's `Font` object.

Your Script Wizard window should look like the one in Figure 30.4.

FIGURE 30.4.
The ComboBox `Click` *event handler.*



Now all you need to do is transfer the code from the Script Wizard to the HTML page. This is done quite easily by clicking the Script Wizard's OK button. The Script Wizard will then transpose the scripting into the HTML page, adding the required `<SCRIPT>` tags and so on.

Listing 30.3 contains the complete source code for this example.

Listing 30.3. An example of using dynamic ActiveX controls.

```
<HTML>
<HEAD>
    <SCRIPT LANGUAGE="VBScript">
<!--
Sub window_onLoad()
     Dim i
     For i = 8 to 24 step 2
          ComboBox1.AddItem CStr(i)
     Next
end sub
-->
    </SCRIPT>
<TITLE>Test ActiveX Page</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<CENTER>
     <OBJECT ID="Label1" WIDTH=400 HEIGHT=60
 CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0">
    <PARAM NAME="ForeColor" VALUE="16711680">
    <PARAM NAME="BackColor" VALUE="16777215">
    <PARAM NAME="VariousPropertyBits" VALUE="27">
```

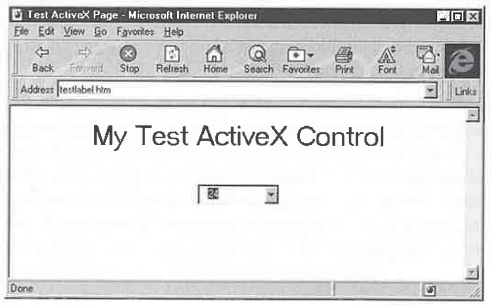


```
    <PARAM NAME="Caption" VALUE="My Test ActiveX Control">
    <PARAM NAME="Size" VALUE="10583;1588">
    <PARAM NAME="FontCharSet" VALUE="0">
    <PARAM NAME="FontPitchAndFamily" VALUE="2">
    <PARAM NAME="ParagraphAlign" VALUE="3">
    <PARAM NAME="FontWeight" VALUE="0">
</OBJECT>
<P>
  <OBJECT ID="ComboBox1" WIDTH=96 HEIGHT=24
     CLASSID="CLSID:8BD21D30-EC42-11CE-9E0D-00AA006002F3">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="DisplayStyle" VALUE="3">
        <PARAM NAME="Size" VALUE="2540;635">
        <PARAM NAME="MatchEntry" VALUE="1">
        <PARAM NAME="ShowDropButtonWhen" VALUE="2">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>

    <SCRIPT LANGUAGE="VBScript">
<!--
Sub ComboBox1_Click()
Label1.Font.Size = CInt(ComboBox1.List(ComboBox1.ListIndex))
end sub
-->
    </SCRIPT>
 </BODY>
</HTML>
```

All that needs to be done now is to save the file and run it through the browser—as long as the browser is MSIE. When you select a font size from the list, the label's caption immediately resizes itself to the new font. Figure 30.5 shows how the page should look in the browser.

FIGURE 30.5.
`testlabel.htm`.



Making Controls Available to Your Users

Although this is not relevant for the preceding example, there are many occasions when you will need to make a control you are using available to the users of your Web page.

Suppose, for example, you decided to use the new SuperWiz WebMenu control throughout your Web site, you design your pages using the control, and you try the pages on your local machine and are ecstatic with the results. You upload the Web pages to your server, ready to win award after award for the coolest Active Content Web site. There's just one slight problem: If your users don't have the SuperWiz WebMenu on their machines, too, all they will see is...nothing!

If you go back to the properties page for one of the controls you added in the example from the preceding sections (you can click the cube icon at the side of the Object definition to edit a control), you will see a property called `CodeBase`. This tells the client browser where it can find the control in question.

In fact, what actually happens is that the browser first checks in a cache directory where it keeps downloaded controls. If it finds the control locally, it ignores the `CodeBase` tag—unless there is a difference in the version information. `CodeBase` can be either a path relative to the HTML file or it can be an absolute path.

ActiveX and Active Scripting Security Issues

Before moving on to look at a way you can create a true Windows type form in an HTML document, let's stop for a moment and consider some security aspects.

First, consider the case of all Active Scripting; I use the generic here because this material applies equally to VBScript and JScript. Client-side scripts are safe. They cannot operate outside of the browser. The only time a client-side script can access a hard drive is via the document's `Cookie` property, and even then it is the browser, not the script, that is accessing the hard drive. VBScript on the client side is prevented from creating any object that would allow it access to anything outside of the browser.

An ActiveX control, on the other hand, is an executable program; as such, it has the potential to contain malicious code. However, Microsoft has put in place the code signing system, which warns the user when an ActiveX control is being downloaded from a new source. If you aren't sure of the source of the control, you should take precautions. One problem for the developer of Active Content pages, though, is that some individuals and organizations have taken their Web security to such a point that they automatically refuse any ActiveX control. When designing Active Content pages, therefore, you should bear this in mind and provide an alternative.

The Microsoft HTML Layout Control

Not very long ago, I was asked to put together a Web site for a company; they wanted the site to mimic one of their in-house Windows applications. The application's user interface was a complex affair, providing many various and dynamic options—that is, options that generated further options.

At the time all I had to work with was the good old HTML form controls, which where just not up to the job, at least not without many trips back and forth to the server in the course of completing just one set of data input. I can still remember wishing that I could replicate the functionality of a Windows application on a Web page.

Those wishes have now been answered by the HTML Layout control, an ActiveX control that you create yourself in less time than it takes to say "graphical user interface."

What Is the HTML Layout Control?

The Layout control is a blank form. Sounds exciting, huh? Let me put it another way. It's like a blank canvas ready and waiting for you to express your innermost creativity in form and application design through the medium of the Web page. Okay, so it's a blank form.

The Layout control is a true Window. It accepts and sends Windows messages like the desktop Windows applications written in C++ or VB. It allows you, therefore, to create applications that look, feel, and actually are true Windows programs.

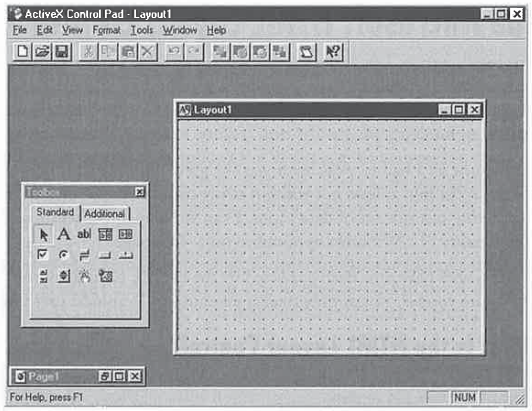
After you drag and drop the form and other controls you need onto the canvas of the Layout control, you can place the controls with pixel accuracy—something that is nearly impossible on a normal HTML document. You then add your VBScript code to the Layout control, save it as one single ALX file, and insert the ALX file into the HTML page just like any other ActiveX control.

Designing Forms with the HTML Layout Control

To give you just a taste of what you can do with the HTML Layout control, the following example will take you through the process of designing a simple input form, adding some validation code, and then adding the Layout control to an HTML page.

To start with, you need to launch the HTML Layout control designer. Launch the ActiveX Control Pad, then select New HTML Layout from the File menu. The Layout Control designer, shown in Figure 30.6, is launched, containing a blank form ready for you to start work.

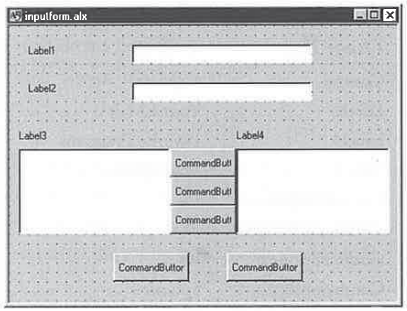
FIGURE 30.6.
The HTML Layout Control designer.



Adding controls to the Layout control is easy. Simply click on the control in the toolbox you wish to add, then click on the Layout background in roughly the area where you want the control, hold the mouse button down, and drag the control to the size you require. After a control has been placed on the Layout background, it can be resized and moved using the mouse. To add the controls for your sample application, follow these steps:

1. Select the TextBox control in the toolbox, click and hold the mouse button down on the Layout background, and drag the new text box into position. For this example, you're going to need two text boxes, one under the other, so add another text box under your first one.
2. Add two labels, one for each text box.
3. Add two list boxes; space them at the bottom of the form with a gap between.
4. Add three command buttons; place them between the two list boxes.
5. Add two additional command buttons at the bottom of the form.
6. Add two additional labels—one for each list box.
7. Save as `inputform.alx`. Your form should resemble Figure 30.7.

FIGURE 30.7.
`inputform.alx`.



Setting Properties of Controls in an HTML Layout

You now need to change certain properties of the controls you've just added. You need to change the caption properties of the labels and buttons to be somewhat more meaningful than simply the control's name. You can also change the color scheme to something more appealing than the standard gray and black. The HTML Layout control enables you to change the properties of several similar controls with one command, as you're about to see. Follow these steps:

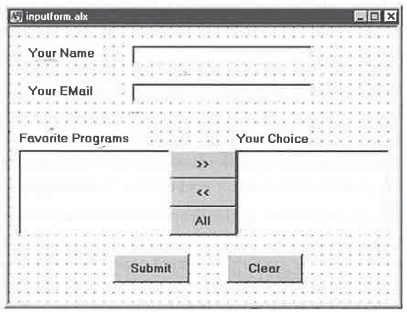
1. Select all the labels and buttons by first selecting one label or button, then hold down the Ctrl key while you click on the other labels and command buttons. Notice that all the controls you've clicked on have sizing blocks on their edges, indicating they have been selected. While still holding the Ctrl key down, you can deselect a control by clicking it a second time. With all the labels and command buttons selected, right-click one of them and select `Properties` from the pop-up menu.
2. The properties shown in the Properties window are common to both the Label and TextBox controls. You can now change the `Font` property and have that change reflected in all the currently selected controls. Select Font from the properties list and click the ellipsis button to display the Font properties dialog. Change the font to 10 point, bold. Click OK and click Accept at the top of the Properties window.
3. You now need to set the captions for each of the labels and buttons. Start by selecting the top label (`label1`); right-click to display the properties window. Select the `Caption` property, change the caption to `"Your Name"`. Click Accept to translate the property change to the control.

4. Similarly, change the caption properties of the other labels and controls as follows:

Change This:	*To This:*
Control	Caption
Label2	Your EMail
Label3	Favorite Programs
Label4	Your Choice
CommandButton1	>>
CommandButton2	<<
CommandButton3	All
CommandButton4	Submit
CommandButton5	Clear

5. The final part of this beautification process is to change the color scheme. Select all the label controls, and change the `BackColor` property to White.
6. Right-click the layout background, select Properties, and change the `BackColor` property to White. Your form should now look like the one in Figure 30.8.

FIGURE 30.8.
The completed user interface.



Attaching Scripts to the Layout Control

Of course, without some scripting behind the controls, there is very little that the form can do. This particular example is an input form in which users can enter their names and e-mail addresses, and select their favorites from a range of programs. You're going to make the following changes to bring the form life:

- Populate the Favorite Programs list as the layout completes downloading into the browser.

- Add the item selected by the user to the Your Choice list.
- Add all Favorite Programs items to the Your Choice list.
- Remove an item from the Your Choice list if necessary.
- Validate the e-mail address.
- Clear all controls.

All this functionality is going to be added to the form using the Script Wizard, so first, click the Scroll button on the toolbar to launch the Script Wizard. Follow these steps:

1. To code the populating of the Favorite Programs list, in the events pane, click the plus sign to the left of `Layout1` and select the `OnLoad` event. Enter the following code in the script window:

```
Dim sPrograms
sPrograms = Array("Microsoft Visual Basic", "Borland Delphi", _
                  "Pegasus Mail", "Internet Explorer", _
                  "Netscape Navigator", "Visual Interdev")
ListBox1.List = sPrograms
```

   This defines a variable that will hold the list in memory ready to assign to the list, and then the list of values is created using the `Array()` function. The `List` object is itself an array, so you can simply assign the array of values to the `Listbox1.List`.

2. Now for some very rudimentary data validation. Here you're going to check that the e-mail address entered contains an at symbol (@); this, of course, is the very minimum requirement for a valid e-mail address. Again in the events pane, click the plus sign to the left of the `TextBox2`, select its `Exit` event, and enter this code in the script window:

```
If InStr(TextBox2.Text, "@") = 0 Then
    Alert "This Email Address appears to be invalid"
End If
```

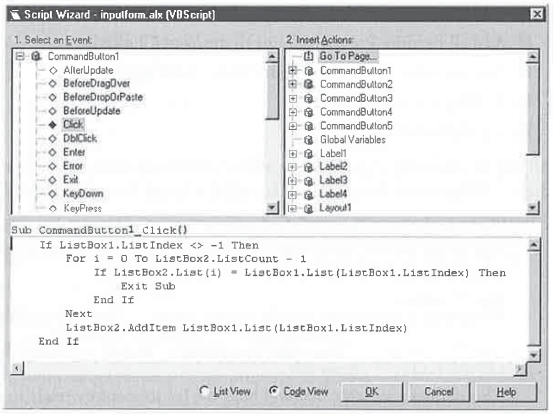
   The preceding code uses the `InStr()` function, which returns the position of the specified character within a string, or returns `0` if it is not found.

3. After the user has chosen a particular item from the Favorite Programs list, he/she clicks the >> button to add the selected item to the Your Choice list. In the events pane, click the plus sign to the left of `CommandButton1` and select its `Click` Event. Enter this code in the script window:

```
If ListBox1.ListIndex <> -1 Then
    For i = 0 To ListBox2.ListCount - 1
        If ListBox2.List(i) = ListBox1.List(ListBox1.ListIndex) Then
            Exit Sub
        End If
    Next
    ListBox2.AddItem ListBox1.List(ListBox1.ListIndex)
End If
```

   The code should look like the code in the script window in Figure 30.9.

FIGURE 30.9.
The `CommandButton1` `Click` *event.*



In this code, you are first checking that a selection has been made. If no selection has been made by the user, the list box's `ListIndex` property will return –1. If this is the case, your script need go no further. However, assuming that the user has selected an item from the list, you must then check that the selection hasn't already been added to `ListBox2` by iterating through the items in the list. If a match is found, this item has already been added and the `Exit Sub` terminates execution of the event handler. If the selected item isn't found in `ListBox2`, you can safely add the item to the list.

4. You can add some extra speed functionality by allowing the user to double-click the Favorite Programs list to add an item to their choices list. Rather than rewriting the preceding code in the double-click event of the list box, you can simply call the button's `Click` event, which replicates the user clicking the button. To code this, click the plus sign to the left of the `ListBox1` object in the events pane and select the `DblClick` event. Move to the actions pane and click the plus sign to the left of the Procedures item and double-click the `CommandButton1_Click` item. This will automatically place a line of code in the event handler:

```
call CommandButton1_Click()
```

   Thus, whenever a user double-clicks an item on the list, it has exactly the same effect as first selecting the item and then clicking the button.

5. You now need to give the user the ability to remove an item from the list of chosen programs. To do this, add the following code to the `Click` event of `CommandButton2`:

```
If ListBox2.ListIndex <> -1 Then
    ListBox2.removeItem ListBox2.ListIndex
End If
```


   Again, you must first check that a selection has been made, then you can call the list box's `removeItem` method, passing it the index number of the item to be removed.

6. As with the favorites list, you can include a `DblClick` event for the choices list that will call the `CommandButton2` `Click` event:

```
Sub ListBox2_DblClick(Cancel)
    call CommandButton2_Click()
end sub
```

7. To add all items from the favorites list to the choices list, you need to assign the `ListBox1.List` property to the `ListBox2.List` property; however, you should first clear out any items currently in `ListBox2`. Add the following code to `CommandButton3`'s `Click` event:

```
call ListBox2.Clear()
ListBox2.List = ListBox1.List
```

8. This example isn't actually going to do anything with the form data, so just add an alert box to the `Click` event of `CommandButton4`:

```
Alert "Submitting data"
```

9. The final code will reset all the controls by assigning zero length strings to the text boxes, clearing the Choices list box, setting the `ListIndex` of `ListBox1` to –1 (no items selected), and finally placing the cursor in `TextBox1` using the text box's `setFocus` method. So, you need to add the following code to the `Click` event of `CommandButton5`:

```
TextBox1.Text = ""
TextBox2.Text = ""
ListBox2.Clear
ListBox1.ListIndex = -1
TextBox1.SetFocus
```

When that's done, all that remains is to click the OK button on the Script Wizard to transpose all of the code you've entered into the ALX Layout control file. Save the Layout control, and you're now ready to add it to the HTML page. Before you do that, though, just make a few amendments to the standard HTML template provided in the Control Pad.

Give it a title of "Sample ALX Form," add a `BGColor ="white"` parameter to the `<BODY>` tag, then under the `<BODY>` tag add the following lines of HTML:

```
<CENTER>
<FONT FACE="arial">
<H2>Computer Users Survey</H2>
```

Inserting the Layout Control into an HTML Page

Place the cursor on an empty line under the `<H2>` heading and select Insert HTML Layout from the Edit menu. This activates a File Open dialog, which should be defaulted to the current directory. Select the `inputform.alx` file and click OK. The ActiveX Control Pad will then insert the HTML code to define your Layout control, and your HTML page should now look like Figure 30.10.

FIGURE 30.10.
The HTML object definition for the Layout control.

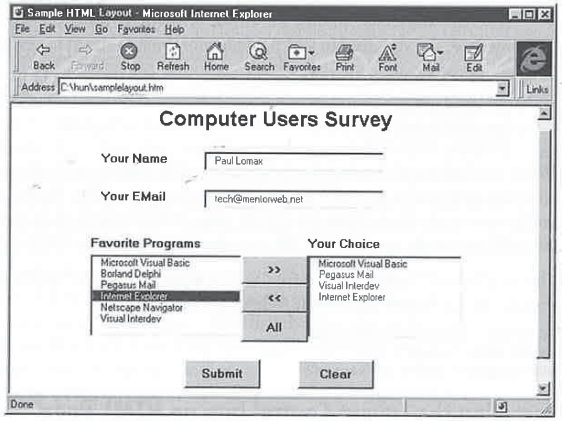
```
samplelayout.htm
<HTML>
<HEAD>
<TITLE>Sample HTML Layout</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<CENTER>
<FONT FACE="arial">
<H2>Computer Users Survey</H2>

<OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
ID="inputform_alx" STYLE="LEFT:0;TOP:0">
<PARAM NAME="ALXPATH" REF VALUE="inputform.alx">
</OBJECT>

</BODY>
</HTML>
```

Save the HTML File as `samplelayout.htm`, and you're ready to run it though the browser. The complete listing for both the `inputform.alx` Layout control file and `samplelayout.htm` are shown in Listings 30.4 and 30.5, respectively. Figure 30.11 shows the Layout control within the HTML page running in the MSIE browser.

FIGURE 30.11.
`samplelayout.htm` *running in the browser.*



Listing 30.4. `inputform.alx`.

```
<SCRIPT LANGUAGE="VBScript">
<!--
Sub Layout1_OnLoad()
  Dim sPrograms
  sPrograms = Array("Microsoft Visual Basic", "Borland Delphi", _
                    "Pegasus Mail", "Internet Explorer", _
                    "Netscape Navigator", "Visual Interdev")
  ListBox1.List = sPrograms
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub TextBox2_Exit(Cancel)
    If InStr(TextBox2.Text, "@") = 0 Then
        Alert "This Email Address appears to be invalid"
    End If
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub CommandButton1_Click()
    If ListBox1.ListIndex <> -1 Then
        For i = 0 To ListBox2.ListCount - 1
            If ListBox2.List(i) = ListBox1.List(ListBox1.ListIndex) Then
                Exit Sub
            End If
        Next
        ListBox2.AddItem ListBox1.List(ListBox1.ListIndex)
    End If

end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub ListBox1_DblClick(Cancel)
    call CommandButton1_Click()
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub CommandButton2_Click()
    If ListBox2.ListIndex <> -1 Then
        ListBox2.removeItem ListBox2.ListIndex
    End If
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub CommandButton3_Click()
    call ListBox2.Clear()
    ListBox2.List = ListBox1.List
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub ListBox2_DblClick(Cancel)
    call CommandButton2_Click()
```

continues

Listing 30.4. continued

```
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub CommandButton4_Click()
    Alert "Submitting Data"
end sub
-->
</SCRIPT>
<SCRIPT LANGUAGE="VBScript">
<!--
Sub CommandButton5_Click()
      TextBox1.Text = ""
      TextBox2.Text = ""
      ListBox2.Clear
      ListBox1.ListIndex = -1
      TextBox1.SetFocus
end sub
-->
</SCRIPT>
<DIV BACKGROUND="#ffffff" ID="Layout1" STYLE="LAYOUT:FIXED;WIDTH:338pt;
HEIGHT:241pt;">
    <OBJECT ID="ListBox1"
     CLASSID="CLSID:8BD21D20-EC42-11CE-9E0D-00AA006002F3"
STYLE="TOP:107pt;LEFT:8pt;WIDTH:132pt;HEIGHT:74pt;TABINDEX:0;ZINDEX:0;">
        <PARAM NAME="ScrollBars" VALUE="3">
        <PARAM NAME="DisplayStyle" VALUE="2">
        <PARAM NAME="Size" VALUE="4657;2619">
        <PARAM NAME="MatchEntry" VALUE="0">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="ListBox2"
     CLASSID="CLSID:8BD21D20-EC42-11CE-9E0D-00AA006002F3"
STYLE="TOP:107pt;LEFT:198pt;WIDTH:134pt;HEIGHT:74pt;TABINDEX:1;ZINDEX:1;">
        <PARAM NAME="ScrollBars" VALUE="3">
        <PARAM NAME="DisplayStyle" VALUE="2">
        <PARAM NAME="Size" VALUE="4710;2619">
        <PARAM NAME="MatchEntry" VALUE="0">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="TextBox1"
     CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"
STYLE="TOP:17pt;LEFT:107pt;WIDTH:157pt;HEIGHT:17pt;TABINDEX:2;ZINDEX:2;">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="Size" VALUE="5530;582">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="TextBox2"
     CLASSID="CLSID:8BD21D10-EC42-11CE-9E0D-00AA006002F3"
```



```
STYLE="TOP:50pt;LEFT:107pt;WIDTH:157pt;HEIGHT:17pt;TABINDEX:3;ZINDEX:3;">
        <PARAM NAME="VariousPropertyBits" VALUE="746604571">
        <PARAM NAME="Size" VALUE="5530;582">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="0">
    </OBJECT>
    <OBJECT ID="Label1"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0" STYLE="TOP:17pt;
LEFT:17pt;WIDTH:66pt;HEIGHT:17pt;ZINDEX:4;">
        <PARAM NAME="BackColor" VALUE="16777215">
        <PARAM NAME="Caption" VALUE="Your Name">
        <PARAM NAME="Size" VALUE="2328;583">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="Label2"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0" STYLE="TOP:50pt;
LEFT:17pt;WIDTH:83pt;HEIGHT:17pt;ZINDEX:5;">
        <PARAM NAME="BackColor" VALUE="16777215">
        <PARAM NAME="Caption" VALUE="Your EMail">
        <PARAM NAME="Size" VALUE="2911;582">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="CommandButton1"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:107pt;LEFT:140pt;WIDTH:58pt;HEIGHT:25pt;TABINDEX:6;ZINDEX:6;">
        <PARAM NAME="Caption" VALUE="&gt;&gt;">
        <PARAM NAME="Size" VALUE="2037;873">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="CommandButton2"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:132pt;LEFT:140pt;WIDTH:58pt;HEIGHT:25pt;TABINDEX:7;ZINDEX:7;">
        <PARAM NAME="Caption" VALUE="&lt;&lt;">
        <PARAM NAME="Size" VALUE="2037;873">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="CommandButton3"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:157pt;LEFT:140pt;WIDTH:58pt;HEIGHT:25pt;TABINDEX:8;ZINDEX:8;">
```

continues

Listing 30.4. continued

```
        <PARAM NAME="Caption" VALUE="All">
        <PARAM NAME="Size" VALUE="2037;873">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="Label3"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0" STYLE="TOP:91pt;
LEFT:8pt;WIDTH:116pt;HEIGHT:17pt;ZINDEX:9;">
        <PARAM NAME="BackColor" VALUE="16777215">
        <PARAM NAME="Caption" VALUE="Favorite Programs">
        <PARAM NAME="Size" VALUE="4075;583">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="Label4"
     CLASSID="CLSID:978C9E23-D4B0-11CE-BF2D-00AA003F40D0" STYLE="TOP:91pt;
LEFT:198pt;WIDTH:66pt;HEIGHT:17pt;ZINDEX:10;">
        <PARAM NAME="BackColor" VALUE="16777215">
        <PARAM NAME="Caption" VALUE="Your Choice">
        <PARAM NAME="Size" VALUE="2328;583">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="CommandButton4"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:198pt;LEFT:91pt;WIDTH:66pt;HEIGHT:25pt;TABINDEX:11;ZINDEX:11;">
        <PARAM NAME="Caption" VALUE="Submit">
        <PARAM NAME="Size" VALUE="2329;873">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
    <OBJECT ID="CommandButton5"
     CLASSID="CLSID:D7053240-CE69-11CD-A777-00DD01143C57"
STYLE="TOP:198pt;LEFT:190pt;WIDTH:66pt;HEIGHT:25pt;TABINDEX:12;ZINDEX:12;">
        <PARAM NAME="Caption" VALUE="Clear">
        <PARAM NAME="Size" VALUE="2328;873">
        <PARAM NAME="FontEffects" VALUE="1073741825">
        <PARAM NAME="FontHeight" VALUE="200">
        <PARAM NAME="FontCharSet" VALUE="0">
        <PARAM NAME="FontPitchAndFamily" VALUE="2">
        <PARAM NAME="ParagraphAlign" VALUE="3">
        <PARAM NAME="FontWeight" VALUE="700">
    </OBJECT>
</DIV>
```


Listing 30.5. samplelayout.htm.

```
<HTML>
<HEAD>
<TITLE>Sample HTML Layout</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<CENTER>
<FONT FACE="arial">
<H2>Computer Users Survey</H2>

<OBJECT CLASSID="CLSID:812AE312-8B8E-11CF-93C8-00AA00C08FDF"
ID="inputform_alx" STYLE="LEFT:0;TOP:0">
<PARAM NAME="ALXPATH" REF VALUE="inputform.alx">
 </OBJECT>

</BODY>
</HTML>
```

Summary

VBScript and ActiveX enable you to create state-of-the-art, dynamic, and interactive HTML pages and Web applications. Whether you're looking to create an outstandingly creative Web site or translate a current Windows application to the Web, Microsoft Active Content is the fastest and most highly developed path to take.

You can use VBScript to interface the browser, HTML Intrinsic form controls, and ActiveX controls. The VBScript language and syntax is straightforward and relatively easy to learn. VBScript also has the advantage of being a subset of Visual Basic; therefore, the millions of VB programmers in the world will be able to get up and running with VBScript in no time at all. Most of the VBScript language actually pre-dates Windows, giving you the benefit of many years of development behind the code you create.

ActiveX controls have been with us for much longer than most people realize. They are, in essence, the very building blocks of Windows itself. The available pool of ActiveX controls is immense, and tools such as the new VB5 CCE allow you to create your own ActiveX controls.

The possibilities of what you can achieve with VBScript and ActiveX are endless and are governed only by your own imagination. Enjoy!

CHAPTER 31

VRML Primer

by Bruce Campbell

IN THIS CHAPTER

- Adding the Third Dimension *630*
- What the Third Dimension Does for the Web *630*
- How a Web Surfer Sees a VRML Model *632*
- The History of VRML and the VRML Community *633*
- Similarities Between VRML and HTML *635*
- Differences Between VRML and HTML *636*
- An Overview of VRML Syntax *637*
- Adding VRML to a Web Page *644*
- Embedding VRML in an HTML Page *644*
- Adding a VRML Frame *645*
- Communications Between VRML and HTML *646*
- The Future of VRML *647*

If the success of a computer language is determined by the number of people using and writing about it, HTML is a very successful electronic information presentation language. Tens of millions of electronic files available on the World Wide Web are formatted following the rules and syntax of HTML outlined in this book. And, HTML is evolving as new developments such as cascading style sheets (see Chapter 19, "Introducing Cascading Style Sheets,") and Dynamic HTML (see Chapter 22, "Dynamic HTML") add new capabilities to the language. There is a second Web language, the Virtual Reality Modeling Language (VRML), which continues to be successful in adding a third dimension to information presentation on the Web. VRML files exist alone on the Web to provide navigable 3D scenes to a Web audience or, as focused on in this chapter, VRML files are embedded within HTML Web pages to present information based on the best of each language.

Adding the Third Dimension

Depth is usually considered the third dimension. Depth is often ignored when designing computer applications because the computer screen is only two dimensional. The third dimension was ignored in the first cave paintings produced by human beings, probably because the flat cave walls had two dimensions, width and height. But as the study of art progressed, intricate theories of building three-dimensional images on two-dimensional canvases were developed, and depth was added to painted images. Today's Web pages are created using a language that handles two dimensions quite adequately for the information presentation techniques of today, but is the Web at the same point analogously as painting was in the days of the cave painter?

What the Third Dimension Does for the Web

Web authors use HTML to provide tours of interesting places to visit. Universities put virtual tours of their campuses on their Web servers so surfers can point and click through digitally scanned photographs of key buildings and green spaces. NASA provides pictures of Mars, Jupiter, Saturn, and other heavenly bodies on its Web pages. City governments do the same to provide virtual tours of their cities. The technique is a successful one because so many Web navigators have used the same presentation formats in their own vacation scrapbooks and have learned to click like crazy when using the mice attached to their computers.

Yet, something is lost in the translation of three-dimensional places into sample photographs placed with text on a Web site. There is no sense of continuity of the pictures into a coherent whole. No easy way to put the whole place in the mind at once. The audience often imagines a place that is laid out quite differently in reality. Mars and Jupiter appear to be the same size and very close to each other when viewed as two successive pictures on a Web site. In fact, they are millions of miles apart and Jupiter is thousands of times larger than Mars.

Visualization is considered one of the most important aspects in the progression of science. Physics, chemistry, genetics, and astronomy all are explained in terms of mathematical concepts that are very difficult to understand. Without understanding the existing theories,

scientists cannot support, improve upon, or debate against the current body of knowledge in their fields. Presenting scientific theories as three-dimensional, interactive models is often more useful for an audience's understanding than a series of pictures supported by text.

The Web is ripe for 3D content that the world can visit and navigate from within a Web browser. Although most people have traditionally learned through text, pictures, and movies and readily accept the same format for Web-based learning, learning through experiencing 3D models compliments other methods quite well. People experience 3D models in six directions: right, left, up, down, forward, and backward. Interacting with 3D models can be as simple as using the arrow keys on a standard 84- or 102-key keyboard and as complex as manipulating sophisticated onscreen controls with the mouse and keyboard.

Imagine two brain surgeons. One has learned to remove tumors dangerously close to the brain stem by studying a series of pictures provided in a medical textbook. The other has actually practiced performing the surgery on a virtual 3D model of a brain with a tumor placed in a very dangerous location. In an emergency, which brain surgeon would most people want if they were to be the doctor's first patient? No doubt about it, people understand the benefit of the third dimension in this case.

Imagine a virtual tour of a new ballpark for the favorite sports team of someone who has held season tickets for 10 seasons. One tour lets the fan click to see pictures of different parts of the stadium. The other lets the fan move around the ballpark to sit in any seat by moving up, down, forward, backward, right, and left within a 3D model. No doubt, most season ticket holders would prefer the second method of checking out the new park.

The third dimension helps the human mind organize information in a format it understands well on a daily basis: a model of the location of a work cubicle within a building, the look of a loved one from any angle. Humans are built to encounter new information in three dimensions and digest it directly in the mind. If only there were a way to use the Web to provide navigable 3D places in a Web browser. Then, for much subject matter, Web authors would have an option to let viewers choose the order in which they experience a new place on the Web, yet still have an organized visit.

VRML Fits the Bill

The Virtual Reality Modeling Language is growing up on the Web as a common interest of people who understand the workings of HTML, support the World Wide Web, understand computers, computer networking and computer graphics, and want the Web to have a standard way to become a navigable place people can visit and do things in. Computers are evolving quickly and finally have the capability to perform desktop virtual reality (VR). Science fiction books such as *Neuromancer* by William Gibson and *Snow Crash* by Neal Stephenson provide the imaginative vision of what a 3D navigable Web could be. There is talk of a second Web continually navigable in three-dimensional space.

VRML has already been standardized twice. VRML 1 defined standard syntax for creating three-dimensional models that could be created by combining 3D models of objects from all over the Web. VRML 2, finalized in August 1996, redefines the syntax a bit and adds an event model to the objects in a VRML model. The event model was necessary because VRML enthusiasts were concerned that VRML 1 output was too stagnant and didn't provide enough interactivity to capture a person's attention. Yet, as creative artists and scientific visualization folks began to use the standard, some very impressive VRML 1 worlds were created and made available on the Web.

How a Web Surfer Sees a VRML Model

VRML files are stored on and delivered from Web servers that need no additional software, just HTML Web servers. Yet, a Web surfer needs additional software to be able to view a VRML file from within a Web browser. The additional software is called a VRML *viewer* and works with both the Netscape Navigator and Microsoft Internet Explorer Web browsers as a *plug-in* application. (Plug-in applications are covered in detail in Chapter 32, "Plug-ins.") During installation of a VRML viewer, the installation routine communicates with the Web browser to set up a new x-world/x-vrml MIME type associated with `.wrl` and `.wrz` file extensions. The `.wrz` extension is an explicit extension for a VRML world file that has been compressed using a common compression algorithm called gzip. Yet today, even WRL files can be compressed and the VRML viewer recognizes that fact when it opens the file in a Web browser.

Figure 31.1 shows an example of an embedded VRML file in an HTML document. The VRML creates a billiard table as part of a billiard arcade. The VRML appears in a square area on the HTML page as defined with the following `<EMBED>` tag of the HTML Web page:

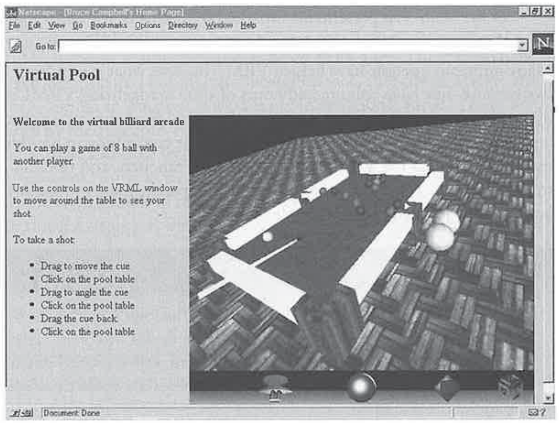
```
<embed src="pool.wrl" border=3 height="500" width="500" align="right">
```

Complete HTML examples follow later in the chapter.

In Figure 31.1, the Web surfer has previously downloaded the latest copy of the CosmoPlayer VRML 2 viewer from Silicon Graphics Inc.'s Web site using the URL `http://vrml.sgi.com/cosmoplayer`. During installation, CosmoPlayer registered itself to Netscape Navigator, the Web browser. Then, the user downloaded the VRML billiard table from another Web site. CosmoPlayer provides the onscreen controls that the user can manipulate with the mouse to move forward, backward, up, down, right, and left within the billiard arcade scene. In fact, the ball middle control allows the user to spin the world in order to quickly examine its geometry.

The Web is full of HTML pages that document the history of VRML and compare, contrast, and explain VRML viewers. The VRML page at the San Diego Supercomputer Center, called the VRML Repository, is often recommended as a starting place for first-time VRML investigators. The URL for the VRML Repository is `http://www.sdsc.edu/vrml`. Home pages for the most popular VRML 2 viewers also include independent VRML information as well as links to interesting 3D worlds. Silicon Graphic's VRML café at `http://sgi.vrml.com/cafe` is another VRML Web site that focuses on current events and white papers about the further development of VRML and related technologies.

FIGURE 31.1.
A VRML file embedded in an HTML document.



VRML 2 has become the most used Web language for presenting interactive 3D content to a Web visitor. It is capable of providing a virtual 3D worm for dissecting, a virtual new Tiger Stadium, a virtual coffee house, a virtual galaxy, and a virtual DNA double helix. VRML viewers let Web surfers interact with scenes like these in intuitive ways. Web authors can add text narrative on the same Web page using HTML with VRML embedded.

The History of VRML and the VRML Community

The history of VRML is most interesting for the fact that much of the work in creating the standard was done on the Internet. Using electronic mail through a simple, yet heavily subscribed to, mailing list and a couple of usenets, anyone interested in helping guide the standard had an outlet for voicing an opinion and responding to others' opinions. The Internet proved a successful medium for getting the right minds together with the right critical thoughts. Proposals for the continual improvement of VRML were promoted by many different individuals and organizations, and the VRML community together discussed and voted on the best suggestions. The VRML Architecture Group (VAG) evolved as a group of individuals responsible for the recording of the standard and its promotion. Their Web site at `http://vag.vrml.org/` still maintains old documents that track the history of the VRML standard as well as the final VRML 2 specification. The history document is especially helpful if you are interested in the history of VRML. For example, the history link at the VAG Web site begins like this:

> In 1994, Tim Berners-Lee (the founder of HTML) invited Mark Pesce to present a paper at the First International Conference on the World Wide Web. Pesce and partner Tony Parisi had developed Labyrinth, a prototype three-dimensional interface

to the Web. His presentation sparked a consensus: the conference attendees agreed there was a need for a common language to specify 3D scene descriptions.

Many impressive people have helped VRML become what it is today; I consider them impressive because they have volunteered hours of time evangelizing VRML as an important tool in the road to human expression, creativity, learning, communication, networking, and socializing (as well as a tool for Earth conservation) while continuing to provide solid technical expertise. The center of the VRML universe is San Francisco, and the lead evangelist definitely is still Mark Pesce. Beyond that, referring to any specific names in a short history section would undoubtedly leave important people out.

NOTE

How can VRML help with Earth conservation? The idea is that VRML models of the Earth can be developed to track depletion of the rain forests over time, current pollution levels, current birth rates, and so on. In fact, in the future, with the appropriate sensors and modeling in place, all kinds of interesting scientific data could be accessible in real-time through a VRML model.

Geographical pockets of VRML interest are truly worldwide. Many of the members of the VRML community are compelling writers whose visionary papers have helped VRML gain acceptance and, perhaps more important, enthusiasm. There are many books on VRML 1 that still should be read for the vision they provide for VRML. Yet, to be technically astute with the current language, a VRML 2 book is a must.

The Java development team at Sun has recently (May 1997) introduced a 3D API as a specification for creating and manipulating 3D applications within the Java language itself. I talked with a representative from Sun, who assured me that the intent was not to compete with VRML 2. (SGI has supported Sun in developing the 3D API specification.) Still, a 3D interface from Sun is worth investigating. OpenGL is another 3D graphics API that has a lot of supporters. In fact, many graphics acceleration boards are created primarily to enhance the OpenGL 3D function calls.

The VRML standard has reached a plateau; any technology is finally considered big business. With the World Wide Web Consortium (W3C) as a template, the VRML Architecture Group is maturing to become the VRML Consortium with a more formally recognized governing body. The formal organization should prepare it for what no doubt is on the horizon: new features in the standard, venture capital to be directed, and ISO standard recognition. These days are an important time in VRML's development, and changes are in the works (which, by the way, is nothing new here).

The next section takes a look at what VRML 2 is today and how it can be mixed with HTML.

VRML Scenes

Creative VRML-based Web content is stored as one or more WRL files. These files contain text following the UTF8 convention that HTML also subscribes to. UTF8 covers a larger character set than ASCII but is similar in purpose, with a more international focus. These WRL files are often called *worlds* by their creators because they focus on the virtual reality aspect of their design. Authors hope their audience feels so immersed in the content that they feel like they are in a different place—not sitting in front of a computer screen—and that their actions as they use the computer are an extension of their bodies. Immersion is a word used by virtual reality developers to describe the degree to which the computer's reality gains the attention of the user's senses. In immersive VR, immersion of sight and sound is aided by a head-mounted device (HMD) that becomes the user's sole field of vision and source of sound.

Web browsers are currently not run immersively by most people. Instead, VRML content is shown via desktop VR, and a user is more apt to think of the content as a model when compared to an immersively experienced world. Somewhere between the world and model descriptions of VRML content is the concept of a scene. In the long run, I suspect, the word *model* might become the accepted term as a descriptor of the output of a VRML file. However, I find that the word *world* is still the most commonly used term.

VRML scenes are created out of *nodes* the VRML-enabled viewer program parses and presents on the Web page. These nodes are put together to define hundreds of different possible features of a 3D scene. Each node has a list of *fields* that can be used to subdefine the properties of the node. The order in which the nodes and fields are listed is significant in many cases, and the aggregation of them all is called a *scene graph.* It is called a graph in the same vein as a mathematician refers to a graph: as a way to describe an organization of objects that possess certain characteristics. Understanding HTML is helpful in learning VRML. The syntax of the two languages has many similarities.

Similarities Between VRML and HTML

Besides being created as text-based files following the same UTF8 standard, VRML and HTML syntax are similar in the following ways:

- VRML uses short text strings to identify the appearance or behavior of every node that makes up a VRML scene; similarly, HTML uses short text strings to declare tags that dictate the appearance of objects on a Web page. For example, VRML uses the word `Sphere` to tell the Web browser to draw a sphere as part of a VRML scene, and HTML uses the string `BR` to tell the Web browser to make a line break in written text.
- VRML nodes have a start and end point; HTML tags have pairs that dictate the start and end of a tag's influence. For example, the `Transform` node in VRML transforms only the objects inside of its braces, and the `<B>` tag in HTML boldfaces only the words between the `<B>` and `</B>`.

- VRML fields help subdefine the node; HTML attributes subdefine the tag. For example, the `transform`, `scale`, and `rotation` fields of a VRML `Transform` node dictate how the objects inside the node are presented in a way similar to how the `BGCOLOR`, `VLINK`, `LINK`, `TEXT`, and `BACKGROUND` attributes of a `<BODY>` HTML tag dictate how the body of an HTML page is presented.
- Both languages read their content sequentially from top down, left to right, and both languages ignore white space such as spaces, carriage returns, and line feeds beyond one white space character unless it is within quotes. Text placement is flexible so that the author can organize the text in any way deemed appropriate.
- Both languages lend themselves to editor applications that can be used to make the text creation process easier. HTML editors enable an author to use buttons and drop-down lists in conjunction with the mouse to create Web pages. VRML editors, often called modeling packages, enable an author to create 3D geometries and color or texture them using drag-and-drop techniques involving the mouse with buttons and drop-down lists. Both languages have many programmers working at making better tools for content creation, yet both allow an author to use a cheap, simple text editor to create the content. Cheap is important if everyone is to have access to authoring on the Web.

Differences Between VRML and HTML

Although VRML and HTML coexist nicely on a Web server and the language mindset is similar, the two languages look dramatically different in syntax, and their domains overlap only slightly in places. VRML does have a `Text` node that places text in a VRML scene. VRML has an `Anchor` node used to provide hyperlinking from VRML scenes to other Web objects such as Web pages or other VRML scenes. But so many VRML nodes are foreign to an HTML expert. HTML is concerned with text, related text objects such as lists and tables, and 2D pictures called bitmaps. VRML, on the other hand, is concerned with representing a three-dimensional model, explicitly defining a z direction perpendicular to the plane of a traditional computer screen. VRML nodes, once learned, are natural and provide insight as to how the human mind organizes 3D information encountered in day-to-day living. Learning VRML makes one ponder human physiology and the human mind.

Visual objects in VRML scenes are made of shapes that are the combination of primitive geometries provided by the language or hand crafted by connecting points together counterclockwise in 3D space. Primitive geometry nodes include the `Box`, `Sphere`, `Cone`, `Cylinder`, and `ElevationGrid`. 3D space is defined along three axes—x (left to right), y (down to up), and z (back to front). Points in space defined as x,y,z coordinates are combined using the `IndexedFaceSet` node in order to create polygons that are combined to create solid objects. For example, the point (10,3,–4) refers to a point 10 units from the right of the origin, 3 units above the origin, and 4 units back from the origin.

Each visual object has an appearance that is defined by colors and textures. Textures are created the same way HTML graphics are created and a Web author can often take advantage of the same HTML background tiling technique in VRML to keep texture file sizes small. An author's VRML textures can also take advantage of the same transparency available to HTML graphics to overcome burdensome shape detail such as that required by a tree or mountain range.

VRML contains different lighting nodes to provide lighting effects, a `Fog` node to vary clarity of the scene, a `Background` node to simplify the creation of objects off in the distance, and sound nodes to incorporate sound into a scene and even associate the sound to a specific 3D location.

VRML `Viewpoint` nodes provide convenient camera coordinates and directions that can be used to move to significant locations within the VRML scene. The VRML viewer is always showing a viewpoint to the Web surfer, yet many of the intermediary viewpoints are reached dynamically by the user's actions. `Viewpoint` nodes are convenient bookmarks in the scene that can be accessed by a visitor who is unsure of where to go.

Many of the new nodes of VRML 2 track the user's viewpoint or user's actions through the mouse or other pointing device to allow a visitor to interact with other nodes. For example, a `TouchSensor` node can be associated with a doorbell object that, when clicked, opens up a door object. A `PositionInterpolator` node identifies how the door should open, and a `Timer` node keeps track of how long it should take to open. Separate nodes can set up a close event for the door. A `SphereSensor` node can set up an event based on the user's current location, triggering an event when the user moves within the hidden geometry of the `SphereSensor` located in 3D space.

Events associate VRML nodes with each other through a `ROUTE` statement. A `ROUTE` statement connects the nodes that can trigger events with nodes that control the timing of the event or the actual transformation of the objects involved in the event. Transformations available in VRML 2 include change of location (called translation in the VRML specification), change of scale, change of orientation, change of appearance, as well as lighting and camera transformations for any one or more objects associated within the same object group.

An Overview of VRML Syntax

A working knowledge of VRML syntax can be quickly gained by learning the nodes that make up the language. There is no need to memorize all the details because there are many reference documents available on the Web, with `http://vag.vrml.org/VRML2.0/FINAL` being the most official. Instead, learning is best done by reading a good book and making small changes to example VRML scenes to see the effect of the changes. After a quick introduction to some key nodes, this section moves on to the VRML billiard table example so you can learn by focusing on the process.

NOTE

A book I co-authored with Chris Marrin, *Teach Yourself VRML 2 in 21 Days*, demonstrates each feature of VRML 2 that enables creative 3D Web page interactivity. Many book titles on VRML 1 and VRML 2 can be found at the VRML Repository Web site at the San Diego Supercomputer Center (`http://www.sdsc.edu/vrml`). The VRML Repository Web site is worth investigating for its wealth of information on VRML and its unbiased presentation style.

The key organizing nodes of the VRML 2 standard are the `Transform`, `Inline`, and `Group` nodes. These nodes are parent nodes to the other nodes that define a VRML scene. A `Transform` node associates one or many children nodes together so that they can be manipulated together as a single object when desired. A group node associates children as well, but without providing any manipulation at that level. An `Inline` node associates other nodes together and provides a URL to get those nodes from another file that is located on the Web; the `Inline` node is important for taking advantage of the unique features of the Web for sharing information and collaborating.

Initially, the children nodes of interest are the ones that add visual objects to the scene. In the case of the following billiard table example, each component of the table is added as a child object to the same `Transform` node. After the components are associated with each other in the same `Transform`, they can be manipulated as a group to change their location, size, or orientation uniformly for all components. For a simple billiard table, the visual objects include a slate surface, six bumper cushions, six pockets, and a table base, as well as 16 balls. To start slowly, the first code example, Listing 31.1, has only one pocket, one bumper, and a solitary cue ball. Figure 31.2 shows the partial billiard table.

Listing 31.1. A primitive VRML billiard table.

```
#VRML V2.0 utf8
DEF overview Viewpoint {
    position         200 250 500
    orientation      0 0 1  0
    fieldOfView  0.785398
    description     "OVERVIEW"
}
DEF Billiards_Table Transform {
    children [
        DEF Slate Transform {
           children [
              DEF TSB TouchSensor {}
              Shape {
                 appearance Appearance {
                      material Material {diffuseColor 0 .2 0 }
                      # texture ImageTexture {url "green1.gif" }
                 }
```



```
                 geometry Box {size 312 512 .05 }
              }
           ]
           translation 156 256 -12
        },
        Transform {
           children [
              DEF Bumper Shape {
                 appearance Appearance {
                      material Material {diffuseColor .8 .7 .7 }
                      # texture ImageTexture {url "pink1.gif" }
                 }
                 geometry Box {}
              }
           ]
           scale 125 15 15
           translation 154 -15 2
        },
        Transform {
           children [
                DEF Pocket Shape {
                 appearance Appearance {
                      material Material {diffuseColor .1 .1 .02 }
                      # texture ImageTexture { url "brown1.gif" }
                 }
                      geometry IndexedFaceSet {
                            coord Coordinate {
                                 point [ 0 0 0, 30 0 0, 20 20 0,
                                         20 30 0, 30 40 0, 40 40 0,
                                         60 30 0, 60 60 0, 0 60 0,
                                         0 0 -80, 0 60 -80,
                                         60 60 -80, 60 0 -80 ]
                            }
                            coordIndex [ 0, 1, 2, 3, 4, 5, 6, 7, 8, -1,
                                         0, 8, 10, 9, -1, 9, 10, 8, 0, -1,
                                         7, 8, 10, 11, -1,
                                         11, 10, 8, 7, -1,
                                         11, 10, 9, 12, -1 ]
                      }
                 }
           ]
           translation -30 480 20
        },
        DEF Ball Transform {
           children [
              DEF TSC TouchSensor {}
              Shape {
                 appearance Appearance {
                      material Material {diffuseColor 1 1 1 }
                 }
                 geometry Sphere {radius 10 }
              }
           ]
           translation 150 400 0
        },
    ]
} # end Billiards_Table
```

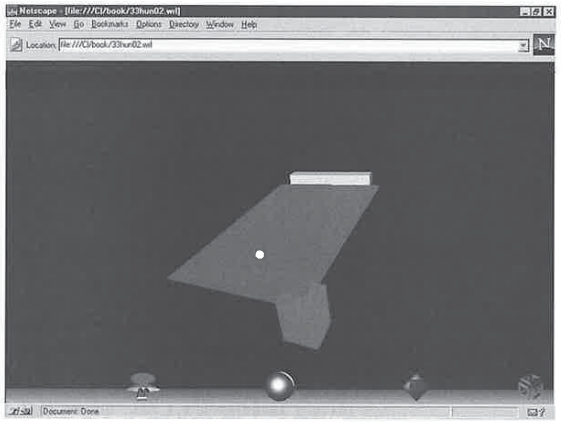


FIGURE 31.2.
A partial billiard table showing one pocket, one bumper, and the cue ball.

The first line in Listing 31.1 is called the header line. The header line identifies the version of the VRML standard to which the file format is complying. Its syntax is very specific, including the # character, which anywhere else in the file acts solely as a comment identifier. A comment is active until the next line feed is encountered. The second line starts the `Transform` node for the billiard table. The `DEF` keyword allows a node to be named and then reused according to name. Corresponding `Transform` nodes for the `Slate`, `Bumper`, `Pocket`, and `Ball` are also named using the `DEF` keyword.

NOTE

The `DEF` keyword acts like an object identifier in an object-oriented programming language. The node it defines becomes reusable in the rest of the VRML scene. As mentioned in the "The Future of VRML" section later in this chapter, other programming languages are being used to manipulate objects in a VRML scene. These languages interface with the VRML file through the nodes defined with `DEF` keywords.

Each visual object is a child `Transform` node of the `Billiards_Table` `Transform` node. Each visual object contains a `Shape` node as a child of the `Transform` node, as well as potential `translation`, `scale`, and `rotation` fields to be discussed after the `Shape` node discussion. Each `Shape` node has two fields: an `appearance` field that specifies the color or texture of the object and a `geometry` field that specifies the area of physical coordinates the object occupies in 3D space.

The `appearance` field expects an `Appearance` node to follow. Yet, this field-node relationship allows flexibility in the future of VRML development when other node types may be appropriate for the `appearance` field. The `Appearance` node contains a `texture` and/or `material` field that specifies the color and texture of the `Shape` node it is a part of. In Listing 31.1, the `ImageTexture` nodes are commented out, so only the effect of the `Material` node is seen in Figure 31.2. Textures use external bitmap files specified using their absolute or relative URLs. The bitmaps are wrapped around the geometry defined in the same `Shape` node. Still, the bitmaps can be the exact files used in the `<IMG>` tag of an HTML Web page. They are traditionally JPG and GIF files. The `Material` node defines a `diffuseColor` field that gives a `Shape` node its predominant color. Textures and materials are a significant part of the VRML specification; several additional fields and transformations are available to the `Shape` node.

NOTE

The hierarchical ordering of the nodes and fields presents the best opportunity for reuse because any node can be defined with a `DEF` keyword at any point in the hierarchy. In this case, the whole `Billiards_Table` can be reused to create multiple tables in the arcade, the `Bumper` can be reused to create six of them on a single table, or an `Appearance` node can be reused to create the same look to other visual objects. The hierarchy also works well inside the computer after the scene has been parsed and stored internally because it is designed to take advantage of a stack and its related algorithms.

The `geometry` field of the `Shape` node is different for each visual object on the billiard table. For the Slate `Transform` and Bumper `Shape` nodes, the geometry is defined as a `Box` node. The Slate box has dimensions of 312 units in the x direction, 512 units in the y direction, and .05 units in the z direction. For the Cue `Transform` node, geometry is defined as a `Sphere` node with a radius of 10 units.

NOTE

Using the primitive shapes of `Box`, `Sphere`, `Cylinder`, and `Cone` reduces the VRML file size, yet is very constricting to the exact shapes. These shapes contain many polygons, and polygon counts are a significant indicator of performance in the Web browser. So, they must be used with caution in a VRML scene.

The `geometry` field of the Pocket `Shape` node uses an `IndexedFaceSet` node to derive a very specific shape out of coordinates in 3D space. Each point is separated by a comma, and the points are connected using the list of indexes in the `coordIndex` field. Each component polygon of the shape is created by a list of point indexes that all end with a –1 and are connected counterclockwise. Five polygons create the corner pocket seen in Figure 31.2. Points are reused in

different polygons to guarantee a solid seam around the object. The pocket is not completely closed off because the neighboring bumpers will make the pockets appear solid. The `IndexedFaceSet` node is at the heart of every unique VRML object shape encountered on the Web. It can be expanded to show incredible detail by connecting tens of thousands of points. Detail requires associated computer processing power that is not always available to each Web site visitor, but the VRML standard is poised for the future through the `IndexedFaceSet` node.

The `translation`, `scale`, and `rotation` fields are critical to the flexibility of the VRML scene. Any group of objects nested in the `children` field can be manipulated together using the other fields of the `Transform` node. The `translation` field defines the point in 3D space around which the objects are centered. The `scale` field defines the relative x, y, and z size of the objects as multipliers of their `Shape` nodes' defined `geometry` fields. The `rotation` field defines the orientation of the objects relative to their `Shape` nodes' defined `geometry` fields. These three fields become necessary when an object is reused in the VRML scene. They also allow for the manipulation of an object defined in an external file that is being added to the scene using the `Inline` node.

The other three corner pockets in the billiard table are similar to the first one defined in Listing 31.1. Instead of creating them from scratch, they are included as child `Transform` nodes of the `Billiards_Table Transform` node based on the original `Pocket Shape` node. For example, the opposite corner pocket looks like this:

```
Transform {
    children [
        USE Pocket
    ]
    rotation 0 0 1 3.14
    translation 340 30 20
},
```

Here, the `Pocket` shape is reused by referring to it with the `USE` keyword. The shape is placed in a `Transform` node in order to rotate the pocket 180 degrees and move it to its appropriate location relative to the `Slate Transform` node. The other five bumpers can be created like this:

```
Transform {
   children [
      USE Bumper
   ]
   scale 15 105 15
   translation 324 375 2
},
```

For the bumpers, the shape is reused by referring to the `Bumper Shape` node definition with the `USE` keyword. Each bumper is scaled from the default `Box` node size of 2 by 2 by 2 by using the `scale` field of the `Transform` node. In this example, the final dimensions will be 30 (2 times 15) by 210 (2 times 105) by 30 (2 times 15). The center location of this bumper will be 324 units to the right, 375 units above, and 2 units in front of the (0,0,0) origin.

Nodes reused by reference to a defined node must appear after the defined node in the text file, so the preceding two reuse examples would appear as child nodes of the `Billiards_Table Transform` node after the child `Transform` nodes containing the `DEF` keyword. Six other child nodes taking advantage of reuse would create the last two corner pockets and four bumpers seen in Figure 31.1. Fifteen additional billiard balls could be created as well as child `Transform` nodes of the `Billiards Table Transform` node. The side pockets could be created using a different `IndexedFaceSet` node. Figure 31.3 shows an improved VRML billiard table after taking advantage of VRML node reuse.

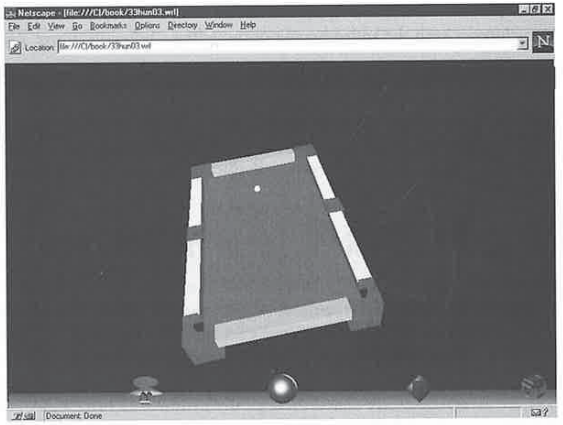


FIGURE 31.3.
A VRML billiard table with object reuse.

Two last points of interest are related to Listing 31.1: First, a `Viewpoint` node was added to explicitly start the VRML viewer camera at a specific location and orientation in the scene. Without it, different VRML viewers start at different viewpoints. Second, `TouchSensor` nodes were added as children to the `Slate` and `Ball Transform` nodes to set up an expectation for user interaction with the scene. `TouchSensor` nodes make the objects in which they are contained aware of the mouse pointer. VRML specifies events that can be triggered by passing the mouse over or clicking upon objects that contain the `TouchSensor` node.

The example in Listing 31.1 is typical of basic VRML syntax and functionality. VRML 2 has an impressive list of nodes and fields that could build this billiard table into an interesting billiard world to explore and interact with. The most likely would include adding lights and simple events. Although some of the vision for enhancing this VRML scene is discussed in the section "The Future of VRML," I encourage you to explore other books for explicit examples. The next section focuses on adding the VRML scene to an HTML Web page assuming the VRML file has been saved in a text editor with a `.wrl` extension.

Adding VRML to a Web Page

VRML can be added to an HTML document using two distinct approaches. The first approach embeds a VRML scene within a rectangular area of the Web page as if it were a bitmap image. In a single Web page, multiple VRML scenes can be added this way. As a Web surfer scrolls a larger Web page, the VRML scene will scroll off the page just as a picture would. The alternative is to load the VRML scene in a separate frame, set up with a frame definition document as described in Chapter 18, "Creating Sophisticated Layouts with Frames and Layers." With this option, the VRML scene will stay onscreen as a Web surfer scrolls a related HTML document in a separate frame. Either approach can be used to set up the look and feel of an electronic encyclopedia with the added benefit of providing a fully interactive VRML scene.

Any method of adding a VRML scene to a Web page usually reduces the size of the VRML scene to appear less than full screen to the audience. VRML viewer plug-ins are programmed to scale the opening viewpoint relative to the available presentation size and resolution of the VRML scene; the smaller the area, the better the VRML scene will perform in responding to the user's manipulation of the scene. Reducing the size also reduces the visitor's sense of immersion.

Embedding VRML in an HTML Page

Figure 31.1 presents a VRML billiard table that has been embedded in an HTML page. The HTML text could continue for many screens worth of information on the rules of play, details about interacting with the world, and even a physics lesson on the dynamics of billiard balls colliding in a constrained area. As the viewers wish to read more, they could scroll down below the VRML scene, read more information, and then scroll back up to continue playing with the scene. The scrollbars are added dynamically by the Web browser window whenever they are needed to see the full Web page's content.

The HTML syntax for the Web page presented in Figure 31.1 is provided in Listing 31.2. The HTML tags used for the text presentation are very common. The file `pool.wrl` is the VRML billiard arcade scene that is embedded into the Web page at the point it is encountered within the HTML file. The `<EMBED>` tag uses an `SRC` attribute that sets up the `pool.wrl` relative URL as the source of the contents of the area created with a `HEIGHT` attribute of 500 pixels high and a `WIDTH` attribute of 500 pixels wide. The VRML scene is aligned along the right of the Web browser window by way of the `ALIGN` attribute. A `BORDER` with thickness of 2 is set up around the VRML scene to keep it from running close to text on the rest of the page.

Listing 31.2. The VRML billiard table embedded in an HTML page.

```
<HTML>
<HEAD>
<TITLE>VRML Billiards</TITLE>
</HEAD>
<BODY>
<EMBED SRC="pool.wrl" HEIGHT="500" WIDTH="500" ALIGN="RIGHT"  BORDER=2 >
<P>
You can play a game of 8 ball with another player.
<P>
Use the controls on the VRML window to move around the table to see your shot.
<P>
To take a shot:
<UL>
<LI>Drag to move the cue
<LI>Click on the pool table
<LI>Drag to angle the cue
<LI>Click on the pool table
<LI>Drag the cue back
<LI>Click on the pool table
</UL>
</BODY>
</HTML>
```


Multiple VRML scenes could be added in this fashion by adding more `<EMBED>` tags, similar to adding multiple `<IMG>` tags to an HTML Web page.

Adding a VRML Frame

To add a VRML scene as a separate frame, a frame definition document, as described in Chapter 18, is created to house the VRML scene. In Listing 31.3, three frames are created: A description frame named `DESC` shows narrative about the game of billiards, an index frame named `INDEX` provides a permanent list of different narrative documents available for reading from the `DESC` frame, and the VRML frame named `VRML` shows the VRML scene.

Listing 31.3. The HTML frame definition document.

```
<HTML>
<HEAD>
<TITLE>VRML Billiards</TITLE>
<FRAMESET ROWS="*,50">
<NOFRAMES>
</NOFRAMES>
<FRAMESET COLS="300,*">
<FRAME SRC="overview.htm" NAME="DESC">
<FRAME SRC="pool.wrl" NAME="VRML">
</FRAMESET>
<FRAME SRC="index.htm" NAME="INDEX">
</FRAMESET>
</HEAD>
</HTML>
```

The `INDEX` frame has a fixed height of only 50 pixels, which is sufficient to print a line of text. The `DESC` frame is 300 pixels wide and fills the rest of the browser window vertically based on the current window size. The `VRML` frame takes up the rest of the browser window, to the right of the `DESC` frame, dynamically obtaining whatever room is left over. The order of the `DESC` and `VRML` frame `<FRAME>` tags can be switched to move the VRML frame to the left of the `DESC` frame.

In that case, the VRML frame has a fixed width. Figure 31.4 shows the presentation of the billiard world in a frame definition document.

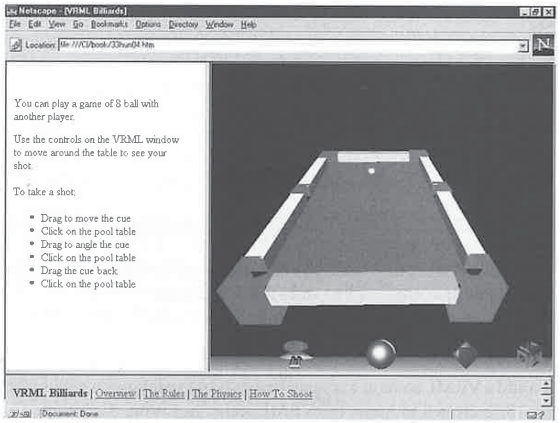


FIGURE 31.4.
Presentation of the billiard VRML world within an HTML frame definition document.

Communications Between VRML and HTML

After a VRML scene has been added to a frame definition document, the scene can communicate with the other frames from within the VRML file itself. The `Anchor` node within the VRML specification is responsible for setting up hyperlinks in a VRML file. An `Anchor` node can be added as a child node to a VRML `Transform` node in the same manner the `TouchSensor` node is added to the `Slate Transform` node in Listing 31.1. A new HTML document containing information on the rules of pool could be opened in the `DESC` frame when a visitor clicks on the slate of the billiard table by replacing the `TouchSensor` node of the Slate `Transform` node in Listing 31.1 with the following `Anchor` node:

```
Anchor {
    url [ "http://www.officialpoolsite.com/rules.html" ]
    parameter ["target=DESC"]
}
```

In this case, the `rules.html` document opens in the `DESC` frame when the geometry of the `Transform` node is clicked on by a visitor.

Any HTML hyperlink set up using an `<A></A>` tag pair can refer to a WRL file in its `SRC` attribute. When the hyperlink is clicked, the VRML scene is loaded on top of the current Web browser window contents. This technique is used when a VRML scene is intended to be explored by itself without any accompanying HTML text. VRML scenes encountered on the Web are usually presented in this manner because many Web surfers haven't yet set up a VRML viewer for use with integrated pages.

The Future of VRML

VRML fills a current need on the Web to present three-dimensional content that can be explored by a Web audience following a walking or flying analogy. HTML has nothing currently in its specification that formats information along those lines. VRML-based content appears to rival CD-ROMs, video games, and interactive television in terms of the market it addresses. But, VRML is integrated with the Web, and Web delivery introduces network latencies not acceptable for many audiences that use CD-ROMs and video games or watch television.

In this regard, VRML seems to be an extension of HTML as an information delivery vehicle. HTML syntax is similar enough to VRML syntax to entertain the thought of merging the two. VRML could refer to HTML content as a special form of texturing scrollable information on simple rectangular polygons. Web browsers could be created to parse one file containing nodes of both HTML and VRML types. This would be worth doing if a significant proportion of the population were willing to immerse themselves on the Web by wearing a special head-mounted device. Yet, there is little talk of merging the two, nor are there any attempts at mass producing virtual reality HMDs.

Instead, the future of VRML seems to be progressing in two directions: increased interactivity and multiuser worlds.

Increased Interactivity in VRML Scenes

As the example of creating a VRML billiard table shows, VRML files are very hierarchical, which allows for any node to be defined with the `DEF` keyword and be reused locally or across the Web. VRML viewer developers are rapidly improving external interfaces to their viewers that let a programmer access defined objects in the VRML file after the file has been loaded in the browser. The VRML community is pushing for standardization of a Java-based Application Programming Interface (API) that passes VRML nodes to Java classes as variables that can be manipulated from within a Java program. VRML 2 standardizes many simple object behaviors inside the VRML nodes and fields specification. VRMLscript adds even more capabilities. VRMLscript is a scripting language similar to JavaScript that is included in the WRL file through use of a `Script` node. It is different from JavaScript in that it is streamlined for VRML consideration only.

Because well-developed networking, scripting, and embedded physics classes already exist in Java, a Java API included in a VRML viewer lets these classes update VRML objects dynamically inside the Web browser. In the case of the billiard table example, the embedded physics classes can be associated with the billiard cue stick and the 16 balls to create a realistic VRML billiard game inside the VRML viewer. Although there are valid reasons to try instead to include these embedded physics within the WRL file itself, the standardization process takes some time while the Java solution is becoming more defined daily.

VRML is Web aware, so interactive VRML worlds could theoretically use information from all over the Net, including other VRML WRL files and databases accessed through Java database APIs. VRML worlds that change based on stock exchange, sports score, or planetary data are possible through a valid API.

Developing APIs include node methods such as `addChildren` and `removeChildren`, and browser methods such as `createVrmlFromString` and `createVrmlFromURL`, which allow new VRML objects to be added and deleted from the current scene on-the-fly based on other variables in the Java program.

Understanding the uniqueness of the Web provides an upper hand in creating new VRML scenes and Web pages that are interactive in ways that are different than today's popular CD-ROM, video game, and television show titles. So far, little has been done to create interactive pages that truly take advantage of the one-to-one connections of person to person, person to server, and server to server available on the Web. This is no surprise, given the history of other new technologies that struggled to define their uniqueness before capitalizing on it.

Multiuser Worlds Using VRML

One unique facet of the Web is being recognized more often in magazine articles and technology reviews: The Web is a powerful facilitator of virtual communities. Chat rooms dedicated to specific purposes where repeat visitors get to know each other are just the start of the realm of virtual communities on the Web.

VRML scenes that are stored on a server, allow multiple people to enter them simultaneously, and then pass 3D locations of each participant to each connected VRML viewer are called multiuser worlds. In many of these multiuser worlds, the location in the scene of each user is represented by a 3D model of a token being, called an avatar. Avatars are represented by talking heads, monopoly tokens, fully fashion-designed humans, and anything the imagination can dream up. Some form of text or voice communication service is provided so that the visitor can chat with others from within the world.

The biggest criticism of 3D modeled virtual communities is that the third dimension adds little value and slows things down appreciably. Sometimes the third dimension is just too literal and leaves less to the imagination. People like to imagine. Perhaps the third dimension is dismissed because the worlds being created do not present a powerful three-dimensional backdrop such as those provided for scientific visualizations. Or, perhaps, because current mass-produced technology is limited to one small world at a time being active in computer memory, the ability to navigate a large, interesting landscape continuously in 3D is not perceived as important only because it cannot currently be done effectively. VRML presents an amazing range of possibilities that are only now being explored. As people continue to develop their VRML skills, the worlds will get better. Add better communications capabilities and integrate some powerful Java scripting, and the sky's the limit.

Enough is being done with artistic VRML worlds to suggest that visiting a Web space with another person will be an exciting adventure in the future if not today. As the right interactions are added to the worlds, multiuser worlds may become even more popular. To finish the billiard arcade example, add eight or more `Billiards_Table Transform` nodes to the VRML scene, with artistically textured walls and ceilings, embedded physics and rules on all the cue sticks and balls, and avatar services enabled to be able to watch and communicate with anyone in the scene to play a game of billiards. This is a powerful vision of a probable future of VRML, possible because the physics are handled locally on each browser and not dependent on short latency communications from the server.

Summary

This chapter introduced a quick primer on VRML 2, the second specification of the Virtual Reality Modeling Language. VRML allows a Web author to exploit the third dimension in providing Web content that is experiential in a personal manner. VRML worlds are navigable by a user, becuase the user has control of her viewpoint and travels up, down, forward, back, right, and left to investigate the scene.

VRML skills are great to acquire as a complement to HTML skills. With both in hand, you can decide whether the third dimension will add or detract from your information presentation and create the best Web page for the occasion. If you already have extensive programming skills, VRML can be extended through scripting in Java to provide Web pages that appear more as applications than as static information presentations.

This chapter is by no means a complete reference of all the workings of VRML, but you have learned enough to get your feet wet. To learn more, open your favorite Web search engine and enter the keywords `VRML` or `Virtual Reality`. Plenty of new and interesting VRML sites are provided on the Web weekly.

CHAPTER 32

Plug-ins

by Mike Sessums

IN THIS CHAPTER

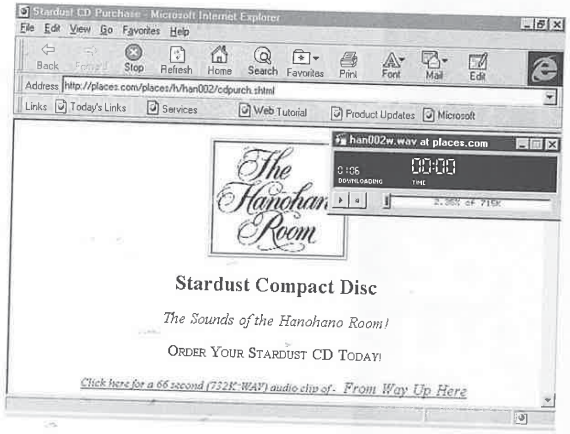
- Fads Versus Useful Tools 653
- Integrating Plug-ins 654

Browser plug-ins can increase your site's versatility and make it stand out from the rest. This chapter takes a look at the most popular plug-ins and some of those on the rise, as well as some of the tools you'll need to make them useful.

Third-party plug-in developers concentrate on the Microsoft Internet Explorer and Netscape Navigator, so I do the same.

Plug-ins are pretty amazing. They enable users to hear a new song at a Web music store before buying the CD. (See Figure 32.1.)

FIGURE 32.1.
An audio control pops up to play a demo clip from a new music CD.

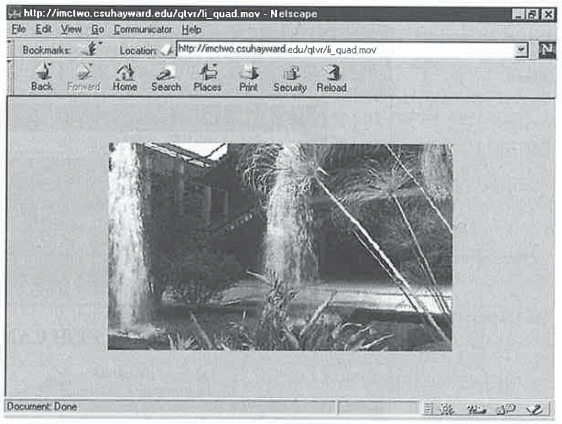


Plug-ins let potential students tour a college campus to check out the facilities without having to buy a plane ticket. (See Figure 32.2.) The possibilities are almost endless.

Plug-ins are also known as *add-ons* and *helper applications*. Many plug-ins are automatically installed, or plugged in, when Microsoft Internet Explorer or Netscape Navigator is set up. Others are available for plug-in at the Microsoft and Netscape browser Web sites, as well as some third-party sites.

There are around 90 plug-ins for Netscape Navigator, and the number is growing. Generally, plug-ins are made for one browser or the other, and are not interchangeable. However, Microsoft now boasts that you don't have to throw away those plug-ins just because you picked the "right" browser (meaning Internet Explorer). Through its ActiveX programming language, the Microsoft Internet Explorer will automatically check the user's hard drive for plug-ins installed on the Netscape Navigator as sites require them. (However, this feature doesn't always work.)

FIGURE 32.2.
Potential students can check out campus facilities using Surround Video.



Microsoft also says there are more than 1,000 ActiveX controls aimed at doing the same thing plug-ins do: delivering "active" content to the user. Movies and animations are examples of active content. An example of static content is standard HTML.

Fads Versus Useful Tools

You might think it would be difficult to keep up with the latest plug-ins, but actually only a few are widely used. The most popular ones are covered in this chapter.

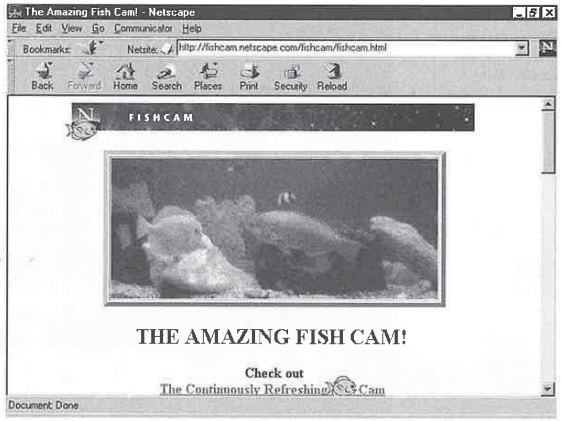
One way to find out whether a plug-in is genuinely useful or merely a fad is to check out what other sites are using. Of course, what constitutes a useful plug-in for one site might be totally inappropriate for another.

A good example of this is VRML. There is no doubt about it: VRML is cool. It's fun. On some sites, it is downright awesome. But here is the test: After playing around with a site's virtual world, what did you learn from it? I mean, what was it there for? Is it just to dazzle, or did it really serve a purpose?

Streaming video is another example. This one's not really so much a fad as it is technology that just isn't quite ready for prime time. The video quality is generally quite poor and choppy. If the technology catches up, streaming video will be the most ideal method for delivering large video content over the Web.

Live Web camera sites border on the fad category because many of them show the Web author's room or fish aquariums. (See Figure 32.3.) Oh, there's a lot of excitement there. Of course, many other people have discovered this technology's great potential.

FIGURE 32.3.
A very clever use of the Web camera: Netscape's Fishcam.



When you design your page, don't include the need for very many of the plug-ins that are out of the mainstream. Otherwise, your audience will have to download and install new plug-ins before they can take advantage of your Web site. An exception to this might be a very specialized audience, such as heart surgeons, who might require very specialized plug-ins that the public doesn't normally use.

Integrating Plug-ins

Browser plug-ins enable the Web audience to experience the content on your Web pages. However, plug-ins are only part of the picture.

Before you can integrate these plug-ins on your home page, you will need to create the content or locate stock items to include on your pages. These could include video, audio, animation, external applications, multilingual support, or Net communications, to name a few.

Video

Whether you want to use stock videos or digitize your own, there are several ways you can include video in your Web site. Video plug-ins (see Table 32.1) help make this possible.

Table 32.1. Video plug-ins.

Plug-in	*Platform*	*Company*
ActiveMovie	Windows 95, Windows NT	Microsoft (www.microsoft.com)
CineWeb	Windows 3.1, Window 95, Windows NT	Digigami (www.digigami.com)
ClearFusion	Windows 95	Iterated Systems (www.iterated.com)
InterVU Player	Windows 95, Windows NT, Mac 68K, PowerMac	InterVU (www.intervu.net)
LiveVideo	Windows 3.1, Windows 95, Windows NT	Netscape (www.netscape.com)
Maczilla	Mac 68K, PowerMac	Knowledge Engineering (maczilla.com)
Moviestar	Mac 68K, PowerMac, Windows 3.1, Windows 95, Windows NT	Intelligence at Large (www.ialsoft.com)
Net Toob Stream	Windows 3.1, Window 95, Windows NT	Duplexx Software (www.duplexx.com)
Quicktime	Mac 68K, PowerMac, Windows 3.*x*, Windows 95, Windows NT	Apple (www.apple.com)
RealVideo	Windows 95, Windows NT, PowerMac, UNIX	Progressive Networks (www.real.com)
VDOLive	Windows 3.*x*, Windows 95, Windows NT	VDONET (www.vdonet.com)
ViewMovie	Mac 68K	Ivan Cavero Belaunde (www.well.com/~ivanski/viewmovie/docs.html)
ViVoActive	Windows 3.*x*, Windows 95, Windows NT, PowerMac	Vivo Software (www.vivo.com)
Web Theater	Windows 95, Windows NT	Vxtreme (www.vxtreme.com)

The video standards for the World Wide Web are Video for Windows (AVI) and Quicktime (QTW). MPEG video has also gained some popularity. However, you're more likely to see videos saved as or converted to animated GIFs because this type is seeing wider use on Web sites.

You can include video links in your Web pages, or you can choose some of the more exciting methods of streaming video and live Web cameras.

The simplest way to add video is to just provide a hypertext link to your video file, in this case, an AVI:

```
<A href="video/yfronts.avi">Mr. Humphries</A>
```

The user clicks the link, Mr. Humphries, and the file called `yfronts.avi` starts downloading from the video directory. An external player starts up outside of the browser.

Streaming Video is a video-on-demand concept that holds some great potential. No longer do users have to wait to view a video. Streaming video starts playing after only a short portion is downloaded. Data continues to download while the user experiences the video.

Corporations such as Sybase are using RealVideo to deliver messages from their CEOs. (See Figure 32.4.) Their employees also use the RealPlayer plug-in as a company training device.

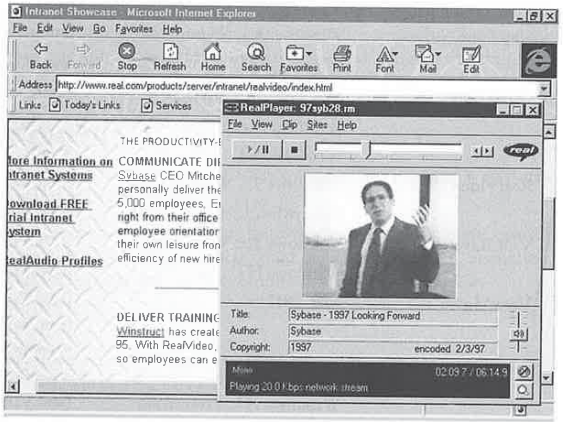


FIGURE 32.4.
Sybase CEO, Mitchell Kertzman, uses RealVideo to deliver his 1997 outlook message to employees.

The problem with plug-ins such as RealVideo is that the technology isn't quite there yet. Playback speed is considerably less than 30fps (frames per second), the normal speed required to give the appearance of motion. Many of the RealVideos on the Web right now resemble low-quality slide shows when viewed over regular telephone lines. If there is a lot of motion or are a lot of screen transitions, the viewer almost needs Dramamine.

For about $1000, you can get Progressive Networks' EasyStart Server to run your own streaming video and audio solutions. A free alternative is pseudo-streaming over a standard HTTP Web server. No additional software or hardware is required for this method. However, Progressive Networks calls it *pseudo-streaming* because, although you get streaming media, the quality is not optimized like it would be on one of their servers. That's a scary thought—especially when you view some of the bad video samples running from their servers. But the company insists that the pseudo-streaming should be good enough for personal Web sites running on a standard Web server.

If you go this last route, you will need to get your ISP system administrator to set up the RealVideo MIME types for the server. (You might also have him set up the RealAudio MIME types.)

After you have the server question solved, you're ready to add the streaming video to your home page. For this example, we'll link over to an existing RealVideo site by placing the anchor reference on our home page:

```
<a href="http://ramhurl.real.com/cgi-bin/ramhurl.cgi?ram=comea.rm">Dr. Katz</a>
```

When selected, this link executes a CGI script on the `ramhurl.real.com` server. The actual streaming video link is located in a metafile called `comea.rm`, and the RealPlayer plug-in is launched when it encounters the `.rm` file extension.

The Dr. Katz video (see Figure 32.5) has good timing and pacing that makes good use of RealVideo's shortcomings. But then, a good comedian is an expert in the art of timing. This video and the Comedy Central site are worth checking out.

RealVideo has strong support from studios showing trailers of upcoming movies, TV networks trying to give an ailing sitcom new life, and recording artists looking to boost sales and air time. The RealVideo guide (shown in Figure 32.6) lists more than 60 sites that take advantage of the RealPlayer plug-in's video capabilities.

Web cameras, or Web cams for short, use a live Internet link to provide "as it happens" video for the Web voyeur. The problem is, there usually isn't much happening.

Visitors to the Netscape Fishcam spend most of their time waiting for the fish to cross in front of the camera. When one finally does, there is a sense of elation similar to the rare sighting of Bigfoot.

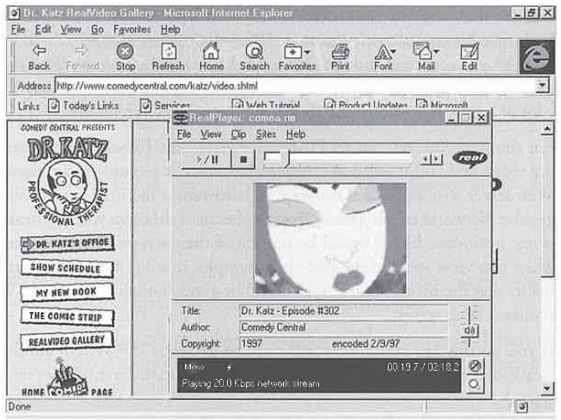


FIGURE 32.5.
Comedy Central's Dr. Katz RealVideo cartoon.

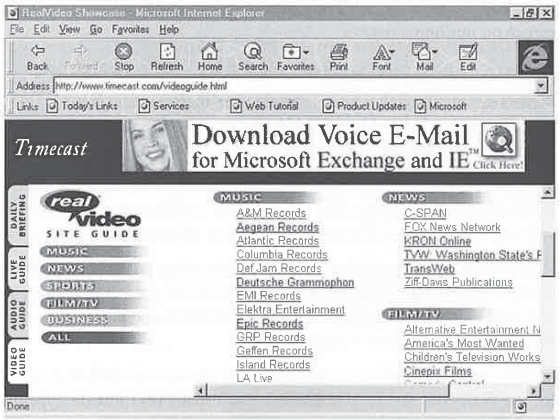


FIGURE 32.6.
The RealVideo video guide to program listings.

Many of the Web cameras are positioned on street corners, classrooms, and people's computer rooms. Some are used for scientific purposes in sort of a time-lapse photography method of observation.

One site that has an excellent use of the Web cam is the Interactive Model Railroad at the University of Ulm in Germany. (See Figure 32.7.)

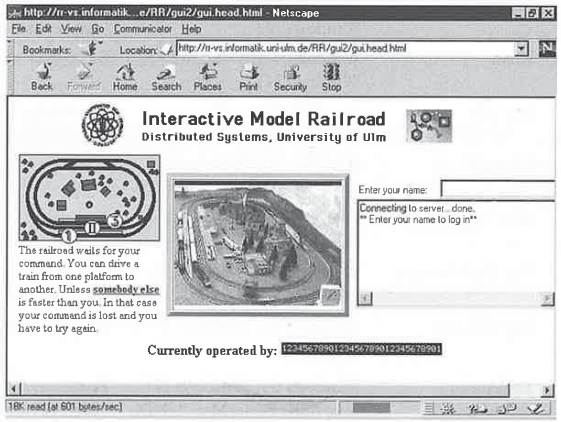


FIGURE 32.7.
A live Web camera lets armchair engineers operate model trains in Germany.

This great little site gives you a clickable map of stations so that you can tell the engineer to which station you want the train moved. You can also select which train you want moved. The video of the train layout refreshes every few seconds to show you the trains switching tracks and moving to the new destination station. Not even Gomez Addams could crash a train on this layout.

Audio

Audio is being used in a variety of ways. There is background audio that plays a music track or sound when a page is loaded. There are live and prerecorded broadcasts of nightly news and events. Table 32.2 shows some of the audio plug-ins available.

Table 32.2. Audio plug-ins.

Plug-in	*Platform*	*Company*
Crescendo Plus	Windows 3.1, Windows 95 Windows NT, Mac 68K	Liveupdate (`www.liveupdate.com`)
Digital Sound & Music Interface	OS/2	Julian Pierre (`www.polsci.wvu.edu/Madbrain/npdsmi.html`)

continues

Table 32.2. continued

Plug-in	*Platform*	*Company*
Echospeech	Windows 3.1, Window 95, Windows NT	Echo Speech (`www.echospeech.com`)
Koan	Windows 3.1, Windows 95, Windows NT	Sseyo (`www.sseyo.com`)
Maczilla	Mac 68K, PowerMac	Knowledge Engineering (`maczilla.com`)
Media Player	Windows 3.1, Windows 95, PowerMac, UNIX	Netscape (`www.netscape.com`)
MIDIPLUG	PowerMac, Windows 3.1, Windows 95	Yamaha (`www.yamaha.co.jp/english`)
Net Toob Stream	Windows 3.1, Window 95, Windows NT	Duplexx Software (`www.duplexx.com`)
RapidTransit	Windows 95, Windows NT, Mac 68K	Fastman (`www.fastman.com`)
RealAudio	Windows 95, Windows NT, Mac 68K, PowerMac, UNIX, OS/2	Progressive Networks (`www.real.com`)
WebTracks	Windows 3.1, Windows 95, Mac 68K	Wildcat Canyon Software (`www.wildcat.com`)

If you have a sound card, you already have the ability to record your own music and sounds for inclusion in your Web pages. Many sound cards, such as the SoundBlaster AWE 32, come with recording studio software. There are also many royalty-free audio clips already available on CD-ROM or on the Net.

The simplest way to take advantage of audio plug-ins is to provide a hypertext link to your sound file, in this case, a wave file (`.wav`).

```
<A href="audio/dangle.wav">Mrs. Slocombe</A>
```

The user clicks on the link, Mrs. Slocombe, and the file called `dangle.wav` starts downloading from the audio directory. An external player will pop up outside of the browser. (Refer back to Figure 32.1.)

If you wanted the controls embedded within the browser window (see Figure 32.8), you can use the `<EMBED>` tag, as follows:

```
<EMBED SRC=" audio/dangle.wav" AUTOSTART=TRUE LOOP=FALSE WIDTH=145 HEIGHT=55>
</EMBED>
```


The `EMBED SRC=` tag works just like the `IMAGE SRC=` tag and tells the browser where to find or search for the file. It always works in pairs and ends with the `</EMBED>` tag. The `AUTOSTART=TRUE` attribute tells the browser to play the wave file when it loads the page. If you change the attribute to `AUTOSTART=FALSE`, the user has to click the Play button to hear the sound. The `LOOP=FALSE` attribute sets the sound file to play through only once. The `WIDTH=145` and `HEIGHT=55` attributes tell the browser what pixel size to make the audio player's control bar. You could also add alignment attributes such as `ALIGN="CENTER"`.

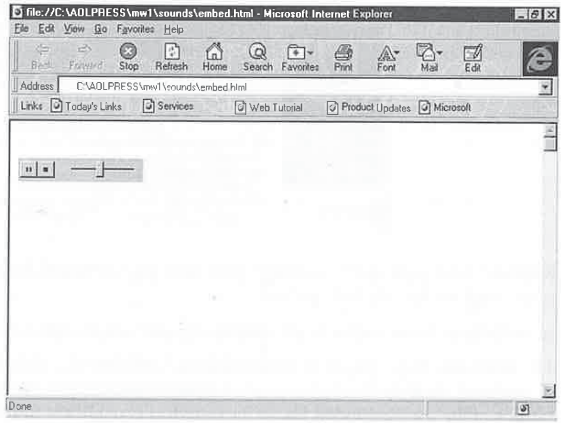


FIGURE 32.8.
This is how the Audio control appears when embedded within the Web page.

Background audio is also a relatively easy feature to integrate into your Web pages:

```
<bgsound src=totoro.wav loop=3>
```

This code calls out the background sound, `totoro.wav`, and plays it three times using the `loop=3` attribute. Please note that `<bgsound>` only works with Microsoft Internet Explorer.

TIP

Keep background sound files small to reduce your page's load time.

Streaming audio like RealAudio works the same way as its partner, RealVideo. And just like RealVideo, RealAudio files can be played using the RealPlayer. (See Figure 32.9.)

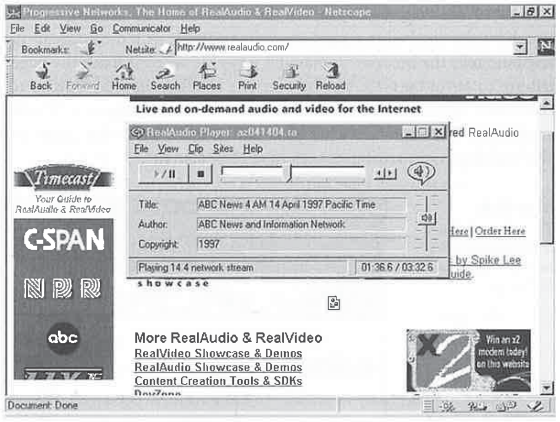


FIGURE 32.9.
Get the latest ABC Internet Hourly News at 15 minutes past the hour with streaming RealAudio.

RealAudio files are as easy to include in your Web pages as RealVideo. For this example, we create a link to the ABC Internet News:

```
<a href="http://www.real.com/contentp/abc24/ramfiles/az041612.ram"></a>
```

The RealAudio Player plug-in is automatically started when files with the `.ram` extension are encountered. Actually, this link is to a text metafile that contains the hostname and path or location of the `.ra` (RealAudio) file. In the preceding example, the RealAudio metafile (RAM file) is called `az041612.ram`. This file would only contain a single line of text, such as

```
http://www.startext.net/rafiles/news.ra
```

In this example, the RealAudio file is called `news.ra` and is located in the `rafiles` subdirectory on the Startext ISP server.

Top radio and news networks have joined the RealAudio bandwagon. At the RealAudio home page (`www.realaudio.com`), you will find information heavyweights such as C-SPAN, NPR, and ABC. You can even download a trial version of Progressive Network's EasyStart Server and the free RealAudio Encoder from the RealAudio site.

The RealAudio Encoder enables you to create streaming audio easily. I took a 4MB stereo file and used the Encoder to compress that down to 100KB. Total conversion time was less than five minutes. Even though the encoder saved the audio file as mono, the RealPlayer did some channel magic that still made the sounds in the new file shift around just like they did in the larger original file.

After you create the RealAudio file, you can use HTTP pseudo-streaming to deliver your audio over standard Web servers. (See the earlier discussion in the "Video" section about pseudo-streaming RealVideo.)

The free RealAudio Encoder 3.0 is available for Windows 95, Windows NT, Macintosh PowerPC, Linux, FreeBSD and various UNIX platforms.

Animation

People traditionally think of Mickey Mouse and Disney when anyone mentions animation. However, Web animation is a huge melting pot of traditional animation, 3D, and multimedia experiences. Plug-ins like those listed in Table 32.3 give Web site developers flexibility as never before.

Table 32.3. Animation, 3D, and multimedia plug-ins.

Plug-in	*Platform*	*Company*
AnimaFlex	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	RubberFlex Software (`www.rubberflex.com`)
Apple Electrifier	PowerMac	Lari Software (`www.electrifier.com`)
Bubbleviewer	Windows 3.1, Windows 95, Windows NT, PowerMac	Omniview (`www.omniview.com`)
Cosmo Player	Windows 95, Windows NT, UNIX	Silicon Graphics (`www.sgi.com`)
CyberAge Raider	Windows 95, Windows NT	CyberAge Communications (`www.miint.net/cyberage`)
CyberHub Client	Windows 95, Windows NT	Black Sun Interactive (`www.blacksun.com`)
DeepV	Windows 95	Heads Off (`www.headsoff.com`)
Enliven Viewer	Windows 95, Windows NT	Narrative Communication (`www.narrative.com`)
Flying Carpet	Windows 95, Windows NT	Accelgraphics (`www.accelgraphics.com`)
HyperStudio	Windows 95, Mac 68K, PowerMac	Roger Wagner Publishing (`www.hyperstudio.com`)
IconAuthor	Windows 3.1, Windows 95	Aimtech (`www.aimtech.com`)

continues

Table 32.3. continued

Plug-in	*Platform*	*Company*
Jutvision	Windows 95, Windows NT	Visdyn Software Corp. (www.visdyn.com)
Live3D	Windows 3.1, Windows 95, PowerMac	Netscape (www.netscape.com)
mBED	Windows 95, Windows NT, Mac 68K, PowerMac	mBED (www.mbed.com)
Multimedia Home Space Viewer	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	Paragraph International (www.paragraph.com)
Surround Video	Windows 95, Windows NT, PowerMac	Black Diamond (www.bdiamond.com)
NetMC	Windows 95, Windows NT	NEC Systems Laboratory (netmc.neclab.com)
OLiVR Viewer	Windows 95, Windows NT, PowerMac	OLiVR (www.olivr.com)
RealiView	Windows 95, Windows NT	Datapath Limited (www.realimation.com)
RealVR	Windows 95, Windows NT, PowerMac	RealSpace (www.rlspace.com)
Shockwave	Windows 3.1, Windows 95, Windows NT, Mac 68K	Macromedia (www.macromedia.com)
Shockwave Flash	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	Macromedia (www.macromedia.com)
Sizzler	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	Totally Hip Software (www.totallyhip.com)
SmoothMove	Windows 95, Windows NT	Infinite Pictures (www.smoothmove.com)
TopGun	Windows 95	7th Level (www.7thlevel.com)
Viscape	Windows 95, Windows NT	Superscape (www.superscape.com)
VR Scout	Windows 3.1, Windows 95, Windows NT	Chaco Communications (www.chaco.com)

Plug-in	*Platform*	*Company*
Vrealm	Windows 95, Windows NT	Integrated Data Systems (www.ids-net.com)
Web-Active	Mac 68K, PowerMac	Plastic Thought (www.3d-active.com)
WebXpresso	Windows 95, Windows NT, UNIX	Dataviews (www.dvcorp.com)
Whurlplug	PowerMac	Apple (www.apple.com)
WIRL	Windows 95, Windows NT	VREAM (www.vream.com)

Three of the most popular animation and interactive plug-ins are Shockwave, Surround Video, and WIRL.

Shockwave

The Shockwave plug-in enables users to experience "shocked" pages. (See Figure 32.10.) Shocked sites can have interactive animation, audio, and multimedia. The newest version of the Shockwave plug-in also gives the Web site visitors all this in a streaming format. AOL, Pointcast, and Marimba's Castanet are just some of the more well-known sites that contain Shockwave content.

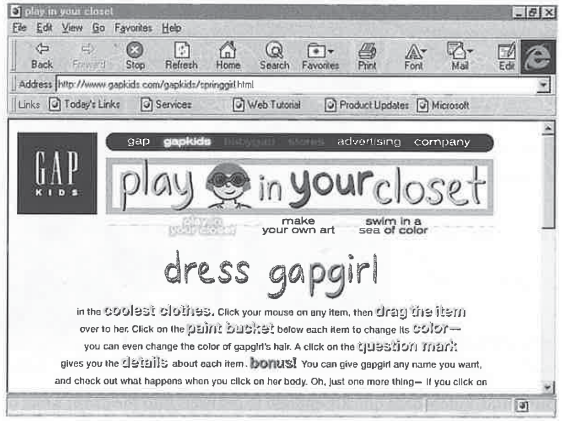


FIGURE 32.10.
This shocked page lets kids play dress up with gapgirl.

Director from Macromedia (`www.macromedia.com`) is one of the most popular methods for creating high-quality streaming interactive multimedia for games, interactive presentations, and CD-quality audio and movies for shocked Web pages.

One problem with Shockwave presentations on shocked pages is they usually take so long to load that you've lost your audience. Most people won't wait five minutes for a page to load, and I've found most pages with Shockwave movies take much longer than that. Shockwave has excellent potential, but it is best to keep the "shocking" simple and the files small.

Surround Video

Surround Video is a rising star in the 3D video arena. So why isn't this plug-in listed in the video category? Well, Surround Video works more like a photo-realistic 3D world viewer or animation than a true video.

Black Diamond (`www.bdiamond.com`) has created the Surround Video SDK (software developers kit) that can create 360-degree interactive panoramic video from any film-based, digital, or 360-degree camera. Surround Video also supports Visual Basic and JavaScript for added flexibility. Many educational and commercial groups have already found an application for this new technology. At CarPoint, potential buyers can use the Surround Video plug-in to check out the luxurious interior and instrumentation of a BMW. (See Figure 32.11.)

FIGURE 32.11.
An interactive Surround Video inside a BMW 5-Series at CarPoint.

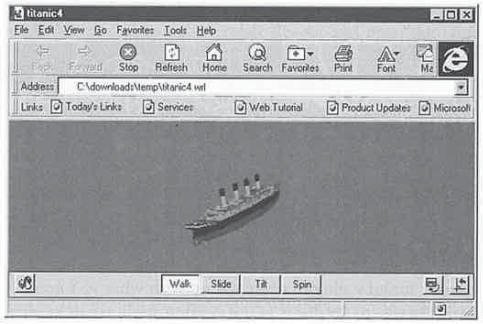


Surround Video loads quickly enough to get users into the driver's seat before they get a chance to surf on to the next site.

WIRL

Plug-ins like VREAM's WIRL immerse users in VR (Virtual Reality) Worlds. An excellent Titanic site created by Macedon Mediature Inc. (`www.mediature.com/titanic/`) has several VRML images of the Titanic. (See Figure 32.12.)

FIGURE 32.12.
VR plug-ins let users check out a virtual Titanic.



Getting VR worlds onto your home page is as easy creating a hypertext link:

```
<a href="vrworlds/titanic4.wrl">
```

The VRML plug-ins will launch a new browser frame or start up its own external viewer when it encounters the `titanic4.wrl` file. One of the most common file extensions for virtual worlds is `.wrl`.

Cosmo Player

One of the most popular VRML world viewers is Cosmo Player from Silicon Graphics (SGI). SGI really put its graphics technology background to work in the Cosmo Player and 3D world authoring tools.

Many VR worlds are limited to a clunky environment with the occasional 2D photo and HTML site link. However, one slick feature of the Cosmo Player and the Cosmo VR creation tools is the capability to create worlds containing 3D spatialized audio. Video can also be embedded into these virtual worlds.

Unlike some VR viewers, the Cosmo Player's constant frame rates translate to smoother interaction with large VR worlds. A few other viewers appear to have problems even at the most simple object level. Count in Java support with the package, and you walk away with some awesome authoring possibilities.

Animated GIFs

Some tools can convert AVIs to Animated GIFs, such as GifBuilder 0.4. This freeware package enables Macintosh users to create animated GIFs from a variety of input. It can import PICT, GIF, TIFF, or Photoshop images and convert Quicktime movies. Smartdubbing is another freeware Mac package for converting Quicktime movies to animated GIFs. GIF Construction Set is a great animated GIF authoring tool for Windows.

NOTE

Animated GIFs play back at a higher rate on the Microsoft Internet Explorer than on Netscape Navigator. Consider which browser you want your audience to use when creating animated GIFs. Also, be sure to include one of those "This site best viewed with..." statements so that the animation will play back as intended.

External Applications

Still haven't found a plug-in to match up with what you need? Maybe you should roll your own. An easy way to do this is to use existing applications as plug-ins.

Almost any software application can be used as a plug-in for graphical browsers. Good examples are Microsoft Word and Microsoft PowerPoint files.

Say I have a family history database that was created in Family Tree Maker for Windows. I want to share this database with other users of the same software package. To do this, I would just copy the Sessums data file to my Web site and provide a link from my home page, such as this:

```
<a href="tree/sessums.ftw">Kader Sessums database for Family Tree Maker</a>
```

With the user's browser properly configured, Family Tree Maker for Windows is opened when the browser encounters a file with the `.ftw` extension.

Microsoft's Internet Explorer is more application savvy than Netscape Navigator. Most registered file types, such as `.doc` for Word or `.ppt` for PowerPoint, will make Internet Explorer 4.0 launch the appropriate application without asking the user any questions. Netscape Navigator, however, requires the user to configure these helper applications manually.

We can't leave a discussion on application plug-ins without looking at the most widely accepted one, Adobe's Acrobat Reader. Many documents are being converted to Adobe's Portable Document Format (PDF) for presentation and delivery on the Web. The point of PDF is to let documents keep the same look and feel as the original document. Images, layout, and fonts stay intact.

One easy way to get your documents into PDF is to save them as PostScript. You can then use a product such as Distiller to convert the PostScript files into PDF. These files are then viewed with the Acrobat Reader plug-in from Adobe (`www.adobe.com`).

To include an Acrobat file in your home page, simply make a link to your PDF document:

```
<a href="documents/nascar.pdf">Nascar Season Stats</a>
```

In this example, the `nascar.pdf` file would launch Acrobat Reader and open the file in my `documents` directory. My Web visitor could now view my Nascar Season Stats.

So what is the benefit of serving up your existing documents as PDF rather than in their native file format? The biggest advantage is universal acceptance.

Not every user will have the software package in which you created your file. Most will not run out and purchase the software just to view it. On the other hand, if your document were available in PDF, the user could view it with the free Acrobat Reader. The reader is available for Windows 3.1, Windows 95, Windows NT, OS/2, and various flavors of UNIX.

Multilingual Support

Some sites need to publish their information in languages other than English. Companies looking toward world trade and individuals looking for rich Web content in their own language can all benefit from multilingual Web pages.

However, the special characters involved in some languages can present a real problem. Luckily, Microsoft Internet Explorer 4.0 and Netscape Navigator 4.0 include multilingual support, so a Japanese Web site looks like Japanese and not gibberish.

The Microsoft Internet Explorer 4.0 supports 98 language character sets from Afrikaans to Zulu. I never realized there were 19 varieties of Spanish! Compare this to Internet Explorer 3.0, which only supports five language character sets: Chinese (simplified and traditional), Japanese, Korean, and Pan European. Users also are required to install the separate Multilingual Support packs just like they would regular plug-ins for earlier versions of Netscape.

Netscape Navigator 4.0 supports only 10 language character sets. These include Western (Latin), Central European, Japanese, Chinese, Korean, Cyrillic, Greek, and Turkish. The editor on Netscape Navigator Gold even supports multibyte input for creating Japanese, Korean, and Chinese pages.

Figure 32.13 shows what happens when your browser doesn't support a Web site's character set.

Figure 32.14 shows what you will see when your browser is savvy enough to speak the language.

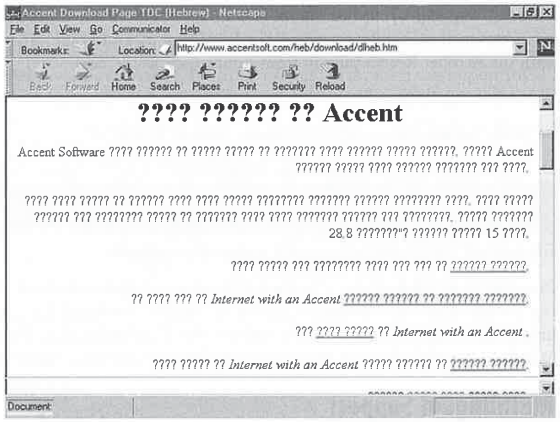


FIGURE 32.13.
Netscape Navigator 4.0 doesn't support Hebrew.

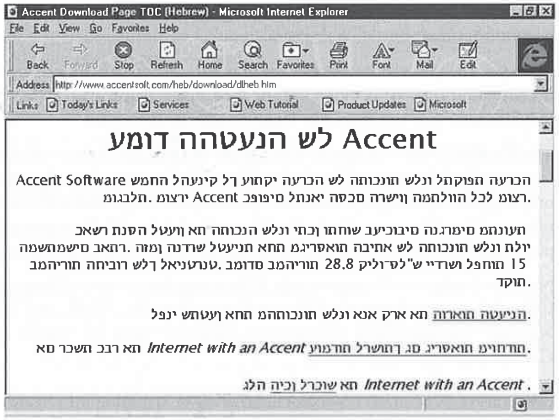


FIGURE 32.14.
Internet Explorer 4.0 is at home even with Hebrew.

A possible solution for the Netscape Navigator problem would be to seek out yet another plug-in by a third-party software group. Navigate with an Accent, available from Accent Software (`www.accentsoft.com`), extends Netscape Navigator's language savvy from 10 to 30. Quite a nice add-on for the world traveler. It even includes Hebrew.

Now that you've found the multilingual plug-ins, you need to create some content to take advantage of them. One Web authoring package that lets you do this is Internet With An Accent by Accent Software. Its Accent Global Author enables you to create Web pages in 30 different languages. Of course, you have to be able to write and understand those languages.

NOTE

Although Accent Global Author is designed to work under any language version of Windows 3.1 or Windows 95, it has only been tested under English and Hebrew versions of these operating systems. Web authors using other language versions might want to download the free demo from Accent Software before making a commitment.

Web authors might find translation software such as Power Translator 6.0 from Globalink very handy. This package handles translation of English, French, German, Italian, and Spanish at up to 20,000 words per hour. Thank goodness most Web sites aren't that wordy! Be aware, though, that translation software like this uses templates for translation and might not be 100 percent accurate. The only way to be sure is to be somewhat familiar with the language to which you are translating.

After you get your alternate page created, you'll want to create a link on at least your home or top-level page that enables users to quickly switch to the language of their preference. Accent Software uses national flags as clickable links to make language identification and navigation easy. (See Figure 32.15.)

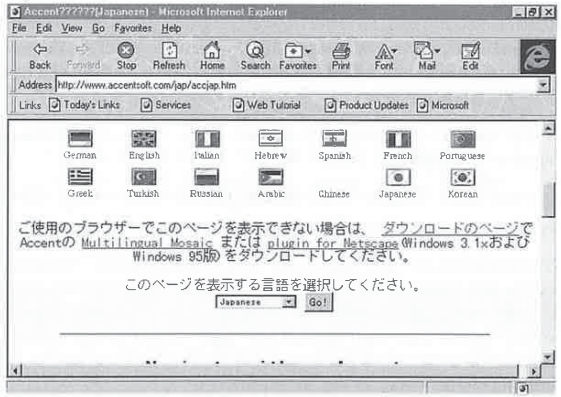


FIGURE 32.15.
Users click a flag to get a duplicate of this page in that country's native language.

By providing a multilingual site, you will not only increase the audience for your message or product, you will gain appreciation from users whose primary language is not English.

Net Communications and Power Meetings

A new level of technical support and site interaction has arrived, thanks to Net communication tools. Tables 32.4 and 32.5 show a few of the more popular communication plug-ins.

Table 32.4. Video teleconferencing.

Plug-in	*Platform*	*Company*
CU-SeeMe	Windows	Cornell University (`cu-seeme.cornell.edu`)
NetMeeting	Windows	Microsoft (`www.microsoft.com`)
VideoPhone	Mac, Windows	Connectix (`www.connectix.com`)
WebPhone	Windows, OS/2	NetSpeak (`www.netspeak.com`)

Table 32.5. Net phones and text conferencing.

Plug-in	*Platform*	*Company*
OnLive! Talker	Windows 95	OnLive Technologies (`www.onlive.com`)
PhoneFree	Windows 95, Windows NT	Big Bits Software (`www.phonefree.com`)
PowWow	Windows, OS/2	Tribal Voice (`www.tribal.com`)

Tribal Voice, the developer of PowWow, uses its chat tool to offer interactive, live technical support. (See Figure 32.16.)

It is easy to add Web phone and chat capabilities to your own site with PowWow. Here's how:

```
<a href="powwow:msessums@fastlane.net">Call Me on PowWow!</a>
```

This works like any other link, except the user links to `powwow:msessums@fastlane.net` instead of the usual file location. This portion tells the PowWow plug-in to look up user `msessums@fastlane.net` on the PowWow server. If `msessums@fastlane.net` is logged on to the server, the PowWow client software will "page" him. When he answers the page, chatting can begin. If the target user is not logged on to the PowWow server, the user who clicked the chat link will receive a `User is not online` message.

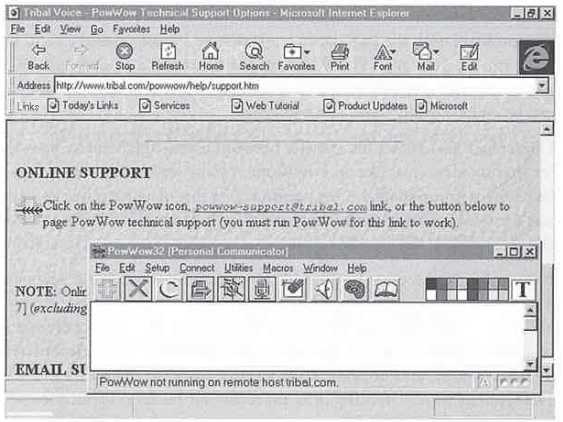


FIGURE 32.16. *PowWow users can get Web phone (voice) support anywhere in the world.*

TIP

You might want to include a note that PowWow must be started before Netscape Navigator or Microsoft Internet Explorer for the chat link to work.

Network administrators who want to run a private PowWow server on their corporate intranet or from behind their firewall can use the PowWow User Location Server (PULS). This server, which runs on Windows NT networks, costs as much as the regular PowWow client software—it's free.

PowWow users on the public Internet don't see it, but when they start up a PowWow session they are connected to a public PowWow server. This server acts as a transparent-to-user locator service that registers a user's IP address in a look-up table each time the user logs on. A white pages listing at Tribal Voice lets users look up the last 100 users who started up (logged on to) PowWow. Users do one of the following: click the name of the party from the PowWow listing, click a PowWow link from a home page, or type an e-mail address.

NOTE

PULS is only required to run the PowWow client software on intranets. You don't need PULS to take advantage of PowWow's registration and lookup feature if you are running on a public network. For most Web developers, PowWow is the only software that is needed.

Conference, included in Netscape Communicator 4.0, lets you talk by either voice or keyboard. The problem with the keyboard chat is that you must press the Send button before the other party can see your text. PowWow, on the other hand, allows anyone to whom you are connected simultaneous viewing of any text you type. No Send button to press. Of course, this can be embarrassing if you're a bad typist!

Conference also allows file transfer between users, collaborative browsing, and a white board for sharing ideas (just like on PowWow). By the way, *collaborative browsing* is just a white collar term for group surfing. One person acts as sort of a tour guide and takes control of everyone else's browsers during the surfing session. They can still talk by voice or keyboard chat while they are surfing. The PowWow plug-in is great for this type of interaction, and they did it first.

Microsoft's NetMeeting features file transfer, white board, and keyboard chat, just like Tribal Voice PowWow and Netscape Conference. It also uses directory servers to keep track of users' addresses, similar to those of the PowWow server. But what really sets it apart from the other conference plug-in tools is that users can experience video teleconferencing, share Windows 95 applications, and do some collaborative work on shared files. These features alone make NetMeeting worth checking into, especially for business applications.

CU-SeeMe was one of the first video teleconferencing tools and is still very popular. Combine the freeware or shareware versions of this software with the Connectix QuickCam and video couldn't be easier.

Users can hold a conference with up to eight CU-SeeMe users at a time. The conference display is reminiscent of "Hollywood Squares," where the player boxes are stacked on top of each other.

CU-SeeMe uses reflector sites to connect users. They can also connect one on one if they know the other user's IP address.

Connectix also has its own conferencing software, VideoPhone 2.0. This tool has some basic features of NetMeeting (although Connectix had it first). Users have a shared white board and can use direct-dial connections.

Web browsers can be configured to automatically start Connectix VideoPhone or Connectix VideoPhone Viewer when they encounter files with the `.cvp` extension on a Web page.

Other Plug-ins

Other plug-ins are available for various image types (see Table 32.6) and specialized business and professional applications. (See Table 32.7.) These tables in no way constitute a definitive list. Many more plug-ins exist and are being registered with Netscape and Microsoft all the time.

Table 32.6. Image plug-ins.

Plug-in	*Platform*	*Company*
ABC QuickSilver	Windows 95, Windows NT	Micrografx (`www.micrografx.com`)
CMX Viewer (vector graphics viewer)	Windows 95, Windows NT	Corel (`www.corel.com`)
CPC View (TIFF, PBM, and CPC viewer)	Windows 95, Windows NT	Cartesian Products (`www.cartesianinc.com`)
CyberSleuth (displays image author and other data)	Mac 68K	Highwater FBI (`www.highwaterfbi.com`)
Dr. DWG Netview (AutoCAD image viewer)	Windows 3.1, Windows 95, Windows NT	Dr. DWG (`www.cswl.com`)
DWG/DXF (AutoCAD image viewer)	Windows 95, Windows NT	Softsource (`www.softsource.com`)
Fractal Viewer (true-color image viewer)	Windows 3.1, Windows 95, Windows NT, Mac 68K	Iterated Systems (`www.iterated.com`)
Imaging for Internet (FlashPix viewer)	Windows 95, Windows NT	Hewlett-Packard (`www.hp.com`)
InterCAP Inline (CGM viewer)	Windows 95, Windows NT	InterCAP Graphics Systems (`www.intercap.com`)
Lightning Strike (wavelet image codec)	Windows 3.1, Windows 95, Windows NT, Mac 68K, UNIX	Infinop (`www.infinop.com`)
LuRaWave (wavelet image viewer)	Windows 95, Windows NT, PowerMac	LuRaTech (`www.luratech.de/luratech_e.html`)
MetaWeb CGM Viewer (CGM viewer)	Windows 3.1, Windows 95, Windows NT	Ematek/HSI (`www.ematek.com`)

continues

Table 32.6. continued

Plug-in	*Platform*	*Company*
NetWriter Viewer (view NetWriter images)	Windows 3.1, Windows 95, Windows NT	Paragraph International (www.paragraph.com)
Pegasus (JPEG viewer)	Windows 3.1, Windows 95	Pegasus Imaging (www.jpg.com)
PNG Live (PNG viewer)	Windows 95, Windows NT, PowerMac	Siegel & Gale (www.siegelgale.com)
SVF (CAD viewer)	Windows 95, Windows NT	SoftSource (www.softsource.com)
TruDef (views 12 graphics formats)	Windows 3.1, Windows 95, Windows NT	TruDef Technologies (www.trudef.com)
ViewDirector (TIFF viewer)	Windows 95, Windows NT	TMS Inc. (www.tmsinc.com)
Visual WebMap 2D (CAD, IGDS, DGN, and GIS map viewer)	Windows 3.1, Windows 95, Windows NT	Project Development (hem.passagen.se/project)
Watermark Webseries (TIFF viewer)	Windows 95, Windows NT	FileNet (www.filenet.com)
Wavelet Image Viewer (Wavelet viewer)	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	Summus Ltd. (www.summus.com)
Whip! (AutoCAD viewer)	Windows 95, Windows NT	Autodesk (www.autodesk.com)

Table 32.7. Business and professional plug-ins.

Plug -in	*Platform*	*Company*
AboutPeople (dynamic address book)	Windows 95, Mac 68K, PowerMac	Now Software (www.nowsoft.com)
AboutTime (dynamic calendar)	Windows 95, Mac 68K, PowerMac	Now Software (www.nowsoft.com)
Autodesk MapGuide (vector-based mapping solution)	Windows 3.1, Windows 95, Windows NT	Autodesk (www.autodesk.com)

Plug -in	*Platform*	*Company*
Carbon Copy/Net (remote Windows access)	Windows 3.1, Windows 95	Microcom (www.microcom.com)
Chemscape Chime (views 2D/3D chemical structures)	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	MDL Information Services (www.mdli.com)
Citrix Winframe Client (Windows apps interface)	Windows 3.1, Windows 95, Windows NT	Citrix (www.citrix.com)
Day-Timer Organizer (electronic organizer)	Windows 3.1, Windows 95	Day-Timer Technologies (www.daytimer.com)
EarthTime (world time)	Windows 95, Windows NT	Starfish Software (www.starfishsoftware.com)
Envoy (document viewer)	Windows 3.1, Windows 95, Windows NT, Mac 68K, PowerMac	Tumbleweed Software (www.tumbleweed.com)
Intermind Communicator (electronic communi-cations automation)	Windows 3.1, Windows 95, Windows NT	Intermind (www.intermind.com)
Netopia Virtual Office (collaborative Web office)	Windows 3.1, Windows 95, Windows NT	Farallon (www.farallon.com)
Nobelnet Opener (Web distribution for apps)	Windows 3.1, Windows 95	Noblenet (www.noblenet.com)
PointCast Network (receives news and information)	Windows 3.1, Windows 95	PointCast Inc. (www.pointcast.com)
Quick View Plus (manipulates more than 200 file formats)	Windows 3.1, Windows 95, Windows NT	Inso Corp. (www.inso.com)
ScriptActive (active scripting solution)	Windows 95, Windows NT	Ncompass (www.ncompasslabs.com)
Surfbot (auto Web site content retrieval)	Windows 95, Windows NT	Surflogic (www.surflogic.com)

Summary

You've seen that a multitude of plug-ins support just about every conceivable Web application. It's even possible to set up various programs as plug-ins if you still can't find the one you need.

Although it would be impossible to keep up with all the plug-ins and controls available, you've also seen that only a few have gained wide acceptance.

One thing to remember when developing your Web site is don't use a feature supported by plug-ins simply to use it. Plan out what you want to do on your Web site, and then check to see whether there are plug-ins and creation tools to make your plans a reality.

CHAPTER 33

HTML and Databases

by Michael A. Larson

IN THIS CHAPTER

- Lotus Approach 97 *681*
- Microsoft Access *682*
- Sybase *683*
- SQL Server *684*
- Oracle *684*
- IBM DB/2 *685*
- Centura (formerly Gupta) *685*
- Computer Associates International *686*
- Informix *687*

With the popularity of the Internet and the rise in use of intranets, many people, individually or in a variety of institutions, are trying to find the best means to take the massive amount of information they have in their databases and make it available to people via a browser. These people run into a problem because most databases, both large and small, were never designed to dish out HTML pages or work with a Web server. Fortunately, a wide range of solutions are available to meet the wide range of needs—from someone on a standalone computer converting database forms, reports, and queries to HTML all the way up to the enterprise level where information systems (IS) personnel have to port very complex, standalone mission-critical applications over to a Web server/browser environment.

HTML is, unfortunately, of little help to database engineers. The language was not designed with database integration in mind. The closest database functionality you can find in HTML proper is the table structure of rows (records) and columns (fields). Another structure, HTML forms, also mimics some of the capability of a database form but with far less functionality and flexibility, and often requiring a whole lot more non-HTML programming to perform simple database-like tasks such as entering the contents of an HTML form into a database. The only other benefits of HTML in relation to databases are that it is a simple set of tags and its file format is straight ASCII, which most databases can easily write to.

One means of communication between databases and HTML that is widely used today is the common gateway interface (CGI). Many lines of CGI script, however, are needed to link each field in an HTML form to a database. This requires a great deal of programming and any modifications can be very labor intensive. This particular problem might be one of the main reasons for the slow integration of databases onto Web sites by commercial and business enterprises.

So you end up with a gulf between databases (many originally engineered to be proprietary, closed systems) not designed to be used in the open environment of the Web and a Web language that was never designed to be used with databases. The means by which many vendors have chosen to fill this gulf is the subject of this chapter.

As I mentioned earlier, there is a broad gradient of problems and needs when an individual or an enterprise thinks about putting database information on a web. Some common problems discussed in this chapter include

- How to get information from established database forms, reports, or queries into the HTML format—at a minimum, into an HTML table and preferably into a customized, more attractive format, such as those typically generated with database report modules.
- How to dynamically query a database from a Web browser and return the results in real-time in a format the browser can read, which is usually HTML.
- How to port a proprietary, enterprise-wide database application into a Web environment.
- Figuring out how to use the Web as a cross-platform tool for tying together the hodgepodge of database systems and platforms typical of many enterprises today. In this case, the problem is actually turned to become the solution.

If you have done any database programming at all, you know that solutions to any of the preceding problems have to be accomplished with a high level of security and confidence that the information in any given database will not be compromised, corrupted, or accessed without permission. Internet proponents tout the need for the ultimate in data availability: Make all information available all of the time to everyone. Database engineers have a very different requirement, often imposed by the institution owning the database: Maintain the integrity of databases that may contain decades of valuable information, and make information available only to those who have the necessary clearance to view it. Bringing these two opposing approaches together in a harmonious solution is a significant challenge, requiring unique approaches and development tools.

I discuss these integration problems on a vendor-by-vendor basis. Each vendor has a solution that addresses all or part of the gradient of problems for getting database information onto a web. Most vendors not only offer a solution for their particular databases, but also one that is inclusive of a wide variety of other databases, database types, and platforms. I discuss the databases and solutions only as they relate to getting database information to HTML and to a lesser degree, to using Java applications to establish communication between the database and browser. Due to both the breadth and depth of currently available database applications and the relative newness of trying to get database information into HTML, I can only cover a few select databases and vendor solutions. My criteria for selecting which vendor solutions to review include the following:

- Market share. Databases that are used by the most people or businesses.
- Increasing market presence. Database solutions that appear to be gaining on the more popular systems.
- Unique technology or approach to tying databases and the Web together.

Be sure to do your own investigations if this is a critical component of your mission. This is by no means the final word. Databases and HTML are a moving target. HTML is also only one component of a web solution involving databases. The solution you choose should also incorporate any other individual or enterprise requirements of the database information.

Lotus Approach 97

This database component to Lotus SmartSuite (`http://www2.lotus.com/approach.nsf`) has some basic and some very advanced HTML publishing capabilities. For the basic capability, this database represents the simplest solution: Any Approach form, report, table, or chart can be saved to HTML without requiring you to know HTML coding. This is called a static solution because it involves only the single step of translating database objects to HTML. The HTML output will usually be in a less elegant form than its equivalent database object because HTML has fewer formatting options and less flexibility than is built into most databases.

The advanced publishing capabilities in Approach are closely linked with Net.Data, a very powerful Internet tool associated with the IBM DB/2 database; Net.Data is discussed later in this chapter.

Approach also has a built-in Web site database where users can store their favorite URLs as hyperlinks and associate them with keywords of their choosing for easier searching.

Microsoft Access

In its latest version, Access 97 (`http://www.microsoft.com/access/`) contains a variety of means for getting information from an Access database to a browser. Some solutions work closely with Microsoft's Internet Information Server (IIS), which is related to Peer Web Services on Windows NT Workstation and Personal Web Server under Windows 95, and some do not require a Web server to work.

Access uses two basic approaches to getting database information to Web users: static and dynamic. The *static approach* involves using the Save as HTML wizard from the menu in Access 97 to translate Access reports, queries, and forms to HTML tables. You can insert tables into HTML template files with the use of a special placeholder tag in the template file showing Access where the HTML table should be placed. With the addition of the new Hyperlink data type in Access 97, you can now store active hyperlinks in an Access database. These hyperlinks remain active in native Access objects such as forms and reports or when any Access objects containing them are translated to HTML via the wizard. Access 97 also enables you to import HTML tables or lists (even over the Internet) into existing Access tables, or you can create new tables. You can also link to HTML table data if you don't want to import the data into your database.

Microsoft has two technologies that work in conjunction with its Web server to provide a more dynamic and real-time interaction between a user with a browser and an Access database. The older of these two technologies is called IDC/HTX and the newer is ASP.

IDC is short for Internet database connector. HTX is merely an HTML file that has placeholders telling Access where to place the returned records. IDC/HTX works as a process involving many different parts. The IDC file is a plain-text file containing the data source and the SQL query. An IDC request is initiated from a browser via a request that looks like this:

```
http://MSWebservername/scripts/somefile.idc
```

The file causes a special library called `httpodbc.dll` to be loaded. This library loads the appropriate ODBC (open database connectivity) driver for the data source. The ODBC driver connects to the Access database, enabling the library to send the SQL query in the IDC file to the database. Access runs the query based on the content of the IDC file and sends the result back through the ODBC driver to the library. The library looks in the IDC file to determine which HTX file to use. The library adds the records at the placeholder locations in the HTX files and returns the now complete HTML file to the Web server. The server passes this file back to the browser that initiated the IDC request. This process is not limited to use with Access databases but can also be used with FoxPro, Oracle, and SQL Server databases. It even works with non-database sources such as Excel spreadsheets or delimited text files. As long as you have the proper ODBC driver, have the data source registered with the ODBC driver, and use the proper

syntax in the IDC file, you can get data from the data source to a browser. The IDC also provides a means of passing a username and password to the database if this is required.

Microsoft recently put out a similar, but more powerful, technology called ASP (active server pages). ASP uses a process similar to the IDC/HTX process but utilizes only one plain-text file (`.asp` extension) for both the requesting file and the HTML template return file. In addition to having to deal with only one text file, you can also add programming logic to your requests using either Visual Basic Script or JavaScript. You can also create server-side objects or add ActiveX controls. This is a much more powerful and flexible system than IDC/HTX. Again, this approach relies on Microsoft Web servers and will not work with any other type of Web server at this time.

These two dynamic approaches enable the database designer to add hyperlinks to a Web page pointing toward fixed queries that will return the latest, most commonly requested reports. Or you can couple these technologies with HTML forms to enable a user to specify one or more parameters for the query and have a special report returned to them.

The Access Save as HTML wizard will generate IDC/HTX file pairs or ASP files from Access queries or forms. This wizard can also start the Publish to the Web wizard that will publish an entire set of HTML-translated objects to the Web server, as opposed to saving them locally and then uploading them to your Web site via FTP.

Sybase

Sybase (`http://www.sybase.com`) uses a technology called NetImpact Dynamo (`http://www.sybase.com/netimpact/internetapps.html`) to Web-enable its own databases—Sybase Anywhere and Sybase SQL Server Professional—as well as many others. NetImpact Dynamo uses a dynamic approach for interacting with the database. SQL commands, scripts, and host-variable macros can be placed into any HTML file (called a template) for processing. NetImpact Dynamo also provides tools for HTML authoring, including wizards for producing templates, site management tools, and a Personal Web Server for offline use. The DynaScript language in NetImpact Dynamo is compatible with JavaScript and can be placed either in a module separate from the HTML page or in the HTML page.

When a browser requests an HTML file that is a template, the Web server passes the request to NetImpact Dynamo, which pulls the template from a template repository, processes any scripts or SQL commands via an ODBC source, and passes the results back to the Web server as an HTML page, which is then passed back to the browser.

NetImpact Dynamo supports multiple gateways including CGI, ISAPI (Internet Server Application Programming Interface), and NSAPI (Netscape Server Application Programming Interface) on Windows 95 or Windows NT platforms.

Sybase also has a similar technology for Sun Solaris, IRIX, HPUX, and Windows NT called web.sql. This program also works in a dynamic manner but does not require a gateway to the database; it is directly connected to the Web server and supports its own inline scripting for

improved performance. The HTML files processed by web.sql can contain SQL queries and perl5 scripts. The web.sql engine uses the Sybase Open Client API, which allows connections to a wide variety of major databases. CGI and NSAPI are also supported.

SQL Server

Microsoft's SQL Server (`http://www.microsoft.com/sql/inetOverview.htm`) dishes out its database information via SQL Web Assistant. The SQL Web Assistant takes a slightly different approach to publishing dynamic information. It uses a wizard-style interface to build and schedule a query. The data source can be selected from a table with a point and click interface, using an SQL statement, or from a stored procedure. The results of that query are then published on a schedule as frequently as needed or in response to data updates in the database. This means that you can publish your data very frequently—which is useful, for instance, in the case of stock quotes—or infrequently, maybe once a week as you might need in the case of summary financial reports. This approach works well for publishing large lists of information such as catalogs, price lists, or inventory.

SQL Server database information can also be placed into a Web environment using the same technology discussed earlier under Microsoft Access, namely using IDC/HTX or ASP dynamic queries in conjunction with ODBC and Microsoft Web servers. Note that the Web Assistant is more of a push model for distributing data, whereas IDC/HTX and ASP represent more of a pull model, providing multiple options for getting information from an SQL Server database into a form viewable by anyone with a browser.

Oracle

Oracle's solution for getting database information into HTML centers around its Developer/2000 report writing module (`http://www.oracle.com/support/products/dev2k/win/html/index.html`). This product can be used with Oracle databases or as a standalone application compatible with many other database systems on many platforms, including all flavors of Windows, Macintosh, and UNIX/Motif, as well as the new networking computers (NCs), also known as thin clients. Developer/2000 performs two Web-related functions: generating HTML output for any report (PDF output is also available) and processing HTML forms.

HTML reports are generated by simply telling Developer/2000 to output to the HTML format. Any chart can also be output as a GIF for use on an HTML page. As in Microsoft Access, any layout object can have a URL associated with it. In addition to generating HTML reports, Oracle's newer WebServer Version 3.0 contains the Web Library. This library is a repository for all Web site objects (HTML, images, Java applets, and so on) that provides security, version-stamping, and other site or file management features.

An HTML form built by Developer/2000 contains an embedded Java applet (the Forms Client) that is downloaded to the browser when the HTML page with the form is called. The Web server then serves as an intermediary between the Forms Client and the Oracle Forms runtime engine (the Application server). In an intranet situation, the Application Server communicates with the database via a remote procedure call (RPC) and back directly with the Forms Client. In an Internet situation, communication between the Web server and the application server is via CGI or a cartridge mechanism if the Web server is an Oracle product. Both of these situations provide a convenient means of using a browser to trade information directly with an Oracle database. This model is an Internet extension to the idea of building and deploying customized database applications, usually on an enterprise-wide basis.

IBM DB/2

Big Blue has updated and enhanced its DB/2 WWW Connection program and renamed it Net.Data (`http://www.software.ibm.com/data/net.data/`). Net.Data covers a wide range of functions, from creating dynamic Web pages via SQL to building enterprise-strength Web applications. Net.Data includes native support (read "faster performance") for DB2, Oracle, and Sybase as well as ODBC access to other data sources, including Lotus Notes, Microsoft databases, SQL Server, and Informix. Flat file data can also be accessed and updated. Net.Data supports APIs for IBM Internet Connection Web Servers (ICAPI), Netscape Servers (NSAPI), and Microsoft Internet Information Servers (ISAPI). Net.Data has several language environments from which to access DB2 databases—REXX, Perl, Java, or C++ applications—for maximum flexibility.

A key component to the cascade of events that Net.Data uses to obtain information from the database or applications is a Web macro. A Web macro is a combination of HTML, a macro language, and language environment-specific statements that can be from SQL, Perl, or REXX. An `.INI` file also is needed for details on your language environment, paths, and so on.

Several layers of security in Net.Data make it a very security-conscious environment for these types of database and browser interactions. Not only can you have authentication (username/password), but you can encrypt data with Secure Sockets Layer (SSL) or Secured HyperText Transfer Protocol (SHTTP). The system also works effectively behind a firewall.

Because you can call just about any application or library from within the Web macro as well as perform SQL queries, Net.Data is ideally suited for enterprise applications. It leverages your existing applications and databases, providing a straightforward means to make a wide variety of information available on the Internet or the company intranet. This package is well-suited for the professional programmer and might be daunting for the casual database user or programmer.

Centura (formerly Gupta)

The Centura Web Developer (`http://www.centurasoft.com/products/development/web_developer/`) can be used to build applications that generate static HTML pages, dynamic HTML pages, or full-blown business enterprise applications, which is the main focus of the package. It can use ODBC interfaces or use NSAPI or ISAPI across many different database types.

The Web Developer system requires a Web server, an application server, and a data server. After a Web Developer application has been built, debugged, and compiled, it is placed on the application server. A URL request from a browser for the application passes to the Web server, which passes the request to the application server, and the application is started. The application server returns the "address" of the application process to the browser via the Web server, and any subsequent interaction occurs directly between the browser and the application process. When the process is completed, results are returned to the browser, or another process is started.

Web Developer supports multiple levels of security, including authentication, encryption, and firewalls. Again, this package is geared toward professional programmers who need to produce complete business solutions that might include a Web component.

For optimal security, Centura recommends replicating your databases to an SQLBase that lives on the Web server outside the firewall to make your database information available to Internet users and leave your primary databases untouched behind a firewall.

Computer Associates International

CAI has two major solutions for deploying database information on a web: OpenIngres/Internet Commerce Enabled (ICE) (`http://www.cai.com/products/unicent/ice.htm`) and Jasmine (`http://www.cai.com/products/jasmine.htm.bkp`) databases. As with many other packages, solutions developed with these systems will work across databases in a variety of formats, both SQL and nonrelational databases, and across many platforms.

ICE supports both static and dynamic HTML report generation through its Report Writer. Dynamic reports utilize all of the conditional processing and multimedia available to its standard database reports. The output of existing Report Writer files can be wrapped in HTML with Report Writer.

ICE also provides an additional level of interactivity via Dynamic SQL that allows for the easy integration of commercial, online transactions or ordering systems into a Web site. ICE is closely tied with the Spyglass Web server or can use CGI on other Web servers to parse the HTML file or initiate the Report Writer for HTML output back to the browser.

CAI's second solution, Jasmine, specializes in handling multimedia content. The Jasmine Application Development Environment (JADE) will build rich multimedia applications that can be deployed on any web and viewed in a browser through the use of a plug-in. JADE is the database equivalent to Macromedia's Director, but it has more of a business and technical orientation. JavaScript or VBScript can be used to interact with the Jasmine application within a browser.

CAI's Unicenter (`http://www.cai.com/products/unicent/unicpd.htm`) enterprise-level management systems also have some powerhouse Web site management capabilities for those dealing with complex and fast-growing Web sites. With extensive monitoring, reporting, and administrative capabilities, you can watch all aspects of your web for problems. Event and security management tools enable you to define the severity of any problems, allowing you to apply the proper solution in the proper time frame. You can define all objects on a Web site as assets and control access to them by developers and reviewers, even on a scheduled basis. Other tools are available for performance and storage management. This represents the widest extension of database technology to cover all aspects of Web site development and maintenance.

Informix

Informix (`http://www.informix.com`) has allied itself with Netscape to provide an Internet solution from an application perspective rather than as a means of generating HTML pages. The Informix database is the foundation of 20 out of the 21 business applications that compose Netscape's AppFoundry. These applications in turn are based on Netscape ONE technologies, providing nonplatform-specific business applications. AppFoundry also includes several powerful enterprise development tools—Borland's IntraBuilder and NetObjects Fusion, among others—for building entire Web sites, Internet or intranet, based on Netscape ONE.

The Informix OnLine Workgroup Server database is also ODBC-compliant and can be used as a data source for IDC/HTX or ASP queries for publishing dynamic content.

Summary

As you've seen, a vast array of solutions are available for getting database information into HTML. Interestingly, most of the solutions are not locked into a single database on a single platform but usually cover a wide range of databases including relational, flat-file, and nonrelational types. This is probably indicative of the fact that any enterprise that has been integrating computers into its processes over the last 20 years or even longer has a menagerie of systems and databases and thus needs a wide-ranging solution if it wants to make much of its information available on the Internet.

Solutions start with the simple, static translation of database objects (forms, reports, tables, queries, and so on) to HTML pages with varying degrees of control over the output. In fact, the simplest HTML solution available for turning a database report into an HTML file is to use the `Print#` command found in most databases to hardcode the HTML tags into the report and output the report to disk. Of course, you have to learn HTML tags.

Dynamic solutions enable users to send variables to a database and receive information that they can read from a browser. This solution is the most powerful because the user can be on a completely different platform from the database and still have access to that information, which creates an instant cross-platform solution with minimal programming. Cross-platform solutions with tighter integration between the client browser and the database are possible by sending Java applets to the client machine. You can even use targeted database tools to manage an entire Web site.

With this range of solutions, any business, government agency, institution, or individual can make a database available on the Internet. Good security and flexibility is now built into many browser/database interactions, which means there is minimal threat to the integrity of the database being queried. Custom solutions using multiple programming languages can provide an even tighter link between the browser and the database for companies using the Internet as a wide-area network. There is no technological reason for not deploying a database on the World Wide Web. The simple HTML language and its associated browser has finally made universal database distribution and availability a reality after decades of struggle by monolithic companies with armies of programmers failed. Truly a David and Goliath epic.

IN THIS PART

- HTML Editors and Utilities 691
- Graphics Programs 709
- Advanced Web Authoring Tools 729
- Web Servers 759

VI PART

Development and Site Tools

CHAPTER 34

HTML Editors and Utilities

by Rick Darnell

IN THIS CHAPTER

- HTML Editors 692
- HTML Utilities 699
- Reading More About Your Choices 706

There are two ways to create and check HTML. The first is to get your old reliable text editor, type in the tags and content, and then view the results on a browser. If you need to check it for compliance with one of the HTML standards, you get your source code in one hand and the standards in the other.

Although this method certainly works, it's kind of like digging a lateral field for your septic tank with a shovel. It gets the job done, but you expend a lot more time and energy than is really required.

Luckily, we live in a world filled with people who think there are easier ways to do everything. Some of these people have turned their energies to creating editors designed especially for producing HTML and engaging in related activities such as updating pages and verifying syntax.

This chapter covers a handful of those tools. It is by no means a comprehensive list, but it covers the major players and popular products available to Web developers. I've included a range of editors for a variety of platforms, plus a few utilities to help with some of your other tasks. In this and the following chapters, you will get a feel for the many options available to fill your HTML and Web publishing needs.

A wide range of tools is covered here, including the old reliable text-based editors and some of the newest offerings in the WYSIWYG (What You See Is What You Get) category. Web editors haven't really kept up with the pace of the World Wide Web up to this point. But, as you can see with recent offerings from Netscape and Adobe, that situation is quickly changing.

HTML Editors

There are two basic kinds of editors—HTML source and WYSIWYG. HTML source editors are similar to customized text editors. These editors enable you to insert and edit various tags and content, but you're still working with a bunch of text and tags. You'll need some knowledge of HTML to work with a source editor. Even though many are well documented and provide intuitive support for many tags, a working knowledge of HTML will make the process much easier.

WYSIWYG is synonymous with graphical editors. When you add a horizontal rule, you see a horizontal rule on the page, not the `<HR>` tag. Knowledge of HTML is not as important with this kind of editor, although it never hurts. Depending on the sophistication of the editor, you may want to work directly with the source HTML code to tweak the page to correct idiosyncrasies, such as where tags are nested or to remove empty tags like `<H1></H1>`.

PageMill

Long the sole domain of the Apple Macintosh, Adobe PageMill is now also available for Windows 95. It is one of the most popular HTML editing tools on the Macintosh platform and is now making a run at the PC market.

TIP

PageMill is available from Adobe Corporation at `www.adobe.com`.

Those who use the Macintosh version will recognize their old friend in its new environment. (See Figure 34.1.) The power and functionality are the same, with the addition of several new features. Creating a Web page is as easy as typing and watching the results take shape through the WYSIWYG option. You can switch with a single keystroke to a source-edit mode with color-coded HTML tags. PageMill also provides options for importing word-processing files from several different formats, including Microsoft Word and WordPerfect.

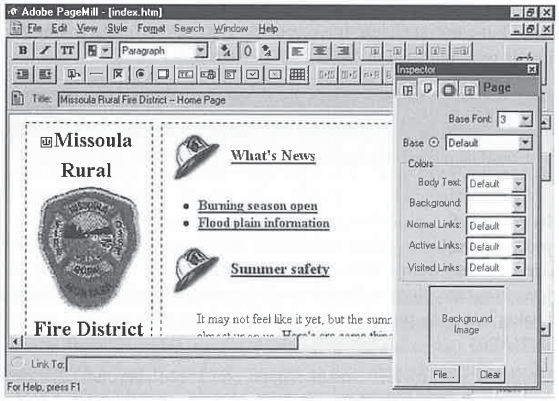


FIGURE 34.1.
PageMill, the popular Web page editing software developed for the Macintosh, is now available for PC users.

One distinctive editor option is the hidden HTML option, with which you can view items that are normally not displayed for the end user while you edit, such as comments, scripts, and anchors. As with many graphic HTML tools on the market, PageMill also supports the creation of image maps. You can add hyperlinks manually or by dragging them from a browser. Links are live within the editor, which means that you can check them while editing without switching to a browser.

Standard features include spell check, search and replace, and intuitive support for frames and tables. You can create frames and tables using a drag-and-drop interface for both content and specifications, and the results are immediately visible on your editing screen. PageMill also supports importing information from Excel files into HTML table format.

Adobe's program has a few quirks. First, it implements HTML headings a bit oddly. Instead of using <H1> through <H6>, PageMill uses Smallest through Largest. Ironically, the world's leading font supplier didn't include support for the <FONT face> attributes.

HomeSite

The story of HomeSite for Windows is one of those "only in America" stories about a guy and an idea. The guy was a cartoonist who was publishing his work on the World Wide Web but wasn't very happy with the tools he had to work with, beginning with a text editor and moving up to some of the more complex software suites. Nothing had the range of capabilities and features he was looking for, so he set out to create his own.

The result is an author's authoring program that is full featured, easy to use, and sells for a low shareware price. It is every bit the equal of other more sophisticated HTML source authoring tools, but costs much less. It is one of the grassroots favorites among Web authors and is commonly recommended in the HTML newsgroups.

TIP

HomeSite is available from Allaire Corporation at www.allaire.com.

Its user interface provides easy access to virtually every HTML element. (See Figure 34.2.) The process is intuitive—place your cursor where tags should appear, and then click a button or select an item from a menu. HomeSite supports the most popular content items, including Java, Netscape plug-ins, and ActiveX, in addition to JavaScript tools for developing scripts. Dialog boxes are provided for most of the tags with extended options, such as <FONT> and <APPLET>.

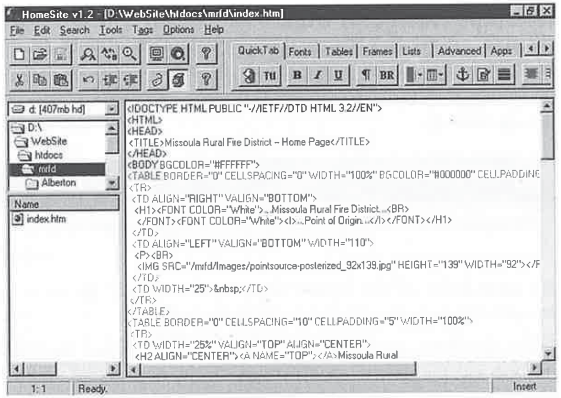
HomeSite also provides extended support for HTML beyond 3.2, including access to browser-specific tags such as <BGSOUND> for Internet Explorer and <EMBED> for Navigator. In addition, it provides a complete set of additional tools, such as search and replace, spell check, color-coded tags, and document weight. It also includes link validation and frame wizards for quicker construction and maintenance of HTML pages.

NOTE

A document's *weight* is the amount of time it requires to download from server to browser. Checking a document's weight is a good rule of thumb to see whether your pages are too loaded with graphics and other special content.

Internet Explorer was apparently the browser of choice for HomeSite's developer because HomeSite defaults to Internet Explorer to view completed pages.

FIGURE 34.2.
HomeSite is an authoring program created by a bona fide HTML author and is one of the grassroots favorites on the World Wide Web.



Some of HomeSite's drawbacks come in the site management area. It relies on FTP programs or Microsoft's Web Publishing wizard to move files from the local computer to the Web site. HTML validation is not supported, although you can still use third-party utilities such as HTML Validator.

BBEdit

BBEdit is the premier text editor for Macintosh programmers and HTML authors. What BBEdit lacks in WYSIWYG (which is currently nonexistent in this product), it makes up for in sheer editing power. BBEdit is fully AppleScriptable, has the capability to find and replace text within many documents at once (which is handy if a link changes or a common element must be updated), and is fully compatible with the Internet Config application to allow simple and seamless integration with your favorite Internet tools including FTP and Web browsers.

A number of BBEdit extensions (most of which you can easily download from the Bare Bones Web site) integrate features found in most other text-based HTML editors. The current champion of the bunch is arguably Lindsay Davies' HTML Tools extensions. This set of extensions allows fast and easy HTML markup using a customizable floating palette and supports most of the HTML tags and attributes used today. These extensions also allow for the addition of user-customized tags with limited scripting. These particular extensions, however, are available and compatible only with the BBEdit Pro version.

As you might have guessed, BBEdit comes in two versions. BBEdit Lite is 100% free and is highly recommended for any Macintosh. BBEdit Pro is not free, but it adds a plethora of features and ships with some of the greatest HTML extensions available. More information, as well as BBEdit Lite and a selection of BBEdit extensions, can be found at http://www.barebones.com/.

Netscape Navigator Gold

Netscape put together an interesting combination with its Gold version of Navigator, available for Windows, Macintosh, and a variety of UNIX platforms. Netscape combined one of the most popular browsers with an editor to provide integrated page creation and testing under one umbrella.

TIP

Navigator Gold is available directly from Netscape at `home.netscape.com`.

Navigator Gold includes support for many HTML 3.2 tags and is tightly integrated with Navigator to easily test and review the finished project.

Working with Navigator Gold is quite easy. It can work with any HTML page—as you surf the Web with the browser side of Navigator Gold, you can click the pencil icon on the toolbar to download the page and its graphics for modification. (See Figure 34.3.) As with any editor, you can also begin with a blank page or work from the various templates and wizards located on Netscape's Web site.

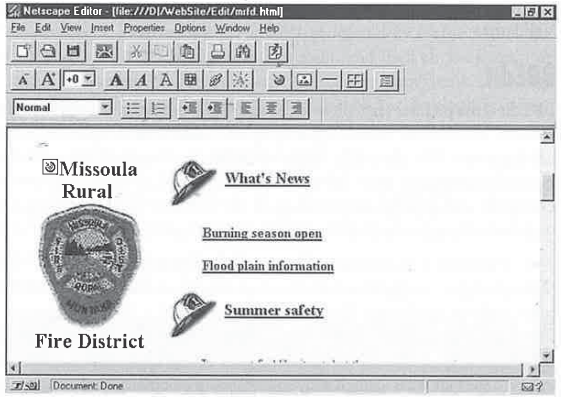


FIGURE 34.3.
Navigator Gold is an enhanced version of the standard Netscape Navigator software. It is a quick and easy way to create Web pages but lacks support for advanced HTML, such as frames and forms.

Netscape Gold includes support for editing HTML tables. The formatting commands are quite extensive and cover all of the current attributes HTML 3.2 supports. The WYSIWYG interface very accurately represents what will appear on the user's browser.

By entering a server name and password, Navigator Gold can also take care of the work of posting your pages to their ultimate destination. In addition to the actual HTML file, the editor also takes care of including all of the related accessories, such as image files.

The editor component of Gold seems to run a version behind the browser. Even though the browser includes support for frames, forms, or typefaces, the editor doesn't support any of these options. It does include an option to insert custom tags by hand, however.

The editor is also short in its list of amenities; it only includes spell check, image map creation, and drag-and-drop operation for items such as hypertext links.

Netscape Composer

Netscape Composer, a part of the Netscape Communicator (4.0) release, is the newest HTML editing program. It is available for all major platforms, including Windows, Macintosh, and UNIX. Composer includes a simpler and more intuitive interface than the Netscape Gold editor, although it includes capabilities beyond its predecessor. Like its counterpart, Composer is geared toward creating individual pages and includes no support for any site management activities. It does provide remote publishing features through FTP and HTTP protocols.

TIP

Composer is available directly from Netscape at `home.netscape.com`.

Composer's editing screen is similar to the Navigator browser window, with additional toolbars for HTML features such as links, tables, and character and paragraph formatting. (See Figure 34.4.) Although Composer supports Java and JavaScript, it doesn't support frames or forms.

You don't need any knowledge of HTML to work with Composer's dialog boxes. For instance, when formatting a graphic, the user is presented with a set of thumbnail diagrams that illustrate how each option affects the relation of the image to the text. You use a similar method for illustrating the result of editing actions as when you work with tables and paragraphs.

Composer is a WYSIWYG editor. If you need to work directly on the HTML source code, use a menu option to start an external text editor. After completing any changes, Composer gives you the option of updating the view in the editing area.

Like its cousin, Navigator Gold, Composer offers tight integration with Netscape browsers, which enables authors to drag and drop links from remote pages directly into an HTML page in progress. You can also drag image files from the browser into the editor or send HTML documents as e-mail or newsgroup messages.

No templates or wizards are included with Composer to make creating Web pages easier, though a resource is located on Netscape's site to make this possible. To do so, download sample pages to edit. Your other option is to use a CGI-driven wizard on Netscape's site to create a page

with the desired headlines, text, images, and bullets that you can then download to your computer to edit.

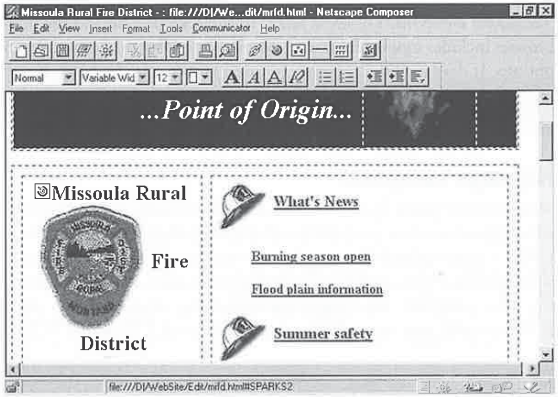


FIGURE 34.4.
Netscape Composer will eventually replace Navigator Gold. Its functionality is very similar, though the user interface is slightly different.

Without forms, frames, or site management capability, Composer is not effective for complex pages or for working with entire Web sites, but it is a good tool for quickly building and publishing basic HTML pages.

HotDog

Once upon a time, HotDog was relegated to the world of "Oh, isn't that cute." It was a basic little HTML editor that depended more on an entertaining canine theme than on actual function and utility. But that's all changed. HotDog includes support for the full range of HTML content, including Java, plug-ins, and style sheets. It runs on Windows 95 and NT.

TIP

HotDog is available from Sausage Software at `www.sausage.com`.

One of HotDog's strongest features is its customizable interface. (See Figure 34.5.) You can add or remove virtually anything from its toolbar, which makes it especially useful for people with special needs or patterns to their HTML page construction.

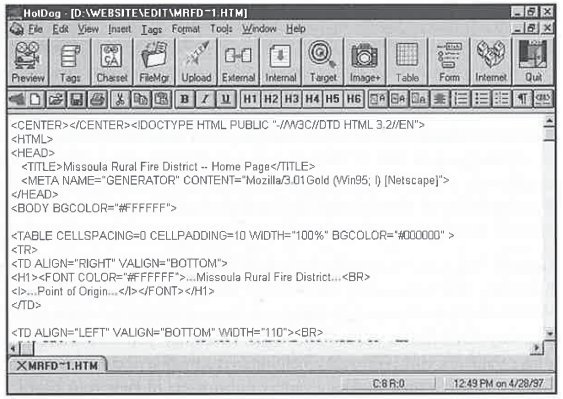


FIGURE 34.5.
HotDog, one of the many available shareware HTML editors, has grown into one of the strongest offerings among its compatriots.

The standard edition links to a browser (usually Internet Explorer) to view what a page will look like when it is completed. The professional version includes Rover, a browser interface that behaves very similarly to Internet Explorer.

An important addition to HotDog's capabilities is its support for style sheets, with which you can implement font or page characteristics for an entire page without working through the entire document. There is also a Resource Manager that integrates all page components for link checking, file management, and graphics preview.

HotDog includes FTP capability to easily transfer files when you're finished editing. This capability also includes the ability to work on files from remote locations, spell check, HTML verification, page wizards, and table and frame tools.

HTML Utilities

The following set of programs doesn't actually create original HTML pages. The utilities make it easier for you to manage your site and the pages within it. To that extent, you might find this selection of programs useful in your day-to-day HTML production.

CSE 3310 HTML Validator

HTML Validator for Windows helps you check HTML documents for correct syntax according to HTML 3.2 and 2.0. Direct it to look at an HTML page, and it produces a list of messages based on what it finds, including whether it finds nothing wrong. It also supports Netscape and Microsoft extensions, current with each company's 3.0 release, plus tags for tables and frames. It includes an option to add your own tags and attributes, so you can update it to any HTML 4.0 features you want to use.

TIP

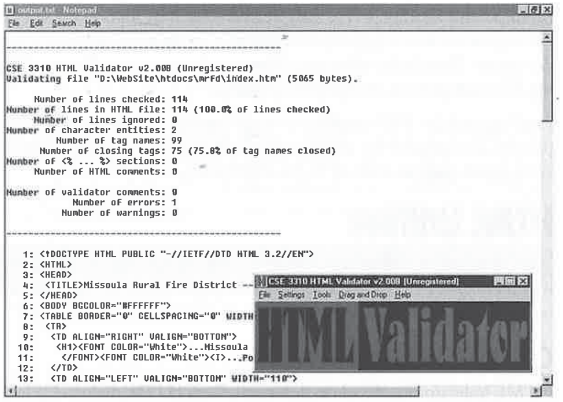
HTML Validator is available from the AISoft Internet Solutions at `www.htmlvalidator.com`.

You cannot check your document for syntax in a standard HTML browser because the browser is only designed to view HTML documents. If syntax errors exist in a document, the browser usually only attempts to guess how the document should appear, which results in documents that display in a variety of ways, depending on the browser. If a document has enough errors, it might not display at all.

Some HTML editors currently include HTML validation, including FrontPage and Backstage. But many don't, including the editors I discussed earlier in this chapter. Before publishing HTML documents, especially those you create manually or with a dumb HTML editor (*dumb* meaning that it has no validation or other checking), you should check for syntax errors. That's where Validator comes into play, helping to ensure that documents are written in correct HTML syntax, which in turn enables a variety of browsers to view your pages.

Use the syntax error list that Validator generates to correct your document before publishing it. (See Figure 34.6.) The list is stored in a text file for printing and use, complete with validation process information and line-by-line document listings.

FIGURE 34.6.
The result of validation is a text file that includes Web page statistics and a line-by-line listing with error messages.



HTML Validator also includes tools that change HTML tags and attributes to uppercase or lowercase, convert different operating system text file formats to other text file formats, and enable you to use templates in your documents.

TIP

Using a validation program doesn't guarantee that a document will be perfect in its syntax. Although Validator finds the vast majority of errors, some will fall through the cracks, especially in HTML code, which is sloppy to begin with. Programs such as Validator will find most syntax errors, especially if you use it after each revision of a document.

Validator finds a majority of HTML syntax errors, including common mistakes such as missing double quotation marks and closing tags, mismatched < and > characters, and tags in incorrect locations. When it encounters an unknown attribute, Validator generates an error message or ignores what it can't understand, and gives the offending item the benefit of the doubt.

HTML 3.2 does not require `<HEAD>` and `<BODY>` tags, but Validator requires them for successful validation. This should not be a problem because using them benefits page structure and construction. You might need to close some tags for proper validation, such as table data (`<TD>`), even though it's not required as part of HTML 3.2. Including them will not adversely affect table display, though having to go back to insert them could be a major inconvenience.

WHAT ABOUT THE NAME CSE 3310 HTML VALIDATOR?

There are lots of strange names for computer software, including HotDog, Fusion, Packrat, xRes, and so on. For this application, CSE 3310 HTML Validator comes from the University of Texas at Arlington college course for which it was created—Computer Science Engineering 3310 (software engineering).

HTML Validator is shareware. After validating 150 HTML documents, it disables itself and requires that you pay a registration fee.

InContext WebAnalyzer

InContext WebAnalyzer—which is available for Windows 3.1, 95, and NT—clearly identifies any broken links on your site so that you can resolve them and spare your users from the agony of `Error 404: Not Found`. Its straightforward operation results in more information about a Web site and its individual pages than you could ever use.

TIP

InContext WebAnalyzer is available for demonstration or purchase at `www.incontext.com`.

The bright side of gathering all of the information is that it doesn't take very long, depending on the speed of your modem and the load on the Web server. When WebAnalyzer is finished with its detective work, you have two graphical displays of the links and resources to which your site refers, plus a text listing of all the files. (See Figure 34.7.)

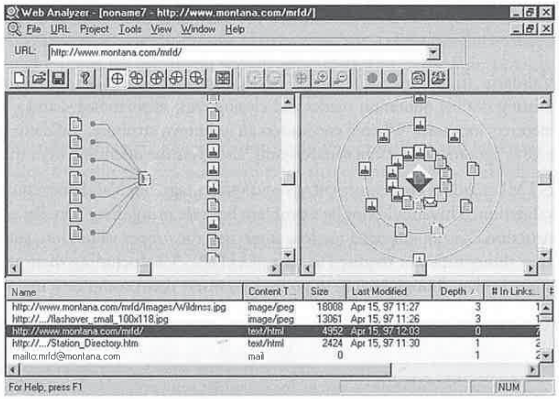


FIGURE 34.7.
InContext WebAnalyzer's interface and functionality is similar to Microsoft FrontPage Explorer's, but InContext WebAnalyzer is quicker and offers more flexibility.

The great strength of WebAnalyzer's intuitive graphical display is not its only asset. The display is interactive. Clicking a node in the WaveFront or file list makes it the center of attention in the individual node view. A special icon represents each link and element external to the page, including hyperlinks, e-mail addresses, images, and other objects.

A text icon with a circle and slash represents unresolved or broken links. Place the pointer over any of the icons to show its URL so you know where to look for the problem. When you specify the problem area, you can launch an HTML editor of your choice to edit the offending link and correct the problem.

Using a browser you specify, WebAnalyzer also presents a comprehensive set of reports in HTML format. (See Figure 34.8.) These reports are a verbose presentation of the information from the graphical views. They include summary reports of indexes and sites plus page-by-page summaries of the links and resources used.

Setting options on WebAnalyzer is especially important. After you give a starting URL, the program begins to crawl outward, following each link and resource. It gradually mushrooms as each page goes in more and more directions. A realistic limit to this mushrooming is two to three levels out for Internet-based sites and four to five levels for intranets. Beyond these levels, too much information to be useful in resolving broken links is returned.

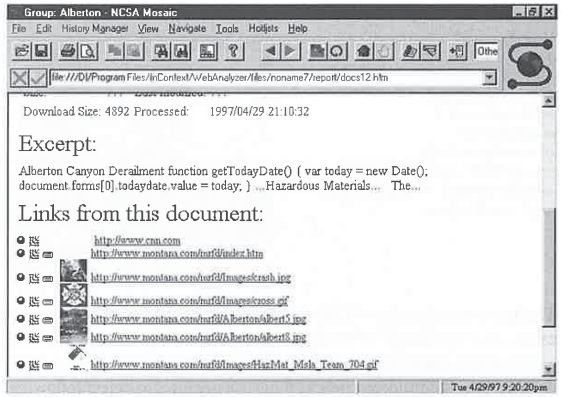


FIGURE 34.8.
You can view all reports generated by WebAnalyzer through a standard browser. These reports include extended information about a page or site, including thumbnail images of graphic files.

Xpire Plus

Often, information on a Web page is only valid for a limited time. To help point out new material, many people place NEW! tags in front of hyperlinks. But as time goes by and the information isn't "new" anymore, you have to constantly search through all of your HTML files to look for outdated information. If you have several pages, this search can be very time-consuming, boring, and error-prone.

TIP

Find Xpire Plus on the World Wide Web at `www.kagi.com/bungalow`.

The Xpire Plus HTML utility, which runs only on Windows 95, assists in this facet of Web site maintenance by helping you keep pages up to date and preventing expired links. Instead of manually searching through each individual HTML file for outdated information, Xpire prompts you for a directory and then takes over checking its entire contents for old information. (See Figure 34.9.) The utility also supports the creation of sections that appear after a certain date.

To do this, encase the time-sensitive items with a pair of comment tags. For example, place `<!--XPIRE10/31/97 -->` at the beginning of a section that should end on Halloween and immediately follow it with `<!--ENDX-->`. You would handle sections that should appear on a certain date similarly, such as `<!--XON 10/31/97 _Today is Halloween_ XON-->`.

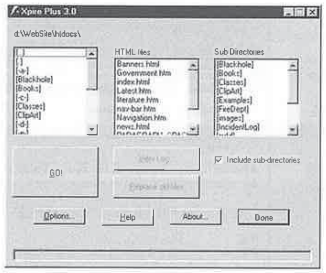


FIGURE 34.9.
Using Xpire with specially coded HTML pages can automate the process of making sure content appears and disappears at its appointed times. It also removes NEW! flags when the content is no longer considered new.

The last feature handles NEW! tags or icons. Xpire uses another comment tag, `<!--XNEW 10/28/97-->`, immediately before the program preferences, and operates it in conjunction with program preferences to remove each item after a predetermined time. You can set the number of days an item is considered new and how many lines to delete following the `<XNEW>` tag.

After you have marked the HTML documents, just run Xpire periodically to make sure each section is added or removed as necessary.

The downside to Xpire is that you have to get used to including the tags with your pages as you're editing them. If your HTML editor automatically generates comment tags, this might not be too big of a problem. Some editors, however, such as FrontPage Editor, include other information within the comment tag to help you with formatting while editing.

Bandwidth Buster

Bandwidth Buster, available for Windows 3.1 and 95, modifies HTML files with tags and features to reduce the download time users need to view your pages. The whole concept is to take advantage of all the little quirks and browser features to optimize file size and efficiency as an HTML file is retrieved from the server.

TIP

Bandwidth Buster is available for download from Sausage Software at `www.sausage.com`.

Bandwidth Buster uses several methods to trim your site and reduce your page's download and layout time. First, it can replace GIF images with compressed JPEG images using a compression ratio that you select. This occurs through a batch conversion process so you don't have to specify each file. It also adds HTML image size and alternate content tags to speed up your page on most browsers, and, optionally, it can create low-resolution copies of your images to insert as `LOW SRC` images in your HTML files. The `LOW SRC` images are very small and loaded first by the browser. After the browser retrieves all other content, it loads the full image.

Lastly, Bandwidth Buster can create a text-only alternative Web site. It removes all references to any images and saves all files with their directory structures to a separate location.

You can access all options from a simple interface. (See Figure 34.10.) After the conversion has taken place, Bandwidth Buster offers an additional feature to measure the download time of specific pages based on preset modem speeds.

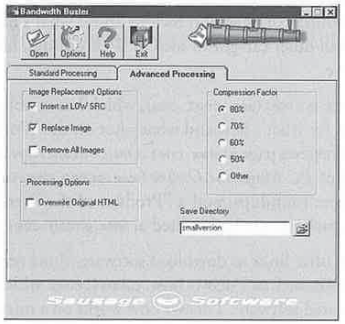


FIGURE 34.10.
Bandwidth Buster, developed by Sausage Software (makers of HotDog), optimizes Web pages for download and display time on most browsers.

Consistent with Sausage's usual style, several entertaining little animations and sound effects are tossed into the bundle for various activities. For example, the start-up flash screen includes a dog grabbing the "Internet cable," complete with the sound of electrical zapping. Though these features aren't entirely necessary, they use minimal system resources and are only slightly annoying in terms of time.

Crunch!

Crunch! is a command-line utility that scans HTML files for unnecessary information. Because it's a command-line utility, you can get Crunch! for Windows 3.1, 95, NT, and even OS/2. If you use a Web authoring system such as Microsoft FrontPage or Netscape Gold or even if you write your own Web pages, you're likely to end up with files that contain many unnecessary characters and tags, such as comments.

TIP

Crunch! is available on the Web from Tennyson Maxwell at `www.tenmax.com`.

Crunch! strips these unnecessary characters and tags from your Web pages, reducing their size by up to 50 percent. It also helps you spot HTML coding errors. After crunching, links with missing quotation marks or incorrectly terminated HTML tags show up as errors when you view them with a browser or pass them through a validator.

After crunching, your HTML files are virtually unreadable, so you must back up your files if you plan to edit them later. The crunched format places all tags and content on one continuous line, making it virtually unreadable in source view, which the Crunch! developers tout as a great way to stop data pirates.

Reading More About Your Choices

Several good sources of news and reviews about what's happening in the HTML editor market—along with other categories such as Web utilities, image editors, and related items—are available online.

The first source is *clnet* (`www.cnet.com`), which has established itself as a leading online information source for what's new and what's hot on the World Wide Web. You can hook up directly to their reviews page at `www.cnet.com/Content/Reviews`. Another source is the "Internet User" section of *PC Magazine Online* (`www.pcmag.com/iu`). It includes lists of new products, reviews, product roundups, and a "Product of the Week" feature. HTML editors and site management packages are highlighted at `www.pcmag.com/iu/author/htmledit`.

Other services offer links to download software, short reviews, and individual rating systems, including services such as Tucows (`www.tucows.com`), which serves as a collection point for a lot of Internet-related software. Tucows now works on a mirror system, in which other Web sites such as our beloved Sams.net (`www.mcp.com/sams`) copy its files and structure.

Summary

HTML editors come in many different sizes and with many different strengths. Each is different in the way it supports creating and editing Web pages, beginning with basic appearance (HTML source or WYSIWYG), type and format of toolbars, and menu commands. The differences grow as extended features are added, such as publishing to remote sites or spell-check capability.

Which editor you choose depends on your needs and your HTML comfort level. If you're new to page editing, you'll probably want to start with a simpler editor, such as Netscape Gold, Composer, or HomeSite. More advanced users will look to options such as PageMill or HotDog. Whichever you choose, make sure the editor extends and simplifies your capability to create Web pages, and doesn't limit your options with a difficult interface or inadequate support for the tags you use.

After you create your pages and Web site, you'll want to look at some of the many utilities designed to keep them up to date and usable. From HTML validators to link checkers to page optimization, don't overlook some of the solutions to common problems. Many utilities are free or very inexpensive via shareware.

Like any good book about emerging technologies, this book will contain several outdated items by the time it hits the bookshelves and your hands. So, even though I've reviewed current software in this section, it's important that you keep in mind that new tools and new revisions of existing tools may have already appeared by the time the ink on this page dries.

For the latest and greatest news about capabilities and features, you need to head straight for the horse's mouth. Of course, even though this chapter supplies URLs so you can get in touch with the respective developers, remember that the developers will tell you that their software is the greatest thing since sliced bread. For that reason, it's good to get a second or third opinion.

CHAPTER 35

Graphics Programs

by Michael A. Larson

IN THIS CHAPTER

- Choosing a Graphics Package *711*
- Bitmap Packages *712*
- Vector Packages *719*
- An Essential Graphics Utility for the Web Author: Debabelizer *722*
- Animation and Video Packages *723*
- Ray Tracing, 3D Rendering, and Fractals *726*

A large part of the appeal of HTML is its capability to place both text and images on the same page. Unless you are transcribing old political speeches to the WWW, eventually you will need to make or modify an image for use in your Web page. Images can add an extra point of interest, summarize a large amount of data, or emphasize some point in your text.

Using graphics does not necessarily mean you have to draw them yourself. Large collections of bars, buttons, and backgrounds are available on the WWW (as discussed in Appendix D, "HTML Resources") for noncommercial use or available for purchase. To a lesser degree you can find collections of images, clip art, and animated GIFs on the Internet that can also be freely used.

But you will almost certainly come to a moment when you need a particular graphic of a particular size, color, or certain "look," and it just isn't out there. If you have the budget, you can hire an artist or get the advertising department to create the graphic you need. A less expensive route might be to purchase an image from a stock photo catalog or buy a clip art collection. Or you might have the image already available, perhaps as a large scanned image. With the aid of the many graphics programs I discuss here, you might be able to draw it yourself, even if you are artistically challenged. Remember that even though these programs are good, you shouldn't expect the next Mona Lisa to flow from your fingertips.

No matter where your image comes from, chances are good it will not be in a form that is ready for use in your Web page. Placing a 10MB scanned image in your Web page will not improve the popularity of your Web site. Minimally, you will need to reduce the size of your image and probably put it into the correct file format. Moreover, you will probably need to put some creative touches on your image by cropping, rotating, changing colors, applying filters, combining it with other images, adjusting the palette or number of colors, adding shadows or transparency, and so on. You need to learn one or more graphics packages to accomplish all this.

The majority of graphics packages, both vector and bitmap, are available for the Windows and Macintosh platforms. There are far fewer native UNIX and OS/2 graphics packages. In this chapter, I review the more popular graphics programs available for all four common computing platforms, discussing their features, ease-of-use, strengths and weaknesses, and the unique capabilities that make a given package stand out from the crowd, especially as it relates to the needs of the Web author. Moving pictures can also be inserted into Web pages, so this chapter includes information about packages that you can use to make animated GIFs or to add streaming video to a page.

I don't discuss the following:

- Fonts, except as they relate to 3D rendering.
- High-end graphics packages such as Pixar or most multimedia authoring tools. Though these packages are extremely interesting and capable, using them for Web images is usually overkill. These packages may become more interesting to the Web author as their prices drop and the bandwidth problems on the Internet ease.

- Making AVI or Quick Time movies. These require special hardware and the current bandwidth problems on the Internet generally make the use of these files in their native format undesirable.

Choosing a Graphics Package

Graphics packages come in many different types on many different platforms. Choosing a package will depend on the following factors:

- Your budget
- The complexity and types of image modifications you want to perform
- How much time you have to master the graphics packages
- The capabilities of your hardware

Graphics packages covering a wide range of sophistication are available for all common computing platforms. Packages that used to be available on the Macintosh have been ported only to Windows and, in many instances, to UNIX. It is unlikely that you will need to purchase a special computer platform exclusively for graphics, although you may need to upgrade existing hardware on your HTML authoring machine.

If you intend to do extensive graphics work, two hardware items that you should give serious consideration to are your monitor and video card. Try not to work with any monitor smaller than 17 inches (diagonal) and get the highest vertical scan rate you can afford in order to reduce eye strain. Make sure that your video card has sufficient RAM to support 24-bit color at whatever screen resolution you generally work at. It is also handy to be able to change color depth and/or resolution on the fly (without rebooting your system) to check how your Web pages look under 256 colors at 640×480.

Graphics packages generally fall into one of two types: vector and bitmap.

A Brief Overview of Vector and Bitmap Editors

Vector programs, often called *drawing* or *illustration* programs, use equations to define the placement of lines, colors, and objects. As such, their native file sizes are very compact and the images scale well; that is, there is no loss in real resolution (apparent resolution might suffer) no matter how large or small you make the image. Vector packages are commonly used by professional illustrators to create images from scratch. Many packages also come with extensive, ready-to-use clip art collections. CorelDraw and Adobe Illustrator are vector packages. From the perspective of putting graphics on a Web page, the strengths of vector programs include

- Nondestructive image scaling
- Fine control over shape
- Many ready-made objects that can be easily disassembled, moved, and distorted
- Responsiveness (except for very complex images or operations)

- Lower hardware requirements than bitmap editors
- Ease of exporting the final image to a file format usable on a Web page

The major disadvantages are

- Native vector file formats are not usable on the Web without browser plug-ins.
- Many packages have a steep learning curve.
- Colors might shift during export, requiring touch up in a bitmap editor.

Bitmap editors work with bitmap images, which also are called *raster* images. A bitmap is a pixel-by-pixel definition of a picture. The total number of pixels (image size) is fixed in any given file, as are the color and placement of each pixel in the image. Image size is dependent upon the number of colors and pixels, although some bitmap file formats use compression algorithms to reduce their size. Bitmap editors work by manipulating each pixel in an image. Adobe Photoshop and PaintShop Pro are bitmap editors. Paint programs, such as Fractal Painter, are also bitmap editors. Some of the strengths of bitmap programs are as follows:

- You can work directly with GIF or JPEG images.
- A wide variety of special effects is available, including drop shadows and transparent GIFs.

Some of the disadvantages of bitmap editors include the following:

- They can be very CPU-, disk-, and RAM-intensive, requiring more hardware than vector packages.
- Many packages have a steep learning curve.
- Image information is lost due to pixel interpolation when an image is resized.

These package definitions, vector and bitmap, are not strict: You can import bitmaps into most vector packages and draw vector curves in some bitmap packages. There are also related types of graphics packages that have special capabilities such as ray-tracing, 3D rendering, or fractal generation programs. If you will be generating or manipulating large numbers of graphics for Web use, graphics utilities such as Debabelizer are also essential.

Bitmap Packages

The vast majority of graphics on the WWW are in GIF or JPG bitmap formats, so bitmap packages are a Web author's first tools of choice. Bitmap packages generally fall into two categories: editing programs and paint programs. Editing programs, typified by Adobe Photoshop, have an extensive array of filters and functions available for modifying existing images. They have fewer tools for drawing or creating images. On the other hand, paint programs, such as Fractal Painter, have an extensive array of "media" (chalk, charcoal, multiple paintbrushes, pencils, papers, and so on) available for drawing, creating, or modifying images. Base your decision of whether to go with an editing or paint program on whether you will be spending more time editing existing images or creating new ones. You might even need both types in your graphics toolbox to give you the most options.

Most bitmap editors have many tools and capabilities in common, including these:

- Selection tools, including a marquee selection box, lasso, magic wand, and selection by color range.
- Basic drawing tools, including a pencil and paintbrush at a minimum.
- Filters, which can range from touch-up filters for sharpening or blurring an image through special effects such as creating drop shadows or applying light sources.
- Support for scanners.
- Support for common bitmap file formats and flavors including BMP, TIF, RLE, GIF, JPG, PCX, PCD, PSD, PNG, and TGA.
- The capability to open images from many different bitmap file formats and save them to many different bitmap formats.
- The capability to flip, mirror, and rotate (in degree increments) images.
- Cloning, which uses one part of an image as the "paint" for another part or another image.
- Retouching, which includes changing colors, color replacement, brightness, resizing, cropping, contrast, hue, or saturation among others.
- Masks for protecting one part of an image while another part is changed.

Many more packages out there are similar in functionality to the ones discussed in the following sections. It is simply not possible to be exhaustive on this topic. I chose the ones I did because of their popularity and features. Even if you find none of the packages featured here to be suitable for your uses, you will at least know what to look for in other bitmap editing or paint programs.

Adobe Photoshop

The latest incarnation of Adobe Photoshop (`http://www.adobe.com/prodindex/photoshop/main.html`) for Macintosh and Windows (95 and NT) is Version 4.0. Photoshop is the most widely used bitmap editor among graphics professionals. Its popularity is due to its combination of incredible power and a well laid out, elegant interface. The program is also highly extensible with a wide array of filters and add-ons available. Adobe Photoshop comes on a CD-ROM with an excellent printed user guide and a second tutorial CD-ROM with beginning-to-advanced lessons.

In addition to the usual bitmap editing features already mentioned, Photoshop has

- Layers and ways to merge them or change transparency.
- Adjustment layers for adjusting color, hue, and so on without modifying the underlying image. Great for experimentation.

- Automation. You can record a set of actions on one image and apply the actions to another group of images.
- Grids for precisely placing and aligning the parts of your image.
- A very intuitive and wide-ranging zooming tool.
- The capability to save selections in the file for later recall.
- A path drawing tool and the capability to export the path to Adobe Illustrator.
- An easy way to create shadows for any text or object and make any color or the background transparent for a transparent GIF.
- Gradient fills with more than two colors and the capability to adjust transparency anywhere along the gradient.
- Thirteen categories of filters, each with multiple capabilities—for example, artistic filters (neon glow), rendering filters (lens flare), and texture creating filters (stained glass)—for a total of more than 90 filters.

More filters are available from a variety of vendors, including the famous Kai's Power Tools. Adobe defined the standard format for filter plug-ins; most plug-ins are written to this standard first and most other bitmap packages support this standard.

As you can see, Photoshop is feature-laden, and this is its greatest strength. This wide-ranging capability provides the ultimate in flexibility for artistic design. You can do almost anything with a bitmap under Photoshop.

This also leads to its two primary weaknesses:

1. It is a resource hog. It is not comfortable with less than 32MB of total RAM on any platform. And, of course, 24-bit color bitmap files can eat gigabyte hard drives for lunch. Photoshop needs a muscle car to run; otherwise, don't even try it. On the positive side, it can handle more than one CPU under Windows NT.
2. It has a significant learning curve. Not only will you need to learn what the features are and how to use them, but you must also learn when it is appropriate to use them and what the artistic effects of their use will be. This is a significant time commitment unless you are already a creative artist.

This program is not for the casual user. The time investment to learn and use it will only pay off if you spend a great deal of time creating or modifying images or need to have a large variety of modification options available for any given image. On the other hand, huge amounts of information are available about using Photoshop, including books, mailing lists, newsgroups, and many Web sites that are repositories of Photoshop tips and tricks. Here are some URLs on the Internet that are dedicated to Photoshop:

- PC Resources for Photoshop specializes in filters and Photoshop plug-ins. It is found at `http://www.netins.net/showcase/wolf359/adobepc.htm`.
- You can find the Ultimate Compendium of Photoshop Sites (yes, that is the name of the Web site) at `http://www.sas.upenn.edu/~pitharat/photoshop/`.

- Stop by Andy's Photoshop tips at `http://www.andyart.com/photoshop/index.htm`.
- My favorite Usenet newsgroup dedicated to Photoshop is `comp.graphics.apps.photoshop`.
- A huge Photoshop site, Photoshop Paradise, is at `http://desktoppublishing.com/photoshop.html`.
- If you want to join a Photoshop mailing list, go to `http://www.csua.berkeley.edu:8000/~kima/photoshop/`.
- The definitive online reference guide to Photoshop, Web Reference, is at `http://www.duke.edu/~ac10/photoshop/index.html`.

Note that many of the Web sites in this list also contain extensive lists of Photoshop sites on the WWW. The preceding list is just a sampler to get you started.

The minimum system requirements for using Photoshop on a Macintosh are the 68030 processor or a Power Macintosh with 16MB of RAM available for Photoshop and System 7.1 or later. For Windows 95 or Windows NT, the minimum is a 486 processor (a Pentium or Pentium Pro is highly recommended) and 16MB RAM. For both platforms, increasing the RAM to a minimum of 32MB is recommended. If you can afford more RAM, get it.

PaintShop Pro

This very affordable shareware package from JASC, Inc. (`http://www.jasc.com/psp.html`) has a wide following among casual-to-intermediate users and is quite popular with people doing typical Web graphics. This package combines ease of use with many of the sophisticated capabilities found in Photoshop. Version 4.12 also reads a wide variety of file formats, including several vector formats: CorelDraw (CDR and CMX), Micrographx Designer (DRW), AutoCAD (DXF), and WordPerfect Graphics (WPG). It also reads Photoshop (PSD) native files. You can easily convert batches of files between different graphics formats with this program. Some other features useful for Web authoring include

- Screen capture capability, full screen or only a selected portion
- Several different drawing media, including crayon and charcoal
- Drop shadows and the capability to create transparent GIFs
- A thumbnail image browser
- Support for Photoshop filters
- The easy creation of image masks

The main strengths of this program are that it is easy to use and learn without leaving out many important high-end features—and it is affordable to boot. Its main weakness is that is has fewer features than Photoshop, including far fewer default filters. Most sorely missed are layers and the step automation for applying the same modifications to a group of images.

The minimum system requirements for this Windows 95 or NT only (a Windows 3.1 version is also available) program again include a 486 with 8MB RAM, but I recommend a Pentium or better and 16MB RAM.

Fractal Painter

If you like to draw, pen, doodle, sketch, do origami, woodcarve, make snow angels, or have any shred of artistic inclination, you are going to love this program. Fractal Painter (`http://www.fractal.com`), currently in Version 4.0 on both Macintosh and Windows platforms, has the widest array of drawing media available in a mainstream computer program. You have over 150 brushes available, and you can modify any one of them. There are six different pencils, chalk, watercolor, pens for calligraphy, felt pens, oil paint, cloners, and airbrushes. You can even paint with more than half a dozen liquids. To complement this, you have a wide array of surfaces to paint on, including many different types of papers, weaves, and colors. Some of Fractal Painter's features are

- Layers (called floaters) and masks.
- An image can become a painting element (random or ordered) with the Image Hose brush. Each image is a separate Nozzle.
- You can create image maps inside the program.
- Capability to read Photoshop 3 files, preserving the layers.
- Support for Photoshop filters.
- Support for collaborative painting, which means several artists can work on the same canvas at the same time.
- Many of the media features can be applied on a frame-by-frame basis to QuickTime or Video for Windows files.
- Capability to import and perform vector editing on files from CorelDraw, Adobe Illustrator, or Macromedia FreeHand.

This vector file import and vector editing is unique to Fractal Painter. Other bitmap editors can import vector files but convert them to a raster format.

Fractal Painter definitely shines as an artistic paint program while supporting most of the basic bitmap editing functions. Its support of multiple media (both paint and canvas types) is unparalleled. Anyone familiar with traditional painting or drawing techniques and tools will find this program very intuitive to use; others will find it moderately difficult to learn. If you prefer to draw more of the graphics for your Web pages, you need look no further. If you can master much of the media, you also will have many more options available for editing existing images than are provided by filters.

For the Macintosh, you'll need a 68030/40 processor or Power Macintosh with 8MB RAM for the application and System 7.0 or later. For Windows (3.1, 95, or NT), you'll need a 486 or Pentium with 8MB RAM (16MB recommended). A Math coprocessor (FPU) is required on both platforms to support some operations.

General Image Manipulation Program (GIMP) for UNIX

If you're doing most of your HTML writing on a UNIX box and are tired of running to a Windows machine or a Mac to do your graphics, there is finally a good, general-purpose bitmap editor for you. This program currently exists as freeware (Version 0.54, soon to go commercial at Version 1.0, `http://www.xcf.berkeley.edu/~gimp/`) with compiled versions and source code available for Linux, Free BSD, HPUX, and Solaris.

This program makes good use of the unique and robust capabilities of the UNIX operating system. You can have multiple views of the same image on the screen at the same time and multiple undo levels, limited only by available RAM memory. It also supports most of the same, common capabilities as other bitmap editors, including masking and layers. Extensions have been written into the program so that custom plug-ins can be added. The freeware comes with 40 plug-ins. File support is available for GIF, JPEG, PNG, TIF, and XPM formats.

If you like to or have to work on a UNIX box, this program will be of great interest to you for getting your basic Web graphics work done. If your UNIX box is also your site's Web server, you are now in a one-stop shopping situation where you can write HTML and compose graphics on the same machine on which you're hosting your site.

GIMP requires support for shared memory from the operating system. It also requires X11 (R5 or R6) and Motif 1.2 or above.

ColorWorks for OS/2

SPG (`http://www.spg-net.com/products.html`) has put together a powerful bitmap editor in ColorWorks, Version 2. Rather than defining a standard set of tools like most other editors, this program allows you to "paint" with any effect or combination of effects, essentially giving you a limitless toolbox. It is also possible to save these combinations as a Graphics State file. SPG provides a number of predefined files for immediate use. ColorWorks is also the only bitmap editor I've seen so far that truly supports multithreading; that is, you can initiate multiple editing effects and each will spin off on its own thread. You do not have to wait for one thread to finish before starting another. You can also manage your threads much as you would manage print jobs through a print spooler, deleting or pausing them at will. Because OS/2 supports multiple CPUs, the multiple threads from ColorWorks will efficiently use all of the CPU power you have available. ColorWorks also uses extensive memory-managing techniques to maximize the RAM you have and the program itself uses only 1MB of RAM.

ColorWorks supports masking and multiple undo, although it lacks the capability to handle multiple layers. This graphics package is a good selection for the Web author working under OS/2 and will allow you to meet most common graphics creation and editing needs. Any PC that will run OS/2 Version 2.1 or higher will run ColorWorks.

Other Bitmap Editors

Several other bitmap editors are of note because they are written almost completely for Web use (Microsoft Image Composer, free at `http://www.microsoft.com/imagecomposer`) or because they are bundled with major vector graphics editors that are discussed next in this chapter. There are many more editing packages available, each with its own combination of features, price, and platform availability. Whichever bitmap editor you choose, make sure it is one you enjoy. The process of creating and editing graphics should give you a sense of accomplishment and not be an exercise in frustration from software that is ill-suited to your work style.

Microsoft Image Composer

This rewrite of the Altamira Composer editing package by Microsoft for Windows 95 and NT specifically emphasizes the needs of the Web author. This package is bundled and integrated with Microsoft FrontPage 97 and Visual InterDev. With Image Composer, any image brought into the program is converted to a sprite (not to be confused with sprites used in game programming). A *sprite* makes use of the alpha channel to basically define an area of transparency around the main objects, allowing for nonrectangular images. For instance, a picture of a potted plant will have the shape of the pot and the leaves; the background is not a part of the sprite. This sprite or a group of sprites can then be used to create a composition that can be saved in the native MIC file format or as a transparent GIF for use on a Web page. A sprite is roughly equivalent to a layer.

Image Composer contains a number of built-in filters and supports Photoshop plug-ins, the use of masks, and the easy creation of drop shadows. You cannot build animated GIFs, although Microsoft gives away a free GIF animation builder at its Internet site that works closely with this program. Because you can move individual sprites or groups of sprites in Image Composer, you can easily save a series of frames that can then be built into a GIF animation. Again, hardware requirements are a bit steep, with 32MB RAM recommended and up to 300MB of disk space needed if you install all of the images that come with the program.

Macromedia xRes

This very capable bitmap editor (`http://www.macromedia.com/software/xres/`) for Windows 95 or NT and the Macintosh can be purchased alone or as part of a bundle with Macromedia Director or Macromedia FreeHand. In many respects, this program is similar to PaintShop Pro, but it contains more high-end features such as layers and paths. Its tight integration with other Macromedia products makes it especially attractive for people working with Director or FreeHand on graphics for Shockwave-enhanced Web sites. In addition to support for Photoshop plug-ins, xRes has the special capability to interpret single-page PostScript files. As with Fractal Painter, you can create image maps directly in the program.

Corel Photo-Paint

This bitmap editor (`http://www.corel.com/products/graphicsandpublishing/`) comes bundled with CorelDraw and is currently at Version 7 on Windows 95 and NT, at Version 6 on the Macintosh as a similar application (Artisan), and at Version 3.5 on UNIX (Xpaint and Photo-Paint). Most of the following comments relate to the Windows Version 7. Corel Photo-Paint can also be purchased as a standalone product for the Windows platform.

Photo-Paint is a high-end bitmap editor supporting layers, masks, and grids for object placement. Photo-Paint gives you very fine control over transparency and has made adding drop shadows automatic. It supports the Adobe Photoshop plug-in standard and comes with a large number of built-in filters. It also includes an image painter similar to the Image Hose found in Fractal Painter. For Web images, it supports the Web color palette, image map creation, transparent and animated GIFs, and the newer Progressive JPEG image format. If you purchase the CorelDraw software bundle, you probably won't need to look at other bitmap editors. Hardware requirements are a bit steep, with 32MB RAM recommended for Windows and 20MB minimum for the Power Macintosh with System 7.5 or later.

Vector Packages

Vector graphic files are not directly supported as image types under HTML 3.2. Presently, there is no Internet-native file format for vector images filling the same role as GIF or JPG files do for bitmap graphics. In many ways, vector graphics are ideal for Web use. Their file size is usually smaller than bitmap files, improving browser page load times, and they are infinitely scalable without resolution loss, which would banish the "jaggies" from Internet graphics forever. For the time being, however, if you create a graphic with a vector package, you will need to export the file to a raster format or use a file conversion utility to rasterize it to a GIF or JPG. Browser plug-ins are available that will allow viewing of a vector file from its native format within a browser, but for an Internet site, it can be a great inconvenience to require all visitors to have a particular plug-in. This can be accomplished, however, within the narrower confines of a company intranet. I fervently hope that a standard vector format is adopted in the near future for use in HTML pages.

Because the vast majority of graphics on the WWW currently are in a bitmap format, you might be wondering why you should even consider shelling out the money and time to learn a vector package. Vector packages give the Web author some unique advantages:

- Most clip art is in some kind of vector format, and many vector editors come with large libraries of clip art images.
- You can easily disassemble, modify, or combine clip art images or parts of images to get more unique images.
- It is very easy to create primitive shapes such as circles, squares, or polygons.
- You can resize vector files without resolution loss and export them as a bitmap image in the precise dimensions you need for your Web page. This avoids extensive resizing in a bitmap program, which can lead to a loss of detail and a nasty eruption of the "jaggies."

If these capabilities look appealing to you and you want to add them to your bag of tricks, please read on; otherwise, you might want to proceed to the next section on an essential graphics utility for the Web author.

As with bitmap packages, vector packages have many types of tools in common. Some of the common features that you should look for are

- A pencil or line drawing tool and extensive means to modify line thickness and color
- A shape drawing tool or group of tools
- A node editing tool to modify curves and lines
- Transformation tools, including those for rotation, mirroring, and flipping
- A zooming and magnification tool
- A text tool for adding and manipulating text
- The capability to fit text to shapes, objects, or paths
- The capability to trace or outline bitmap files to create a vector file
- The capability to fill images with color, gradients, textures, or bitmaps
- The capability to combine, group, and ungroup elements
- Page layout tools and grids for precise placement of elements
- The capability to export images at a user-specified size to a bitmap format
- Special effects, including fitting text to a curve, extruding, perspective, and defining clipping paths

You do not need to subdivide vector graphics programs into those with an editing or painting emphasis as you do bitmap programs. Vector programs use more of the drawing metaphor and have fewer of the painting capabilities seen in bitmap paint programs. This is starting to change, however, most notably with Fractal Expression (`http://www.fractal.com`), which allows many special painting capabilities (using oil, watercolor, chalk, and so on) to enter the vector world for the first time.

Again, I've chosen the vector packages discussed in the following sections based on their popularity and capabilities. All vector programs I've found run under Windows or on the Macintosh. There are UNIX versions for each of the two most popular vector programs, Adobe Illustrator and CorelDraw. I am not aware of any vector editors for use under OS/2.

CorelDRAW!

CorelDRAW! (`http://www.corel.com/products/graphicsandpublishing/`) is a high-end vector program that comes in a bundle by the same name along with Photo-Paint, Dream 3D (a 3D rendering program based on Ray Dream Designer), and Presents (a multimedia presentation program). In addition to improved performance over earlier versions, Version 7 for the Windows platform contains several new features, including enhanced and flexible transparency controls and the capability to apply Photoshop plug-ins to imported bitmaps. There are also several new capabilities geared toward the Web graphics user. You can directly export a vector graphic as a transparent GIF or in Progressive JPEG. You can also export an entire page as HTML or to Barista, Corel's new Java-based format.

The bundle also includes 32,000 pieces of clip art or symbols, 1,000 photos, and 1,000 TrueType and Type 1 fonts. It also contains Corel Depth, which allows the quick and easy creation of 3D text, and Texture for creating an infinite number of natural textures. There is also a Script editor and a multimedia manager.

CorelDRAW! is a moderately difficult program to learn, but worthwhile when you consider the entire package and the unique Web capabilities not found in other vector packages.

Recommended system requirements include Windows 95 or NT on a Pentium with 32MB RAM. Version 6 requires a Power Macintosh, System 7.5 or later and a minimum of 20MB RAM. The UNIX Version 3.5 will run under HP UX 9.0 or greater, a DEC ALPHA OSF/1, Version 2 or higher, Sun Solaris 2.3 or higher, IBM AIX 3.2 (RS/6000) or greater, and IRIX 5 (SGI) or higher. The Macintosh and UNIX bundles differ in content from the Windows Version 7 bundle.

Adobe Illustrator

As an Adobe application, Illustrator (`http://www.adobe.com/prodindex/illustrator/main.html`) started life on the Macintosh and now also runs under Windows. Currently, at Version 7 on both platforms, this program has a unique knife tool that can cut objects along a freeform path. You can mask objects with any shape and convert spreadsheet data to graphs. You can import bitmap images and apply any Photoshop plug-in filter effect to bitmaps imported into Illustrator. You can export images to GIF or transparent GIFs and read and edit PDF files.

The program is moderately difficult to learn. Features of interest to the Web author include

- The capability to assign a URL to an object and have the program create a link to an image map
- The built-in 216-color Web palette
- Export files to JPG or GIF89a
- Opens and exports to the PNG format

The recommended system for the Macintosh is a 68030 processor or better with 16MB of install RAM and 8MB of RAM available for the application or a Power Macintosh with 32MB of installed RAM. For Windows 95 or NT, the recommended system is a Pentium under Windows 95 with 32MB of RAM. The program will not run under Windows 3.1. For UNIX (Solaris 2.3 or 2.4), a Sun SPARCstation 2, IPX, or faster processor workstation with OpenWindows or Motif and 32MB of RAM is needed.

Macromedia FreeHand

FreeHand 7 for Windows or Macintosh (`http://www.macromedia.com/software/freehand/`) can be purchased alone or as part of the FreeHand Graphics Studio that also includes xRes, Fontographer (for creating and modifying fonts), and Extreme 3D (for 3D rendering and animation). FreeHand graphics can be added to "shocked" Web pages, which incorporate

Macromedia's Shockwave technology. Shockwave is a plug-in that maintains the vector nature of FreeHand files, allowing for image zooming on a Web page and for independent treatment (such as hyperlinks) of each object in the graphic.

FreeHand supports multiple levels of undo and has the unique capability to make styles that include both text and graphical style elements such as spacing and line specifications. These styles can then be applied to other objects or text. You can import bitmaps into FreeHand and apply Photoshop standard plug-in filters. You can do Search and Replace operations on graphics items. Scripting in Java or AppleScript is available. You can also save graphics as PDF files.

In addition to the applications mentioned, the FreeHand Graphics studio also contains 10,000 pieces of clip art, 500 fonts, many templates, stock photographs, and 3D models.

Recommended system requirements for Windows 95 or NT include 16MB of RAM; for the Macintosh, System 7 or higher and 6MB of application RAM are required.

Other Vector Packages

The number of vector-style graphics programs not already mentioned is far smaller than the bitmap crowd. Some packages contain fewer features than those discussed, but they might be easier to learn. Corel Xara (`http://www.xara.com/corelxara/`) fits into this category. Other very high-end programs, such as CAD programs, have little day-to-day utility for most Web authors. No matter which program you choose, I do recommend that you obtain a vector-style graphics program to give yourself the most capability and flexibility in editing or creating graphics for your HTML pages.

An Essential Graphics Utility for the Web Author: Debabelizer

Even if you have high-powered bitmap and vector editors, you still might not have all of the tools you need for preparing Web graphics. Most traditional editing programs concentrate on maximizing content and detail, both of which are usually essential for sharp-looking graphics. This, unfortunately, often leads to large file sizes. Most users on the WWW, however, will not tolerate slow downloads, no matter how good looking your graphics might be. Until this bandwidth problem goes away, you have to make graphics that are small and fast loading as well as attractive. You can reduce GIF or JPEG file size using one of three methods:

- Have fewer pixels; that is, make the image smaller.
- Reduce the number of colors.
- Use compression techniques.

This is all complicated by the fact that many browsers support only a 216-color palette (the "Netscape" or "Web" palette), which means that for optimal viewing on all browsers, all of the graphics that you use need to be mapped or dithered to this palette. The need to make some combination of these problems work together for all of the graphics you use on any particular Web site and still have your graphics look good has opened a niche market for programs that optimize graphics for use on the Internet. The first horse out of the gate, and, in this case, the best tool to get this done, is Debabelizer.

This product by Equilibrium (`http://www.equilibrium.com`) has long been the target of envy from Windows users because it initially was available only for the Macintosh. Now it is also available for Windows 95 and NT. Its capability to automate many of the image reduction steps saves you countless hours in adding graphics to your Web site and ensures that your graphics will be of the smallest possible size and have the same appearance across different browsers.

You can use the SuperPalette in Debabelizer to create an ideal palette for a group of images and map them all to it. An HTML parser will place all of the files from a Web page onto a list for SuperPalette creation. You can use scripts in Debabelizer to automate the performance of a series of effects to a group of images, such as changing file formats to transparent GIF or Progressive JPEG, reducing colors, palette mapping, and so on. The number of ways to change and customize palettes is unrivaled by any bitmap editor. The Windows version also provides a wizard to optimize graphics for Web use.

Debabelizer also has basic bitmap editing functions and supports the Photoshop plug-in standard. Other non-Web capabilities in the program include a host of capabilities for applying effects to video, including text overlay and blue screen removal.

Recommended system requirements for the Macintosh are at least 2.5MB of application memory and System 6.0.7 or higher. For Windows 95 or NT, a Pentium and 32MB RAM are recommended.

Animation and Video Packages

Moving graphics on a Web page can be a major source of interest, perhaps drawing attention to an important item, or they can be irritating, even going so far as driving the user off your page. If used properly, moving images, whether they be graphics or video clips, can add considerable interest and content to a Web site. So you will need to know what tools to use to create or modify moving picture content for use on an HTML page.

Four primary types of moving picture technology are available for use in a Web page:

- Animated GIFs
- Standard video files, Video for Windows (AVI), or QuickTime movies (MOV)
- Plug-in support for proprietary animated graphics packages
- Streaming video

Each of these involves a trade-off in file size versus frame rate versus picture quality. Using AVI and MOV files in Web pages isn't discussed in detail here because relatively few users on the Internet have the bandwidth to view these in real-time at an acceptable frame rate. Most users have the ability to download and view these files, but their large file sizes for even short clips makes even this offline viewing inconvenient.

GIF Animation

Animated GIFs are composed of multiple GIF frames in one file, as supported under the GIF89a file format. Because even a single GIF file can easily exceed 50KB, the size restraints on animated GIFs are fairly severe. Most animations are small, or only a small part of the image is animated. And the number of frames must be minimized to maintain a small file size. The process of making an animated GIF is very laborious. You must plan the animation (how many frames, what will move, how will it move, and so on), generate each frame in a bitmap editor or paint program, assemble the files in an animation construction program, and add the display time for each frame. This process is very similar to how cartoon animation was done several decades ago. If you want to try your hand at it, there are several packages available to choose from.

GIF Construction Set

This shareware package from Alchemy Mindworks (`http://www.mindworkshop.com/alchemy/gifcon.html`) for Windows 95 or NT gives you complete control over all aspects of the animation. After you have created all of the frames for your animation as separate GIF images, you can open a new image in GIF Construction Set and begin inserting the frames one by one. GIF Construction Set has several palette options, including the capability to generate a global palette based on your first image and map the remaining images to that palette. A block metaphor is used to build the animated GIF. A header block is created in a new file and contains some basic file information, such as image size. A comments block can be added for notes (text will not display). Each frame image is an image block, and a control block is needed for each image to control the frame display time for that block. Plain text blocks enable you to add text that will display in the animated GIF, and a loop block will control how often the animation loops or plays. After you get the animation to run as you wish, GIF Construction Set will save the file in the 89a format so that it will run as an animation in a browser.

GifBuilder

This freeware program for the Macintosh, authored by Yves Piquet (`http://iawww.epfl.ch/Staff/Yves.Piguet/clip2gif-home/GifBuilder.html`), makes it easy to create an animated GIF. After you have your frames, open GifBuilder and start dragging and dropping your files into it from their folder. You can change the order by dragging the filenames around if you need to. You can then use the Options menu to set the looping, palette, and interframe delay. You can select all of the files and set the same delay time or select individual files and set a separate time for each frame. You can also easily change the background of the animation to transparent. Save your file and you're finished.

Microsoft GIF Animator

Microsoft offers a high-quality GIF animation program for free on its site at `http://www.microsoft.com/imagecomposer/gifanimator/gifanin.htm`. This program is designed to work optimally in tandem with Microsoft Image Composer, but it can easily be used as a standalone program.

The program is very easy to use. You can import any image that you can paste to the clipboard, import any GIFs, or drag and drop from Image Composer. You also can import the image palette or select a browser palette. You can resize your animations and select the number of times they will loop. You can select one color to be transparent. After you've added all of your frames, you can easily rearrange them by selecting a frame and clicking on an up or down arrow. Animated GIF building doesn't get much easier than this, and you can't beat the price.

Other GIF Animation Programs

The preceding are just two programs of many available on the Internet. For UNIX, MultiGIF by Andy Wardley (`http://www.peritas.com/~abw/code/multigif.html`) works as a command-line utility to generate an animated GIF. Source code is also available. Consult Appendix D for more links to GIF animation builders.

Macromedia Flash: Proprietary Animation on the Web

This recent addition to Macromedia's line (`http://www.macromedia.com/software/flash/`) came from the purchase of FutureSplash. It requires a browser plug-in (80KB to 150KB in size depending on platform) to work but produces such high-quality, low-bandwidth animations that it should be seriously considered by Web authors as an alternative to animated GIFs. A major advantage to this format is that it streams, giving the user some immediate visual feedback, and continues to play as it downloads. Creation of animations is also eased by use of *Motion Interpolation*. All you need to do is define the starting and ending points and rotation for the object and Flash generates the in-between files. No need to laboriously create each frame as with an animated GIF, so you can spend more time experimenting. You can also embed links in any portion of the animation.

Recommended system requirements for authoring on a PC are a 486 or faster and 16MB RAM; For the Macintosh, a 68040 processor or faster, System 7.5 or later, and 16MB of RAM are required.

Macromedia Director: Proprietary Multimedia for the Web

This package (`http://www.macromedia.com/software/director/`) is a full-featured, multimedia authoring package for doing things such as producing CD-ROMs. This high-bandwidth technology is of interest to Web authors because of the availability of a Shockwave plug-in and Macromedia's Afterburner technology, which can compress a Director file to a size acceptable for Internet use. This gives you the option of putting more sophisticated media on your Web site. The program is based on a theater metaphor and allows you to combine a wide variety of media types—including audio, video, graphics (2D and 3D), text, and database objects—and script them with a built-in language, Lingo. Director is used by many Web sites to produce games and simulations. Because of its many features, the program will be moderate-to-difficult to learn.

System requirements for Windows 95 and NT are a 486 processor or better with 16MB RAM. For the Macintosh, a 68040 or faster, System 7.1 or higher, and 16MB RAM are required.

Streaming Video

The high bandwidth requirements of playing standard video in real-time at an acceptable frame rate makes the process unavailable to most Internet users. New technologies that stream and compress video, however, now make it possible to use video in real-time broadcasts or to play video clips to users who only have modem connections to the Internet. Although the video quality, size, and frame rate are all only of medium quality in the best of situations, look for this important technology to improve with time. I have chosen VivoActive and RealVideo as examples of this genre of applications.

VivoActive

VivoActive (`http://www.vivo.com`) is an example of *serverless video*, meaning that you don't need a special Web server to feed the video to the browser and RealVideo, which uses the server form of video streaming.

VivoActive from Vivo Software uses a program, VivoActive Producer, to transform AVI or QuickTime Movie video clips to a new format. Compression rates are quite impressive, with a 30MB AVI file reduced to less than 100KB by Producer. This compressed file is then read and streamed through the Vivo browser plug-in when a link is clicked or a page is loaded. If you have a modem connection to the Internet at less than 28.8kbps, you can expect significant delays during the playing of the Vivo clip. Video and audio quality are both acceptable. Unfortunately, at this time, players are available for only Windows and Macintosh platforms. A MIME entry is also needed in the Web server table. The authoring software, Producer, is available for Windows and Macintosh.

RealVideo

RealVideo (`http://www.realvideo.com`) is a recent development from the folks at Progressive Networks who brought you RealAudio. This form of video requires a dedicated video server to stream the video in response to a browser request. Video is processed using a set of Encoder tools before being placed on the server. Many high-end, professional options are available for processing, including support for Adobe Premiere plug-ins and Terran Interactive's Movie Cleaner Pro, among others. Command-line batch processing is available for automation of the process. Efficient playback often requires using the UDP protocol rather than TCP, which may necessitate reconfiguring any firewalls. The player will not work if the user connection is substantially below 28.8kbps. Players are currently available only for Windows and Macintosh, but this is sure to change.

Ray Tracing, 3D Rendering, and Fractals

At times you might not be able to get the effects you want with regular bitmap or vector graphics programs, especially when it comes to generating 3D images or fractals. Programs that render in 3D also enable sophisticated lighting and texturing support, which adds a very realistic touch to an image. Fractals can be used to generate textures, generate landscapes, or just to make that cool graphic you need to decorate a particularly blah part of your Web site.

POV-Ray

POV-Ray (Persistence of Vision Ray-Tracer) is freeware (`http://www.povray.org/index.html`) and belongs to a group of programs called *ray tracers*. A ray tracer renders a scene by shooting rays into it. A scene is built from shapes, lights, a camera, textures, and other items. POV-Ray uses ASCII scripts to define each of these elements. The elements are then rendered into a very high quality and, often, attractive 3D image. You can generate many types of effects including fog, fire, and steam. Simple and advanced primitives are available, as are many predefined patterns and textures. Many example files are included with the program. It is not difficult to learn how to build the scripts using the tutorial. This program has proven addictive for many people, so handle with care.

POV-Ray runs on Macintosh, Windows, and UNIX boxes. For PCs, you need Windows 3.1 or later, a 386 or better CPU and 8MB RAM minimum. A DOS version is also available. For the Mac, you need a 68020 CPU (with or without FPU) or Power Mac, System 7 or better, and at least 8MB or RAM; for Linux, you need a minimum 386 with 4MB RAM. POV-Ray also is available for the Sun OS and the Commodore Amiga.

Ray Dream Designer

Ray Dream Designer (`http://www.fractal.com/products/rds/index.html`) from Fractal Design Products (the same engine is also used in Corel Dream, part of the CorelDraw bundle) is a visual 3D modeling program. You use many of the same elements as POV-Ray such as lights, cameras, and textures (called shaders here). But you interact with the shapes and other elements on the screen in whatever level of detail you want, from a wire frame view to a mostly rendered view. You can grab the 3D object and rotate it, apply different textures, try different lights or camera angles, and so on. Then when you have it just the way you want it, you can render it into a final image. The package comes with several hundred premade models and two wizards: one for automatically creating indoor or outdoor scenes and another that will lead you step-by-step through building your model. If you purchase Ray Dream Designer as part of Ray Dream Studio, you can also animate almost all aspects of your 3D model and scene. Scenes from this program can be directly exported to VRML as well as GIF or JPG.

To run this program on the Macintosh, you need a 68040 processor or Power Mac (no FPU required), System 7.0 or higher, and a minimum of 12MB application RAM; for Windows 3.1, 95, or NT, you need a 486 or Pentium CPU with 16MB RAM for best performance.

Fractint

Fractal images are the pictorial representation of fractal geometry or sets of mathematical points. These iterative formulas can generate very complex, abstract, and beautiful graphics. Using Fractint (DOS) or Winfract (both freeware from The Stone Soup Group), you don't have to worry about drawing anything; let the equations do it for you. By using different equations, changing parameters in the equation, or using different color maps, you can generate some striking images. You do not need to be a math wizard to do this. All of the equations and color

maps are named. The images can then be saved as GIFs and used on your Web page. You can find the DOS or Windows version at many file archives on the Internet such as `http://filepile.com`.

Similar programs are available for UNIX. Xfractint (`http://spanky.triumf.ca/www/fractint/xfractint_port.html`) is an X-windows port of Fractint and Xmfract is a Motif, multiwindowed version. A Macintosh version is not available. Only basic system resources (no FPU required for most equations) are required on all platforms.

Summary

The choices for graphics packages are many and varied. In this chapter, you've seen

- Bitmap editors and paint programs
- Vector-style graphics packages
- A key graphic utility, Debabelizer
- How to add moving pictures to a Web page with GIF animation or streaming video
- Some proprietary solutions for adding animated graphics or multimedia
- 3D rendering programs and fractals

If you work with a team and most of the drawing or graphics creation is done by a professional, you might need Debabelizer only for palette touch-up or image resizing. If you plan on creating more graphics yourself, choose the set of tools that will help you get your job finished the easiest and fastest. If you have no formal artistic training and find yourself doing a lot of graphics creation, I highly recommend learning more about color and design techniques either through further reading, hanging out with artistic types, visiting award-winning Web sites and examining their appeal, or taking courses at your local institution of higher learning. Unfortunately, many Web authors understand how to use many high-powered graphic packages to create very low-powered Web garbage graphics. Learning to create effective graphics is just as important as having the right tools. Give a hoot—don't pollute the Web.

CHAPTER 36

Advanced Web Authoring Tools

by Rick Darnell

IN THIS CHAPTER

- Microsoft FrontPage *730*
- Macromedia Backstage *742*
- NetObjects Fusion *751*

In the previous chapters of this section, we've discussed various tools and utilities that help you manage your Web site and pages, including creating pages, creating images, checking links, and validating HTML.

Currently, three major Web authoring suites are available to developers on Windows and Macintosh platforms. The first is Microsoft FrontPage, which Vermeer Technologies originally developed and then Microsoft purchased. Microsoft has since integrated the package with its other software applications, including the popular Microsoft Office.

Another option is Macromedia Backstage. Backstage doesn't offer the additional feature of integration with other applications, but it is still a powerful set of programs in its own right. It includes an add-on server that supports the Web server in delivering dynamic content and supports HTML files located on the local machine and on remote Web servers.

The third Web development suite is NetObjects Fusion by NetObjects software. This package, like Microsoft FrontPage, is available for both Macintosh and Windows users. Furthermore, it's currently the only suite that provides for pixel-level precision of your Web page design.

NOTE

Both Web management programs I discuss in this chapter provide options for creating dynamic HTML pages. A traditional HTML page is a static creature. After it's created, it doesn't change, similar to a newspaper, in which the words don't change after they are set on the page in ink. You can go back to look at the words as many times as you want, but they'll still be the same pages in the same order.

A dynamic HTML page can be different every time you see it, similar to the difference between the five and ten o'clock evening news on television. The format is the same, but many of the stories are changed or updated from the earlier broadcast.

Microsoft FrontPage

FrontPage, an advanced Web management and authoring tool, features an integrated set of tools that are designed to help you in creating, implementing, and managing Web sites.

It includes multiple author and administrator capabilities, enabling system administrators to spread tasks across a larger group of people and still maintain a grasp on what needs to be done and what is complete.

FrontPage has two major components: FrontPage Editor, a WYSIWYG HTML editor that eliminates the need to remember the nuances of each markup tag, and FrontPage Explorer, which provides a visual and intuitive interface that tracks links, pages, images, and other page components.

NOTE

How do you want to Explore today? With FrontPage, Microsoft now has three applications that go by the name *Explorer*: the file management utility packaged with Windows 95, Windows Explorer; the browser known as Internet Explorer; and, included with FrontPage, FrontPage Explorer. Any mention of Explorer in this chapter is a reference to FrontPage Explorer. I will refer to all other Explorers by their full name.

TIP

The three primary FrontPage components–Explorer, Editor, and To Do List–won't work without a connection to your Web site through a Web server. FrontPage provides the Personal Web Server for use on individual computers. For actual deployment and use, you'll want to use a more advanced server, such as Microsoft Internet Information Server or O'Reilly WebSite. For more information on the strengths and weaknesses of various servers, see Chapter 37, "Web Servers."

FrontPage also provides a set of Web robots, commonly referred to as Web bots, which are specialized CGI scripts that provide implementation for discussion groups, full-text Web searches, and form handling.

As with many Microsoft products, FrontPage makes extensive use of wizards to convert tasks into a series of questions and choices, such as creating Web sites and pages.

FrontPage Explorer

The central component to the FrontPage suite is Explorer. Explorer provides a visual interface for administering your Web projects. Depending on your Web site's size and organization—whether it's an Internet site or an intranet implementation—Explorer provides an intuitive interface for creating, editing, and deleting pages and verifying hyperlinks. (See Figure 36.1.)

Explorer's initial view serves as a type of Web control panel. It shows your Web in Hyperlink View, which is divided into two parts. On the left side of the screen is an outline view that shows each page included as part of the site. Clicking a page in this view makes it the center of the link view on the right side of the screen.

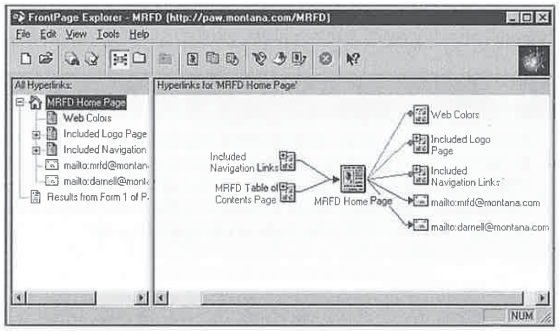


FIGURE 36.1.
FrontPage Explorer serves as the front end to all other FrontPage applications in addition to providing a graphical view of your site's construction and links.

NOTE

Which web is which? FrontPage refers to each set of related directories and files as a web. So, if you've created an area on your company server for each of three departments using FrontPage, each area is its own web.

In essence, each of the FrontPage webs is a Web application—a group of HTML pages that serve a specific purpose, whether it is customer service, corporate presence, or hosting a discussion group.

The link views show a page and all links to and from it. All items to the left of the page are links from another source on the current web, and all items to the right are destinations from the current page. If a page has more links that are not displayed, it is marked with a plus sign (+).

You can also display the Web site as a series of directories by selecting the Folder View button. Initially, doing so only displayed other HTML pages and CGI scripts. You can also choose to display links to images, multiple links to the same page, and links to anchors within the current document.

You can add specialized items to any web. For example, if you're creating a department site on an intranet and need a discussion group, you would start the Discussion Web Wizard and select the checkbox for adding it to the current web. Any web can be moved to a server for publication on an intranet or Internet.

One of FrontPage's strongest features is its selection of wizards and predefined Web templates to quickly and easily build a Web-based application.

Explorer Templates

Templates are preconstructed webs with a predetermined set of pages already in place. FrontPage includes the following four templates that fit this description, plus an option to create a Web site completely devoid of any page:

- Normal Web: A web template that consists of a single page and a folder for images.
- Customer Support Web: This template is designed for Web sites that serve a high volume of technical support inquiries. This web helps to organize and improve Web-based customer service.
- Project Web: This template creates a web for a specific project, including lists of contributing members, status reports, schedules, discussions, and archives.
- Personal Web: This template is a starting point for creating a set of personal Web pages, including a home page. In effect, it's the same as the Normal Web.

FrontPage also includes an Empty Web, which sets up the necessary support directories for a FrontPage web but doesn't include any HTML pages.

Explorer Wizards

Web Wizards are similar to templates, but instead of a one-size-fits-all approach, it generates a customized set of pages based on your response to a set of questions about your intent for the new web.

- Corporate Presence Wizard: This wizard generates a set of pages and directories to establish a basic Web presence for a company. You can also tailor it for individual departments on an intranet server. It provides templates for products, services, department information, mission statements, and personnel directories.
- Discussion Web Wizard: This wizard sets up a portion of your server for hosting a Usenet-type discussion group without installing a dedicated news server. It includes support for threads, tables of contents, text searching, and administration.

There's a third Web Wizard, the Import Web Wizard, although it is slightly different in purpose than the first two. The Import Web Wizard creates a place for a new FrontPage Web and then copies files from another directory on your system or from a remote server.

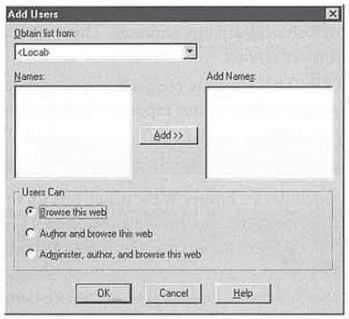
FrontPage Security

When you use Explorer to manage a site, you must ensure that every web has at least one administrator who is identified with a name and password when the web is first created. The web administrator controls access by setting permissions for end users, authors, and other administrators, such as the following:

- End users: By default, a web is open to everyone. If it's necessary to control who has access to a web, administrators can identify individuals to be excluded or included for browsing. If a user attempts to browse a protected web, he or she is prompted for a username and password before he or she is granted any access.

- Authors: Web authors can create, edit, or delete pages, but they cannot create, change, or remove webs. By virtue of this level, they're also included as end users.
- Administrators: Administrators have permission to create and delete webs, pages, and authors, and to designate other administrators. (See Figure 36.2.) They also set which users have access to browse the finished product.

FIGURE 36.2.
In the Add Users box, define which users have access to manage or browse the Web site.



By default, security settings are inherited from the root web. All administrators, authors, and end users defined for the root web are valid for any other web created under the root web unless otherwise specified. A user's permission level is checked any time a web page is accessed for any reason.

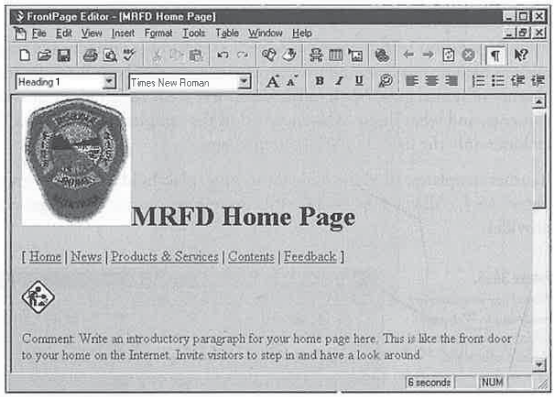
In addition to screening name and password information, you can screen all permissions by entering a mask IP address. Using a series of numbers and wildcards, you can restrict access to specific computers or groups of computers on the network. Even if a perpetrator knows a name and password combination, he or she must still use the right computer to gain access.

FrontPage Editor

FrontPage was one of the first HTML editors to offer a WYSIWYG interface for creating Web pages. It doesn't require that you have any knowledge of HTML, even to create complex pages that include forms and embedded objects such as ActiveX controls and Java applets.

To launch Editor from Explorer, click its icon from the toolbar or double-click a page icon. When it is opened, Editor manages virtually all page editing tasks from the toolbar, which includes buttons for text formatting, hyperlinks, tables, forms, Web bots, and embedded objects. (See Figure 36.3.) Access other page features, such as a page title, default text and link colors, and background color, from the Edit menu.

FIGURE 36.3.
FrontPage Editor provides a graphical interface for creating Web pages with text, graphics, forms, tables, and other specialized content.



Working with Editor is much like working within a word processor. You can type, cut, and paste any text on the page. It also provides advanced formatting of the text, including changes in size, color, and typeface.

TIP

Even though FrontPage supports advanced text formatting features, the final results might not appear as you expect on all browsers. Be sure to test your pages on a variety of Web browsers to ensure consistency and compatibility.

One feature that makes Editor especially easy to use is drag-and-drop hyperlink creation. If you need to copy links from Explorer or other Web pages, click and hold the desired page or link and drag it to the appropriate location in the editing window.

Editor supports forms in two ways. First is the Form Page Wizard (described later in this chapter). Through a series of questions and checkboxes, the wizard asks which kind of information you need and assembles the elements into a finished HTML page. You can also create a form manually by inserting individual HTML form elements. After the first form element is inserted, it's surrounded by dotted lines to denote the boundaries of the form. Then, you can specify which CGI script will process the form's contents.

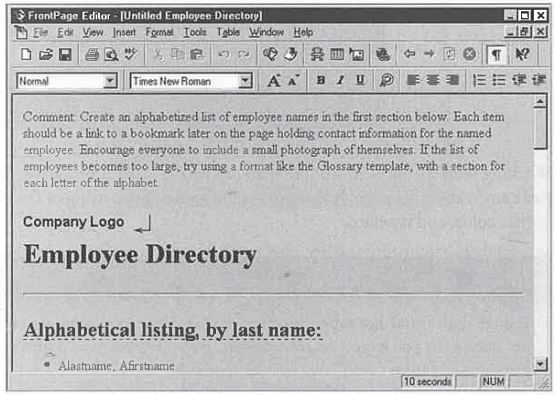
Like Explorer, one of Editor's strongest features is its use of templates and Wizards to speed the development of individual pages.

Editor Templates

FrontPage Editor includes a set of templates that cover virtually any page publishing need. Its options include bibliography, employee directory, feedback form, frequently asked questions, glossary of terms, guest book, office directory, event registration, software data sheet, table of contents, and what's new. Also included in the template selection is a "Normal page," which includes only the basic HTML structure tags.

In other templates, all of the basic formatting, placeholder text, and graphics are in place. (See Figure 36.4.) All you have to do is substitute your content for the generic material that Editor provides.

FIGURE 36.4.
FrontPage templates are ready-made Web pages, which make it faster to develop new pages.

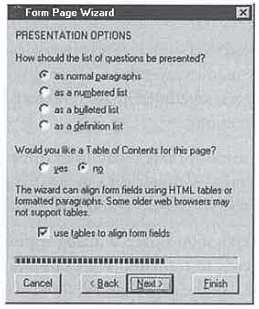


Editor Wizards

Editor includes four wizards to walk you through creating pages that normally require extensive HTML coding or CGI scripting. The following are the four wizards:

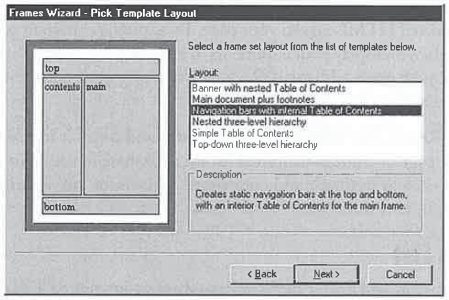
- Database Connector Wizard: Using an ODBC connection and a special template file created with Editor, a developer can create an Internet Database Connector (IDC) file. The IDC enables Structured Query Language (SQL) queries (including database additions and modifications) to any compliant database accessible from the Web server machine.
- Form Page Wizard: Similar to the Home Page Wizard (discussed in the last bullet), this Wizard prompts you for the kind of information you want to collect from users. You can choose to submit the information to an e-mail address or CGI script or post it to a Web page for later viewing. The wizard completes the process by adding text and formatting the page, which results in an attractive, immediately usable form. (See Figure 36.5.)

FIGURE 36.5.
The Form Page Wizard is one example of a tool that builds a ready-to-use form based on your answers to a series of questions.



- Frames Wizard: Frames are becoming more accepted as a serious design tool for Web pages. The Frames Wizard visually defines how the frames should be placed within the browser window (as shown in Figure 36.6) and creates the HTML code to implement the result. Chapter 18, "Creating Sophisticated Layouts with Frames and Layers," provides more information about using frames.

FIGURE 36.6.
The Frames Wizard builds the HTML needed to present a series of frames to the user in one of several predetermined layouts.



- Personal Home Page Wizard: By asking a series of questions about you and your activities and interests, this Wizard constructs a one-page document that serves as your personal Web page. There are enough options that you can use this either for a strictly personal situation or as an online biography/resume for corporate and business situations.

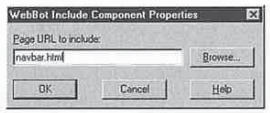
Web Bots

Web robots, also called Web bots, are objects that you insert on a page to include capabilities that would normally require extensive CGI programming. The list of capabilities includes inserting other Web pages, scheduled images and text, discussion groups, tables of contents, time stamps, and other functions.

You can implement some of these capabilities (time stamps, scheduled images, inserting other Web pages) by placing a Web bot directly on the page, but to implement others, you must integrate them through Web wizards and form handling (discussion groups, feedback forms).

After you insert a Web bot on a page, a dialog appears to collect the configuration information that the Web bot needs to function. For example, if you insert an Include Web bot, you're prompted for the URL of the page that will show when the page is displayed. (See Figure 36.7.)

FIGURE 36.7.
FrontPage uses Web bots to add functionality to Web pages, such as inserting another Web page at runtime.



You would also include other Web bots during specific actions, such as adding comments or unsupported HTML tags to your page. By adding a comment to a Web page, you put a Web bot in place to display your comments in a highlighted color. This means that you can see what notes you've made while you're actually working on the document, even though they're not displayed when the user views the page.

A Save Results Bot takes input from a form and stores it in a location you define, such as an HTML page or comma-delimited list for a database to use. This makes it easy to collect information such as feedback and comments, registration information, and other end-user information.

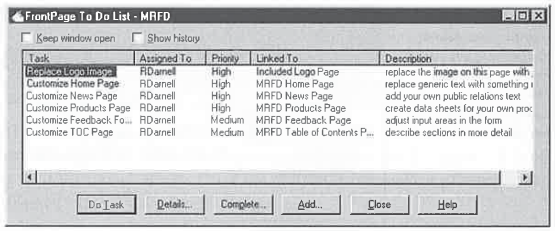
To Do List

The Explorer and Editor in FrontPage share an integrated To Do list to help with delegation and tracking of Web editing tasks. The list includes the task, who is supposed to do it, and the priority for completion. (See Figure 36.8.)

To more easily work directly from the list, you can highlight any linked job and click the Do Task button to launch Editor and load a copy of the page for editing. When the page is saved, you're prompted to remove the associated item from the To Do list.

You can extend the usefulness of the To Do list by using Editor to include comments to the affected Web page and to describe in greater detail what needs to be done or corrected on the page. The comments will only appear during editing, not when the page is displayed.

FIGURE 36.8.
The To Do List provides a way of tracking assignments on your Web site.



Bonus Pack Add-ons

In addition to the three core components of FrontPage, Microsoft also includes several additional programs and features with its product. These include a Web server specially designed for standalone or small group operation, a wizard for posting pages to your Internet Service Provider, and a new image utility. And, like every other Microsoft product available on the market, it includes a copy of the latest version of Internet Explorer.

Personal Web Server

Many Web authors are learning the usefulness of having a Web server installed on their desktop computer to develop and test their pages. The Microsoft Personal Web Server, a 32-bit Web server, is a useful answer to this need.

You can also use the Personal Web Server to operate a small Web site for a workgroup or small peer-to-peer intranets. This 32-bit application runs on either Windows 95 or Windows NT and requires a minimum of 8MB RAM and 1MB of hard disk space. It requires more disk space to hold Web pages.

Although it doesn't include many of the advanced features of a full-fledged Web server, it is more than adequate for developing and testing Web pages and forms. It includes support for handling pages for the HyperText Transfer Protocol (HTTP) and for working with forms and data for the Common Gateway Interface (CGI).

The Personal Web Server does not support secure transaction options. Although its official limit is five users, two to three is a more realistic option, depending on the operating system and available resources.

You can automatically install the Personal Web Server through the FrontPage setup window or by downloading the software from Microsoft at `www.microsoft.com/ie/iesk/pws.htm`. You manage the Personal Web Server through the Windows Control Panel and a browser interface. (See Figure 36.9.) Use forms and hyperlinks to gather information and update the server configuration files.

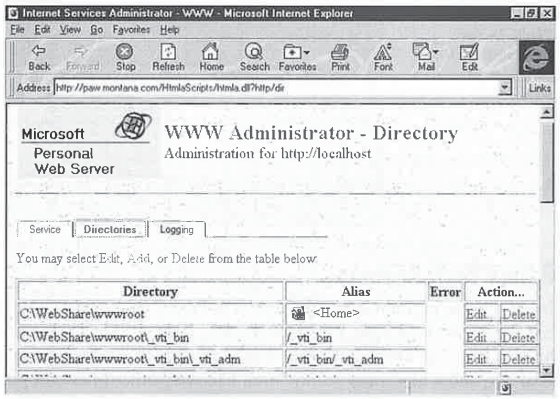
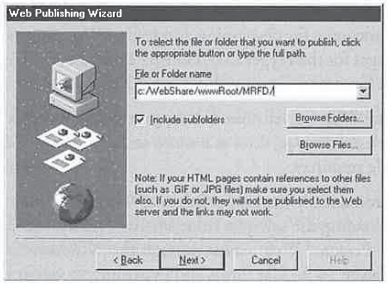


FIGURE 36.9.
Instead of a standard dialog box, a series of forms manages the Personal Web Server through a browser interface.

Web Publishing Wizard

After your Web pages are ready for viewing by the rest of the browsing world, you'll need a way to transport them to your Internet service provider. Using an FTP program can be a clunky process, so Microsoft developed the Web Publishing Wizard. This handy little utility enables users to easily post Web pages to their ISP or intranet site. It supports a wide range of providers, including CompuServe, Sprynet, and America Online, in addition to compatibility with Apache Web server, Microsoft Internet Information Server, and Internet servers that conform to NCSA HTTPD. (See Figure 36.10.)

FIGURE 36.10.
The Web Publishing Wizard simplifies the process of posting a set of files or directories to a remote Web site.



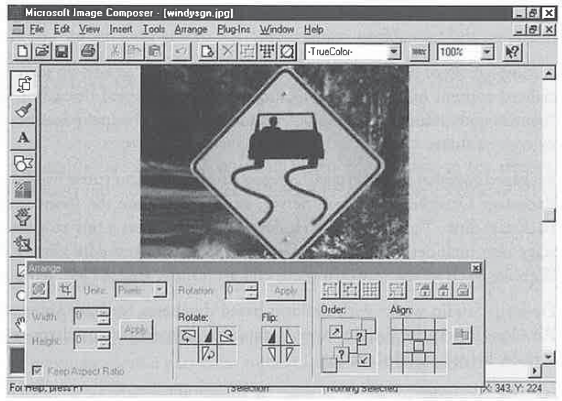
The Web Publishing Wizard connects to the ISP, determines the protocol it needs to copy the files, and then uploads the files to the appropriate directory on the Web server. To further simplify the process, the Web Publishing Wizard uses the same connection information as your browser. If you can connect to your ISP to browse, you can connect to update your Web pages.

Image Composer

With its new Image Composer software, Microsoft has moved past the Paint program it shipped as standard equipment with Windows. Where Paint is a multipurpose tool for print or screen, Image Composer is designed explicitly for screen images.

Like many other image editing utilities on the market, Image Composer treats bitmap images like objects, which makes editing and combining images for different effects easier. More than 600 images are included with the program, which you can layer or combine to create complex collages and effects that you can then use in your Web pages. In addition to the images, Image Composer also includes 500 graphic effects with which you can easily alter a picture's appearance to the desired effect. (See Figure 36.11.)

FIGURE 36.11.
Microsoft Image Composer is a graphics tool designed specifically for creating on-screen images, including GIF animations.



Although the available effects are on par with professional image programs such as Photoshop, they are easier to use thanks to a simplified pushbutton interface. This could lead to abuse and convoluted finished products if it is not used carefully.

Macromedia Backstage

Macromedia Backstage is the latest entry into the growing field of Web site development applications. Backstage draws its name from its behind-the-scenes approach to implementing Web pages and features. Using Backstage to create Web pages on a local or remote server is only the beginning of its capabilities, although it is an important capability.

Perhaps Backstage's strongest feature is its full support for database connectivity through simple-to-use objects. This enables you to post user information directly to a database from a Web page and generate reports directly from the data into other Web pages, all without learning a CGI scripting language.

Backstage is designed for both first-time Web authors and experienced developers. You can complete a basic site including advanced features and make it available to the browsing public in hours, not days.

HTML files created with Backstage are compatible with any Web server and any browser that can interpret HTML 2.0, such as Netscape Navigator, Microsoft Internet Explorer, and NCSA Mosaic. Support is also included for some HTML 3.2 extensions, such as tables, advanced image handling, and logical formatting codes.

A complete set of Backstage objects supported through an add-on server offer insertion of specialized content on your pages, including hit counters and discussion groups. In addition, a "conditional include" object enables you to tailor individual pages on the fly, depending on who views them.

Packaged together as Backstage Desktop Studio and Enterprise Studio, the basic elements—Backstage Designer, Manager, Server, and objects—are the foundation applications of the Backstage line. To make the Web development process a one-stop-shopping process, Backstage also includes a Web-optimized version of Macromedia xRes, a full copy of O'Reilly WebSite Web server, PowerApplets, clip art library, and HTML templates.

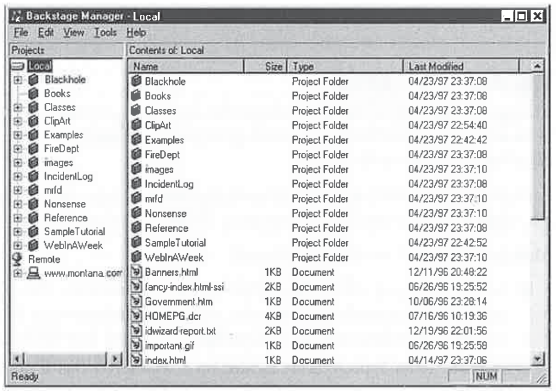
Desktop Studio works with desktop-based databases, such as Access, FoxPro, dBASE, and Paradox. Enterprise Studio extends database connectivity to client-server databases such as Oracle, Sybase, and Informix.

Backstage Manager

With Backstage Manager, you can administer your Web site and individual projects from a bird's eye view. You can create new content and edit existing files on local or remote Web sites.

Backstage Manager gives a structural view of your Web with each set of pages included in its own *project folder*. (See Figure 36.12.) This provides a central point for creating, editing, and managing your Web projects. As you use Backstage to navigate through different Web sites, local or remote, you can launch Designer to edit pages or other image, sound, and multimedia applications for other content items.

FIGURE 36.12.
Backstage Manager is a project-level tool that manages a Web site and some of its more advanced features, such as discussion groups and user profiles.



Most sites are administered through local access, but Manager can work with sites in remote locations through FTP server access. You can download an entire site for editing, disconnect from the server, and upload it later. Or, you can directly edit a page for spot changes.

One option for working with Manager for remote projects is to create all the files and structure you need on the local host. Then, copy the completed project to the remote site by dragging and dropping it onto the appropriate remote server icon. When you connect to the remote site, Manager will automatically upload the new files without working through a tedious FTP process.

Backstage divides each project into four elements:

- Web pages: These pages are HTML files and the related content within in them, such as images, applets, and plug-ins.
- Discussion groups: A Backstage discussion group, similar to a Usenet newsgroup, includes a series of messages about a particular topic. Unlike FrontPage's method of creating an HTML file for each posting, Backstage maintains the messages using a simple database and displays them using a small group of pages.
- Database queries: Every project includes a set of queries used to access the databases that project needs. If more than one project needs a query, you must duplicate the query in each project that needs it.
- User profiles: As with a Web server, you can include a list of users who have special privileges within a Backstage project. These users, whom you manage separately from the Web server, are unique to each project.

Using these four elements, each project becomes a web application that stands on its own for its own purpose, whether it is customer service, intranet departments, or a personal Web site.

Working with Projects

Manager works with HTML and related files as projects. A project is the starting point of any Web site created with Backstage, which gives you the flexibility to develop unique sites or applications simultaneously.

Each project maintains its own set of users, discussion groups, and database access and queries. Backstage Objects, inserted on pages using Designer, further support each of these features.

Manager recognizes two types of projects:

- Local: These projects are located entirely on your machine. They stay on your computer, and you can access them at any time, regardless of whether you have access to the network. Web sites typically are developed and tested here before they are deployed.
- Remote: Remote projects are located on a Web server that is not a part of your local machine. Remote projects are organized by the server where they are deployed. To connect to the remote site, you need to have FTP server software installed on your local machine.

Clicking any of the project icons within a local or remote server shows an expanded view of the project's contents in the right pane of the Manager window.

The project directories further divide into public and private parts. When you install Backstage on your computer, it creates a `Private` subdirectory parallel to your document subdirectory (such as `html` or `htdocs`) within your Web site. The `Private` subdirectory contains information about discussion groups, queries, and users. For security, this information is kept in `Private`, which you can access through the Web and FTP server only by Backstage Manager and Server. Although you can browse these files directly using other software, editing them directly can compromise their integrity and prevent the project from working properly.

Keeping Links Current

Part of the problem in maintaining a Web site is keeping the hyperlinks within the site valid and working. If you move or delete files or restructure project folders, hyperlinks within the site can suddenly point to pages that no longer exist. Maintaining these links is one of the most important aspects of site management; nothing says poor craftsmanship like lumpy duct tape and broken hyperlinks.

To help keep a handle on this problem, Manager has two built-in functions to track links and pages. The first is an automatic link update utility for files that are moved via the drag-and-drop feature.

NOTE

Dragging and dropping files from project to project and folder to folder within Backstage doesn't perform a traditional move. (The file is not cut from its old location and pasted in the new one.) Instead, it acts as a copy operation, leaving the file in the original location and adding a new copy of it to the destination.

Whenever you move a file with the drag-and-drop feature in Manager, a dialog box appears to update the hyperlinks for the file in its new location. This box looks at each relative link on the Web page that moved, ignoring links that consist of a complete URL, and updates the links to reflect the new path to their targets from the Web page's new location.

This feature is limited to one-way updating; it updates the links only within the page that moved. It does not change hyperlinks to that page from other pages because the original file is left in its original location. If you delete the original file, the links to it are no longer valid, which presents another problem: How do you find out which pages have hyperlinks affected by missing pages?

Manager includes a utility called the Link Wizard that looks at every hyperlink in the Web site to confirm its validity. You then have the opportunity to provide new address information for each hyperlink.

At your choosing, Backstage Manager can automatically fix broken links. The only way an automatic fix occurs is if it finds only one file on the Web site that matches the name of the missing link. The spelling is case-sensitive and must match exactly. The other option gives you the choice of specifying a new file for every broken link.

If you move a large number of files using a file manager or change your site in other substantive ways, the Link Wizard helps you keep a handle on maintaining the integrity of hyperlinks within your Web site. If you use the Link Wizard regularly, the result is a user-friendly site that doesn't leave people frustrated by the message `Error 404: File Not Found.`

Backstage Designer

Backstage Designer, the HTML editor of the Backstage suite, provides a graphical interface so that you can see what your page will look like on the browser with a set of powerful tools with which to use HTML tags in the HTML 2.0 specification. (See Figure 36.13.) This spec includes images, links, tables, forms, and buttons, in addition to newer items from HTML 3.2 and custom extensions, such as Java, ActiveX, and Netscape Plug-ins.

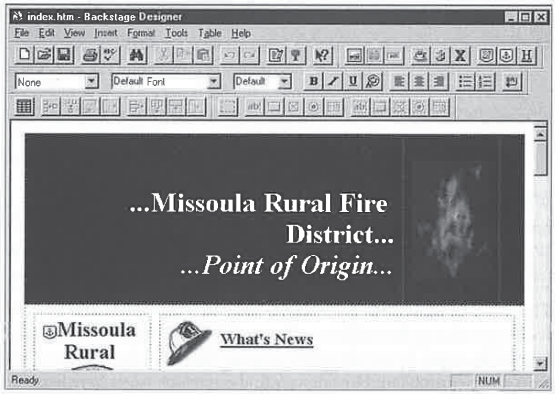


FIGURE 36.13.
Backstage Designer is an HTML editor that displays your pages as they're likely to appear on a browser, rather than as a list of tags and attributes.

NOTE

Backstage does not yet support frame tags, one of the newer items on the list of HTML tags.

Some pages use tags in slightly different ways that Backstage does not entirely understand. For example, many editors don't close the new paragraph tag, `<P>`, with a closing tag, `</P>`. Backstage reports formatting errors on these pages and adds or removes tags as it sees necessary.

These inconsistencies between the way different editors format their pages are an inconvenience at times, but Backstage can still open the pages for editing.

The Backstage Designer toolbars and graphical editing area provide access to virtually every HTML feature available. You also have access at any time to the HTML source code for adding advanced items, such as JavaScript and browser-specific extensions.

Designer's Text Capabilities

Not so long ago, your choice of text formatting was limited to whatever the browser used for its default. With the evolution of HTML, it is now possible to define a wide range of custom attributes using the `<FONT>` tag. This tag gives you a lot of flexibility in controlling exactly what font face you want to use, as well as its size. Unfortunately, learning all of the nuances for the tag's attributes can be a time-consuming process.

With Backstage Designer, it's all a matter of point and click. Highlight the text you want to change, click the palette button, and select a color from the list.

Other attributes are as simple to assign using point and click. You can access headings, fonts, font styles, justification, numbered lists, and unordered lists from the toolbars, in addition to special-use tags such as citation, definition, emphasis, and strikeout.

Working with Images and Image Maps

Graphics are now an integral part of a Web page, whether they're simple bullets, company logos, navigation bars, or complex image maps.

Designer supports GIF and JPEG images in all of these roles through its point-and-click interface. Designer controls all the basic features of a graphic through a properties box. These features include image alignment, borders, horizontal and vertical spacing, and scaling.

Designer also provides an editing screen for creating an image map. An image map is a graphic with hyperlinks embedded in various areas. It offers a great opportunity for creativity and functionality when you can create multiple links within an image.

Images and image maps are covered in more detail in Chapter 12, "Adding Images to Your Web Page."

Adding Specialized Content

One of the fastest growing areas of Web page development is specialized content, including real-time sound, animation, virtual reality, and other non-HTML items.

NOTE

For the latest on the W3C proposals and developments on working with objects embedded in HTML, check out the W3C Web site for ActiveX and Java at `www.w3.org/pub/WWW/OOP/`. For the latest on real-time audio and video content, see `www.w3.org/pub/WWW/Consortium/Prospectus/RealTime.html`.

Backstage Designer directly supports the inclusion of Java applets, ActiveX controls, and Netscape Plug-ins.

To include a Java applet, the class and related files, such as images and audio, should reside somewhere in the root web. After specifying the location of the class, add additional parameters through a tab in the dialog box. (See Figure 36.14.) At the time of this writing, Netscape Navigator 2.0 and later, Microsoft Internet Explorer 3.0, and Sun HotJava support Java applets. Java applet support is included on a "wish list" for NCSA Mosaic, but no timetable is available for when it might be implemented.

Microsoft developed ActiveX controls to provide advanced object linking and embedding control for content from other applications, such as Excel and Word. Like Java applets, the various aspects of ActiveX controls are controlled through a dialog box.

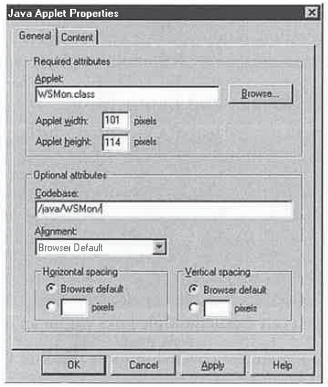


FIGURE 36.14.
The Java Applet Properties dialog box provides an interface to control the attributes and parameters, including an image to display if the browser is not Java-compatible.

In addition to support for embedding traditional applications such as word processing and spreadsheets, ActiveX controls are available for linking Netscape Navigator Plug-ins with Microsoft Internet Explorer. Functionally, ActiveX controls extend the capability of the browser the same way as a Netscape Plug-in, in addition to providing a way to include standard Windows interface items (buttons, window layouts, and menus) within the Web page. Controls are loaded into the browser automatically when needed and don't require a separate installation process.

Netscape Plug-ins provide inline support for third-party add-on content such as VRML, RealAudio, and animation. Without Netscape Plug-ins, to view content not directly supported by the browser, you must launch an external application. When you use a plug-in, the content is included inline with the rest of the document without the need for switching to an external application.

Going Beyond HTML

For features that Backstage toolbars or menus don't support, such as frames, Backstage Designer includes an HTML source-edit mode (shown in Figure 36.15) to view or change the source code for the page. When you save the changes, the modifications are displayed in the editing window. If you put in HTML tags that Backstage doesn't recognize, the style does not appear, but the text still appears.

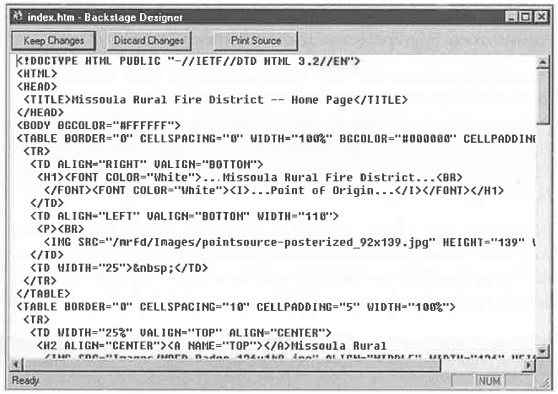


FIGURE 36.15.
The HTML source edit window enables the page designer to view or change the base code, including adding unsupported items such as JavaScript or VBScript.

Designer also goes one step further to ensure correct HTML code by performing HTML validation. If it detects a formatting tag that was not closed or an incompatible nesting of tags, it corrects the errors and alerts you to the inconsistencies.

Because of its capability to edit the source code, Designer will support future extensions of the HTML specification without requiring an immediate upgrade to a new version of Designer, although these extensions won't display on the editing screen as they were intended to appear.

The script editor, related to the source-edit window, includes JavaScript and VBScript code for the page. This free-form text window encases its contents in `<SCRIPT>` tags on the page. Although scripts are invisible to the user, they appear within Designer as a small yellow icon. Double-clicking the icon enables you to edit the code.

TIP

Even though Designer directly supports editing scripts, you still need a compatible browser to test the script's operation. Designer doesn't include any provision for interpreting and executing content within `<SCRIPT>` tags.

Backstage Server

If you want the full power of Backstage Objects, you'll need the Backstage Server to work in conjunction with your existing Web server. The Backstage Server is the key to providing dynamic Web pages and content, including database access and discussion groups.

It is also one of Backstage's most confusing features. It is *not* a Web server. It's an add-on that enables an existing Web server to provide the content that Backstage objects generate.

The Backstage Server does not replace an existing Web server; it only provides functionality for the Backstage objects and database features. The Backstage Server services the object library. Backstage objects are small applications that generate HTML content at runtime (another way of saying "dynamic page creation"). These objects include simple functions such as the current date and page hit counter, and advanced features such as posting information to a database or generating custom reports.

Backstage requires that you install 32-bit ODBC drivers for the databases you'll be using with your Backstage applications. Backstage is compatible through ODBC with database applications from Oracle, Microsoft, Sybase, Informix, CA-Ingres, and Borland. Most vendors' ODBC drivers require that you own a 32-bit version of their product before you install it.

NOTE

Microsoft developed Open Database Connectivity (ODBC) as a standard interface for databases. It uses Structured Query Language (SQL) for retrieving, updating, and posting data. Using ODBC, a single application such as Backstage can access a multitude of different databases using a single set of commands, enabling developers to build client/server applications without becoming tied to a specific database program.

You'll need to install and configure a separate Web server, such as those that Microsoft, Netscape, or O'Reilly WebSite, bundled with Studio, offer. Chapter 37 provides more information. Backstage Server works as a companion to the Web server to dynamically create pages at runtime.

Server Administrator

As information about your Web server changes, you can change the Backstage server to match. To make sure Backstage knows how to work with your server, the Administrator provides a means to enter information about the server root directories, logging requirements, and CGI virtual path mapping.

The Administrator provides no direct control over the Web server; you must use the Web server's configuration programs to control it. You will find details on configuring the Backstage Server and a variety of Web servers in Chapter 12.

Basic Requirements and Configuration Considerations

Backstage Server runs on Windows NT and Windows 95. Its memory requirements depend on which Windows system you use—24MB for NT or 16MB for Windows 95.

Designer and Manager also run on Windows 3.1. The minimum processor required is a 486 processor running at 33MHz. The memory requirements for these two programs are the same as for the Backstage Server. If you run Designer under Windows 3.1, you will need 8MB of RAM.

TIP

As with many applications, Backstage's performance will improve with additional memory. You can never have too much memory.

You must have an HTTP-compliant Web server on the same machine as Backstage Server. Several products are available, including offerings from Netscape, Microsoft, NCSA, and O'Reilly and Associates.

A Web browser is also good to have on hand. If you include Java applets on your pages, your options are limited to Netscape Navigator (2.0 or later), Microsoft Internet Explorer (3.0 or later), or Sun HotJava. ActiveX controls and Netscape Plug-ins will work with both Internet Explorer and Navigator. You'll need a special ActiveX control from Microsoft to include Netscape Plug-ins in Internet Explorer, however, and a special Netscape Plug-in to use ActiveX controls in Navigator. Nobody ever said the Web was always going to be a simple place.

To work with remote projects, you must have an FTP server on the machine with the Web server. Backstage Manager includes built-in FTP client software but not a server.

Bonus Add-ons

Backstage also includes a couple of items to increase its value as a turnkey solution to creating a Web site. As I mentioned earlier, the Backstage Server is designed to support an existing Web server. If you don't already have a Web server installed, Backstage includes a full-featured version of O'Reilly WebSite.

To aid in creating Web pages, Backstage also includes a large set of clip art and templates to speed the design process.

NetObjects Fusion

NetObjects Fusion, another tool in the growing field of integrated Web tools, features a site and HTML editor and graphics manager. It emphasizes site structure and style and offers tools with which you can control and implement sitewide elements such as page borders, styles, banners, navigation bars, and hyperlinks. Editing one of these site features automatically updates the entire site.

Fusion is a site-oriented tool that works with your Web pages as a whole, rather than as a collection of HTML pages that happen to link to each other. In the site context, Fusion generates a map of the entire site structure and allows global changes from a centralized location. This approach encourages top-down design and planning, which results in sites that are more intuitive and easier to browse.

NetObjects Fusion includes a graphical editor, database publishing, and asset management.

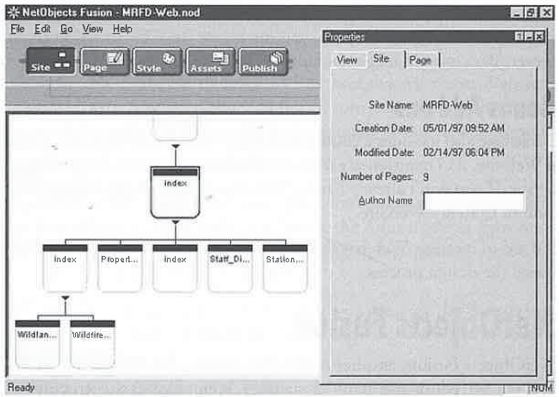
Although NetObjects Fusion is listed as a tool for users with a broad range of experience levels, it is primarily geared toward people with previous Web and graphic design experience. It requires Windows 95 or NT with at least 20MB of hard drive space and 16MB of RAM to run.

Rather than breaking down Fusion into a series of programs, I'll look at it in the variety of views it offers for your site and pages. Fusion offers five different views, each of which serves a different need.

Site View

You use the Site view to create a hierarchical structure of Web pages or import an existing site (as shown in Figure 36.16). You can choose from one of Fusion's predefined formats or create your own.

FIGURE 36.16.
The Site view displays a hierarchy of your Web pages with parent, child, and sibling relationships that appropriately places links in each document.

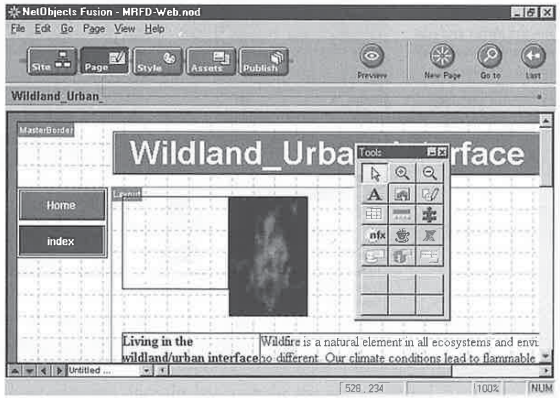


With the visual interface, you can change or revise the structure as needed to fit the needs of your site. If you're working from an existing site, NetObjects Fusion also includes an option to import the pages into its proprietary format. The conversion process has several holes but still enables you use Fusion's powerful site management features and layout tools.

Page View

With the Page view, you can create layouts and add or edit content to Web pages. (See Figure 36.17.) One of the most intuitive editors on the market today, it gives the developer precise control over a page's appearance and maintains a graphical interface with drag-and-drop capability.

FIGURE 36.17.
Fusion's use of HTML tables and other automatic formatting features precisely controls each page. A Properties palette controls the overall view of the page and its specific attributes.



You perform most editing by dragging text, images, applets, or other content to the desired location on the page. Fusion automatically creates a section of HTML table code to ensure that the content appears correctly in a browser. Grids, guides, and snap-to features also aid in aligning different elements.

The drag-and-drop interface extends to interactive content. With Fusion, you can deposit Java applets, ActiveX controls, Netscape Plug-ins, and sound and video files. The program ships with six Java applets, called NetObjects Fusion Components, that fill needs such as tickers, site-mapping, and discussion groups.

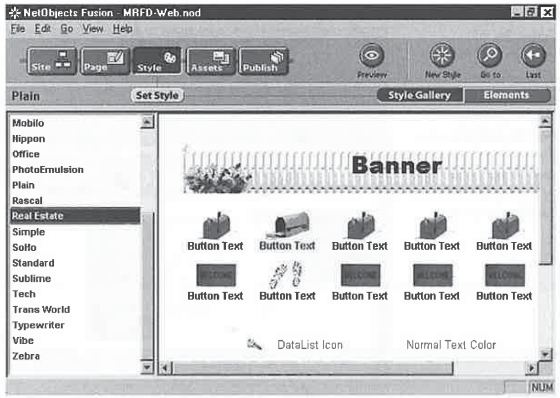
NetObjects Fusion supports all basic HTML plus extended HTML and scripts for additional features. Its method for dealing with hyperlinks is both creative and practical. Smart Links defines hyperlinks by their relation to the destination. You can point links at a page's parent, child, or other relative, and Fusion will construct the appropriate link when the site is published. This enables you to rearrange files at will without worrying about redefining the links within it.

You gain database access through ODBC data-object controls, which fill in a table with information from a database table.

Style View

The Style view is the access point for 50 prebuilt site styles that give a consistent look to your project's pages. (See Figure 36.18.) Style features that you can control from this view include color schemes and navigation icons.

FIGURE 36.18.
With the Style view, you can define and edit the graphical standards that define your site.



Several predefined site styles that you can override according to your own needs are included with Fusion. Each style is based according to a theme and provides a centralized method of managing and applying the overall graphical appearance of your site.

The predefined components available with each style from the Fusion Site Gallery include backgrounds, banners, primary and secondary navigation buttons, fonts, lines, icons, and text colors. You can also create new styles from scratch, if needed.

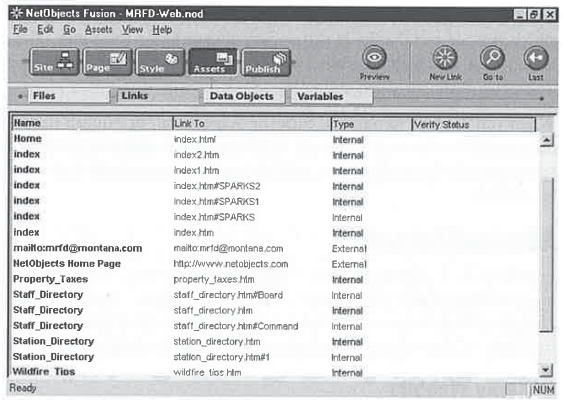
Assets View

With the Assets View, you can track various files, links, and database information, including image files, applets, objects, and plug-ins. (See Figure 36.19.)

The best part about using the Assets view to change files is how it integrates with the rest of the program. For example, if you change the name of a GIF file, the Assets view also automatically changes any references to that file anywhere else in the site to the new value. It treats links to external resources in the same fashion and lists them with the rest of the other site assets.

The Assets view marks assets that are included as part of a site but not used within any pages, enabling you to easily identify these space-wasters and delete them, if appropriate. It also identifies broken links and missing files for appropriate action.

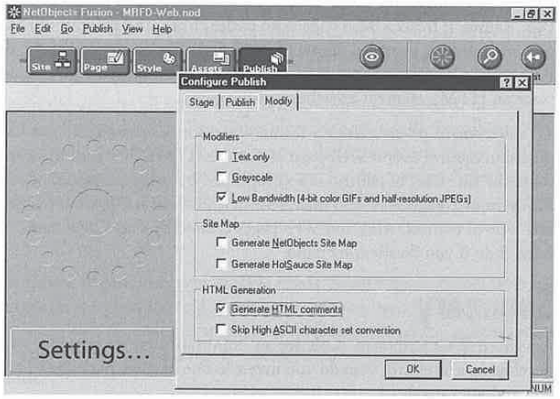
FIGURE 36.19.
Whether it's a file, hyperlink, or database connection, the Asset view lists it for you to review or change.



Publish View

After all the work is completed on the site, use the Publish view to send the finished product to its final location. (See Figure 36.20.) You can publish a site in its original or text-only form. To decrease upload time, NetObjects Fusion gives you the choice of publishing the entire site or only the changes that have occurred since the last time the site was published.

FIGURE 36.20.
The Publish view configures the final settings for the site and sends it to the server for publication and use.



You have two options for configuring a site for publishing. First is a local staging server, with which you can test the site in the privacy of your home computer. The other is a remote Web server, with which the work is available for use by the rest of the world, your company, or whomever is the intended audience.

When you're ready to publish, click the Stage (local) or Publish (remote) buttons to compile the site and its elements into a set of HTML pages and associated files and send them to the server you designated. Other publishing options include text-only, grayscale, and low-bandwidth.

Advanced Features

Functionally, Fusion gives you all the features you expect from a Web design suite. But it also has a number of advanced features, which aren't always inherently obvious to the casual user. Not all of these features are available in other Web development suites. Consequently, if you find some of these features particularly powerful, you might want to consider getting Fusion.

O'Reilly WebSite

O'Reilly WebSite is a full-featured Web server that works hand-in-hand with the Backstage Server to provide the support that you need to run a basic Web site and the additional capacity and power to fully implement Backstage Server capabilities. It also includes several add-on applications to provide other services for your users, including the capability to index and search your entire Web site.

PageDraw

PageDraw is unlike most other Web page editors because it doesn't simply provide access to tags. Rather, it treats a Web page as an entire entity, and you put HTML elements on it. This type of approach provides unprecedented pixel-level control over the display of a Web page. Rather than having to concern yourself with what the user's browser does, you can explicitly place an HTML element anywhere on the page.

All this control doesn't come without some sort of drawback, and the drawback of getting precision control over a Web page is that the HTML files tend to be much larger. This is because the fine level of control is accomplished by using a number of advanced HTML tricks. Furthermore, PageDraw also makes use of numerous transparent images to help it accomplish this type of control. Also, the Web pages created by PageDraw tend to be a bit more convoluted than if you do them by hand.

Summary

Advanced Web authoring tools are an important item to consider when you're considering developing a Web site. Should you invest in one of these packages? That decision depends on you and your needs.

If you're just creating an occasional page or two for your department or company, a simple HTML editor will probably fit your needs. If you're responsible for managing the content of a folder or site, an application such as FrontPage, Backstage, or NetObjects Fusion will probably be a big help. If you're a webmaster, either for a department server on an intranet or for a Web server on the Internet, an integrated authoring tool will quickly become a necessity.

Remember that these applications don't do the management work for you. They provide a one-stop-shopping approach to the various tools that make establishing and maintaining a Web site an easier process.

FrontPage, Backstage, and NetObjects Fusion each have their strong points and weak points. FrontPage has a more intuitive interface for its management program (Explorer) and a more powerful editor than Backstage. Using wizards and templates to create Web sites is very helpful, especially when you're creating new Web sites from scratch. Database access requires much more knowledge of ODBC and SQL than the average user has. Its discussion group feature depends on HTML files, which can lead to problems if a message is deleted for inappropriate content.

Backstage excels in database access. It requires that you have only enough knowledge to set up an ODBC connection to a database (you can find directions in Windows help files) and a passing knowledge of SQL (provided in the manual). Designer then provides several ways to read and write information from databases as part of the HTML page. Backstage also has stronger user management features, including the capability to tailor a browsing session for each user. Manager and Designer are weaker tools in general than their counterparts in FrontPage and are not as intuitive to use. Designer also doesn't support frames.

Fusion takes the approach of creating integrated Web site management from that of the Web author. As a result, its features tend to focus more on the needs of the typical Web author. Web authors are concerned about the overall look and feel of their Web sites; for that, you have the SiteStyle Manager. Web authors are concerned about links breaking unexpectedly; for that, Fusion automatically updates all links. Some Web authors are interested in very precise placement of HTML elements; for that, PageDraw gives authors just that capability. The weaknesses of Fusion tend to lie in anything that isn't particularly important to the Web author. For example, few of the user and group security features that exist within FrontPage are included in Fusion. Also, Fusion does a lot of its automaintenance by keeping a private database of objects. If something happens to that database, you might lose a significant portion of your Web site.

Which application you choose depends on your specific needs. FrontPage is considerably less expensive than Backstage and is a good starting point for people just assuming Web management responsibilities. If your Web site requires any level of database interaction, such as online ordering, membership management, scheduling, or customer service, Backstage offers the best in features and ease of use.

CHAPTER 37

Web Servers

by J. Gregory Bryan

IN THIS CHAPTER

- Present Purpose *761*
- Future Needs *765*
- Cost *766*
- Now, Let's Talk about Servers *767*
- Moving on to the Web Server Software *771*
- Features of Popular Web Servers *772*

Some things never change, which can be both a comfort and a curse when it comes to purchasing a new or improved computer system. In this chapter on Web servers, some of you may be comforted by the fact that in purchasing or upgrading a server system, you will be guided primarily by a solitary, yet very important question. It's one that should be comforting because you've heard it so many times before: *"What are you going to use it for?"*

Remember the first computer you ever bought? What did the salesperson ask you? *"What are you going to use it for?"* Right? How about the last equipment purchase you ran past your boss? Same thing? Even though it can be a little aggravating at times to have to explain the subtle complexities that led you to your triumphant conclusion that you must upgrade, it's still somewhat comforting to realize that—whether you're preparing to purchase a new Web server, a word processor, a microwave oven, or even a fancy fishing rod—you always have to ask yourself, *"What do I really need to get this job done?"*

It's especially true for something as complicated and potentially expensive as an in-house Web server system. And that brings us to why this question can also be a curse, or at least why it might make you curse before the decision-making process is over.

If you are already an experienced HTML programmer, a systems analyst, or some other type of Web developer or site administrator, you already know how fast the Internet industry is changing. Or should I have said *industries*? There are, and continue to be everyday, an increasing number of highly specialized hardware and software industries that are exploding into existence as the popularity of the Internet and the World Wide Web increases. At the very forefront of this tidal wave of technology is the humble Web server and everything associated with it.

In this chapter, I operate under two assumptions:

1. That you're already experienced enough with the Internet or intranets to know the value of broad scale, distributed computing
2. That your decision to purchase a new Web server or upgrade an existing one is for the general purpose of better exploiting the treasures of the Web

If you're nodding your head in the affirmative and thinking "yeah, that sounds about right," let me be the first to tell you—it's not going to be as simple as you think.

If, on the other hand, you already know how complex this decision can be, I'm sure you'll agree that there are several key factors on which you *must* be clear before making your decision. The first half of the chapter centers on these important considerations. Then, in the last half of the chapter, I show you the broad landscape of available options and focus on a variety of examples from several different product categories.

Okay. Let's get started.

Consider again the underlying question, *"What are you going to use this for?"* I must caution you not to answer too quickly. Obviously, if you've decided to set up a Web server, you have a primary interest in connecting to the Web. Maybe some of you have rented service at an Internet service provider (ISP) and have finally decided that to get the kind of hands-on control you need, you must have an in-house server. Some of you might have had an intranet in service for quite some time but now are ready to branch out onto the Internet for the very first time. In either case, I can virtually assure you that the things you want from your Web server today will not be the same things you'll be demanding from your server six months or a year from now.

Let's look at the big issues first and the factors that define them:

- Present purpose
- Future needs
- Cost

Present Purpose

Your present purposes for upgrading or purchasing a new Web server are probably varied and readily justifiable. They might include any of the following:

- To connect the organization to the Web
- To take advantage of the content out on the World Wide Web
- To market and promote your own content
- To expand your marketing and customer service capabilities
- To compete more effectively

Now here are some questions you'll have to answer before you can do any of that:

- What kind of in-house system or facility do you currently have? Is it dedicated to one type of platform or many?
- How many different operating systems (OS) are currently in place?
- Does the organization have a preferred software standard (such as Microsoft, Lotus, Novell, and so on)?
- Will existing intranets, local or wide, be connected through the Web server?
- How many users will access the Internet internally? How many hits are expected from outside sources?
- What about security? Do you need a firewall? Multiple firewalls?
- Will you be engaging in any sort of electronic commerce over the Internet?
- Will you be transferring files, parts of databases, or applications through the server?

- Are any of the transactions going to be of the "mission critical" type? How much fault tolerance can you afford?
- How much storage space will you require? Do you need a sophisticated RAID (Redundant Array of Inexpensive Disks) system or would mirrored servers be more appropriate?
- How are you going to communicate with the Internet—by modem, ISDN (integrated services digital network), T1s, T3s, frame relay?
- How knowledgeable is your in-house staff in *all* aspects of network computing and telecommunications protocols?
- Will you need a single-, double-, or triple-tiered system to adequately meet your needs?
- What percentage of your IS budget is devoted to Internet-related activity? How many personnel will be assigned to manage the system full-time?
- How well does your CEO really understand what you're trying to do?

Those are just a few questions to get you thinking. Chances are good that no matter what your level of sophistication is as a Web development professional, you never have had to deal with so many different issues surrounding one decision before. If you make your Web server decision in a responsible fashion, however, all these questions and more will come to bear on your selection.

You'll notice that I haven't spent any time at all talking about communication and transaction protocols (TCP/IP, CGI, FTP, HTTP, SMTP, NNTP, MOM, ORB, and so on). For one thing, the acronyms are piling up faster than anyone could hope to list them. Beside that, in spite of the plethora of protocols, a growing effort to standardize the most basic elements of internetwork communications has occurred in the last few years.

Practically every server you might possibly consider will accommodate an increasingly standardized set of communications protocols. There are a few remaining battles, to be sure, such as that being fought by the Object Request Brokers (ORBs) and the older but still entrenched Remote Procedure Calls made to the Message Oriented Middleware (MOM); but even the most aggressive and often prejudiced software designers are working hard to make sure these little skirmishes don't injure the would-be customer. Usually a variety of software patches are at hand for anyone who is inadvertently bruised in such conflicts.

Having said all that, however, it does behoove a Web server shopper to pay attention to the *kinds* of functions the various server applications can perform. An almost unimaginable variety of Web server vendors and Web server products is in the marketplace today. The prices vary from free to many thousands of dollars. Most of them do very similar things. None of them do everything.

Later on, I'll look at the features of some of the most popular Web servers. For now, let's think a little more about the questions at hand. Back in the early days of Web servers and the Internet (I'm talking less than a half decade ago), a would-be Web surfer had to have a different tool for

every task. After the invention of the HyperText Markup Language, HTML, and the first GUI browser, MOSAIC, users quickly learned that by following the "Blue Text," they could go just about anywhere and find just about anything they wanted. Getting it, however, required very specialized tools and, frequently, completely different servers.

Here are just a few of the more common types of the original, independent servers:

- Gopher servers transport files of various types, which was fine as long as you had the proper application software to read them.
- File Transport Protocol (FTP) servers enable you to do basically the same thing as Gopher servers but with much greater flexibility.
- Network News Transport Protocol (NNTP) servers deliver Usenet or newsgroup messages and articles.
- Simple Mail Transport Protocol servers (SMTP) deliver electronic mail.
- Telnet servers are required to conduct remote operations on distant computer terminals.
- HTTP servers enable users to talk to one another.
- DNS servers keep the growing number of domain names organized.
- DBMS enables users to sample, calculate, or summarize from virtually any database located anywhere in the world.

I don't know how many different kinds of servers exist. I still remember when the newfangled print server was the greatest thing since sliced bread. The point here is that just couple of years ago, setting up a Web server that had even the slightest variety of functionality would have been beyond the means of most IS or DP departments, but that is not the case anymore. Most popular Web servers now feature a collection of the common server functions. Many even include sophisticated development tools, a variety of security options, and highly adaptable browsers. Again though, quite a lot of disparity still exists between the features of even the most popular Web server products. Even among those that offer exactly the same kinds of features, one can find significant differences in hardware requirements, maintenance requirements, ease of setup, administrative options, and scalability. It pays to be very clear about your present needs and the needs of the others who will ultimately visit your Web server.

Assuming that you wisely choose from your selection of desired server functions, you still must get a few other critical issues right. It goes well beyond the scope of this chapter to try to discuss them all. These are some of the more important factors to consider, however, regardless of your intended application:

- **Compatibility:** It will be particularly important for you to select a server that is compatible with your lead operating system (OS) or network operating system (NOS). Although you might at first intend for the Web server to run on a completely independent platform, that probably won't be the case for very long. You might need to interface with a mainframe to have access to existing files or databases. You might

decide to connect the LANs in accounting, shipping, marketing, customer service, and credit to the Web server. If you plan to have a Web site with an artistic flare, you'll want to connect the people in the graphics department. They might have Macs and you might not.

It's clear to see that as the popularity and functionality of an in-house Web server grows, the compatibility issue will become more and more important. Equally to the point is the fact that by selecting compatible server software, you will avoid any unnecessary staffing or training problems. If you're in a house of many Windows, for example, you could be in trouble if you take a fancy to a UNIX-based Web server.

- **Platform requirements:** For many of the same reasons I previously stated, it is important that you select a Web server that will run either on an existing platform within your organization or at least on a platform that will be compatible and suitable for present and future needs. You must consider these questions: How many users are expected to transact with the server? Will it support a sufficient number of IP addresses? Will there be enough RAM? Can you add more? What sort of disk array is required for your purposes? How many CPUs will be required? Is that expandable? What about connectivity issues? Networking, telephony, and so on? The list goes on.

  Related and equally important to these questions is the question of fault tolerance. How well-integrated is your hardware and the required operating system? Is it extremely stable and capable of handling faults of all types without locking up or shutting down? If you have mission-critical or secure commerce requirements, will you need multiple backup systems to meet your Web server's needs?
- **Scalability:** The demands on your system will grow. It may be difficult to imagine in the beginning. And you might have other overarching considerations, such as a boss who is encouraging you to "just get the basics to get us started." In either case, however, you will be sorry if you don't select a server and platform combination that is at least 50 to 100 percent more capable than you initially require. Think about it. There's no room to elaborate here, but it's the same principle that was at work back when you thought it impossible to fill up that gigantic 20MB (!) hard drive. Enough said?

Of course, numerous other considerations are related to the hardware/software issue. I hope, though, that the notes and questions I gave you will stimulate sufficient discussion among the people in your organization so that you can avoid unnecessary pitfalls.

Speaking of discussions among people, it is likely that the move to an in-house Web server will require, at some point, a sales job on the executives who control the purse strings. It might have been a relatively easy sell, convincing the president of the company that you needed to put up a Web page when virtually all of his/her competition is represented on the World Wide Web. It will, no doubt, be a harder sell to get the *many* additional thousands required to build and properly maintain an in-house Web server.

In the recent article "Do Execs Get the Web?" in the November 1996 issue of *Internet World*, Joel Maloff touched on a number of very important topics concerning your impending sessions with upper management. His basic message is that, even today, most executives still don't "get the Web." As incredible as it may seem, most chief executives still think this Internet thing is some sort of fad that will soon go away. Those few who have seen the Web's potential, Maloff points out, are scared stiff. They don't understand it. They're afraid to get on it. They're afraid to stay off!

These are generalizations, of course. No doubt, a growing population of savvy executives out there have jumped on the wagon and are hanging on for dear life. The majority, though, must certainly fall into the categories described in Maloff's article. More than likely, your boss is in one of those categories as well.

Why have I chosen to dwell on this point? It's a topic you should be prepared to discuss before you endeavor to fling open your company's doors to the rest of the world.

"Flinging open the doors!" Indeed, that's the way most upper managers will see it. All the nightmares about computer hackers, industrial espionage, drug trafficking, child pornography, and whatever other unsavory evils one might imagine will be swirling through your boss's head as soon as you start talking about putting up an in-house Web server. Don't laugh! The boss has a good point and one to which you'd better be prepared to respond.

Future Needs

You know there will be some future needs (and benefits), but have you written them down? Have you discussed the fact that company-wide e-mail will now be possible, whereas you might not have had it before? Does your customer service department really understand the fact that it will be able to double or triple the number of service inquiries to which it can now respond, and that with the use of autoresponders and FAQs, it can now provide 24-hour, worldwide service at virtually no extra cost? Does the marketing department appreciate the opportunity it will now have to study site usage reports and gain insight into prospective customers who think in ways never imagined before? Does the sales manager know she can now conduct interactive sales meetings worldwide or offer online tutorials while, at the same time, virtually eliminating the normal travel costs? Use your imagination.

To discuss the future needs and benefits issue, as well as the many other questions you will be asked to address, it makes sense to pre-plan your presentation and do so with the aid of professionals from other departments, such as Marketing, Customer Service, Shipping, and Accounting. Spend as much time as is required to educate these colleagues regarding the present and future benefits of the desired system. Spend a lot of time explaining security issues and the expected increases in teamwork and efficiency, especially when internal intranets will be connected with the Web server.

In other words, prepare a business plan surrounding your intended purchase. If you present a well-organized and professional plan that has the support of your company's key departments, you'll probably be surprised how agreeable upper management can be. Most executives have no trouble whatsoever investing in things that they feel will grow the business. *All* executives, on the other hand, will balk at spending a nickel on anything they don't understand or have doubts about. And speaking of spending, that brings us to the next topic: cost.

Cost

Keep the following in mind. Regardless of whether you intend to use freeware on an existing platform or intend to string wire from here to Timbuktu and set up your own in-house ISP, you are going to spend some money. How much, of course, depends on the answer to that very first question I asked: *"What are you going to use it for?"*

It's only after answering this question as honestly as possible that you will be able to determine your costs. The dollars, from low-end to high-end, can easily range from $5,000 to $100,000. Those who intend to have a *big time* operation with multitiered systems, proxies, routers, high-end security, fully integrated enterprise, legacy, and commercial applications will spend hundreds of thousands. It all depends on your ultimate objective. If that objective is well-defined and understood, you can speak confidently about the numbers, knowing for certain well the benefits will greatly outweigh the costs.

Remember also that you must include the cost of personnel in your overall figures. One frequently made mistake is the assumption on the part of first-time site managers that they will be able to handle the whole operation by themselves, including site design, administration, maintenance and repairs, content preparation, content updates, programming, interdepartmental liaison duties, budgeting, purchasing, settling the inevitable squabbles and disputes, and so on.

Regardless of the kind of organization or type of operation that you intend to run, you will likely need the full-time assistance of several other people. Believe me, many more capable heroes have tried to go it alone. Interestingly, some of the best and most innovative developments in server technology have been made by super technicians who finally decided they needed to eat or go home and sleep! One person can't do it all. Don't sell your new Web server operation short by assuming that you can.

In your case, even if the boss is loathe to add additional staff, it's better to be honest about your staffing requirements rather than underestimate the venture and end up in abysmal failure. Besides, business executives hate being "nickeled and dimed" over things you should have warned them about in the first place. Don't get discouraged on this point. Being fiscally conservative is what good executives do best. That's their job and it's what they are expected to do. Do your job and prepare a good and thorough presentation. Speak in plain English. Address all the points. You'll get your new Web server—and maybe the Employee of the Year Award!

Now, Let's Talk about Servers

As I prepared to write this chapter, purely by chance one of my colleagues was in the process of upgrading an important in-house server. In his case, it was a packing server that keeps track of all the orders being packed and shipped out of a busy warehouse. Although his particular server requirement was fairly limited in application but mission-critical in nature, I watched him go through a very similar process of defining his needs. The platform/server combination he chose was a highly compatible, fault-tolerant, and expandable system that will not only meet the present needs of the shipping department but will also provide the initial infrastructure for much more widespread server applications. The company, by the way, though large and very successful, has not yet reached the point of establishing an in-house Web server. It's still in the process of laying the appropriate groundwork.

It occurred to me while talking with my friend that probably many organizations are in a similar state of infrastructural development and that many such organizations might not even be as well versed in server technology as he is. These people (and perhaps readers of this book) might find themselves in the position of having to gather as much information as possible in a very short time to assist them with their Web server selection. In an industry that is changing as rapidly as this one is, you can never hope to know all there is to know. Using the Internet, however, the broad range of readily available magazines and newsletters, product brochures (which are just as easily acquired), and personal interviews with knowledgeable people—such as I have done to gather background information for this chapter—a person can educate himself very quickly.

A NOTE TO FIRST-TIME WEB SITE DEVELOPERS

Although most of the information in this chapter has been slanted toward the experienced Web developer, many of the considerations required of people establishing in-house Web servers will also apply to the first-time developer. One additional consideration, however, is especially important for those who have never established or maintained a Web site.

First-timers should strongly consider the option of having their site hosted on a remote server operated and managed by a reputable Internet Service Provider (ISP) company. The main reasons for this suggestion are very important:

- The cost of having a site hosted by an ISP is considerably less than building and maintaining an in-house system. Delaying those costs makes good sense for anyone who is still low on the learning curve. Additionally, the use of ISP servers at this stage of Web-presence gives the beginning Web developer an opportunity to test and compare different kinds of services and different kinds of server environments. This will be very useful information when the time comes to establish your own system.

continues

continued

- The effective management and administration of a Web site requires knowledge and skills that cannot be gained without some experience. When your site is hosted by a professional service provider, you can begin developing your Web presence with the expert guidance from those who have performed the task many times before. A good ISP will make suggestions about site design and, perhaps even more importantly, will provide you with a wide range of administrative reports. Such reports might include the following:
  - Usage statistics to show the number of hits your site experienced, the frequency of repeat visits, the amount of time spent on various pages, and the addresses of visitors
  - Error logs
  - Message logs, e-mail, and other items of use and interest

The study of administrative reports provided by the service provider yields valuable information regarding the function and effectiveness of your Web site.

In other words, learn all you can from the experience of others before you endeavor to set up your own internal Web server. Connecting your organization to the Internet is obviously an important venture. Mistakes avoided in the early development stage can save loads of aggravation and money down the road.

In the descriptions that follow, I will not attempt to cover all the possible platform/server combinations. That would be a multibook-scale project all its own. The information presented, however, is what I have gleaned from the variety of media sources I previously mentioned. These sources do an amazingly good job (collectively) of keeping abreast of new developments. For example, more than a hundred different magazines are devoted to computer technology and provide regular features on server- and Web-related technologies. Most of these magazines are staffed by good writers and editors who compete on a daily basis to bring you the most up-to-date information possible.

Likewise, many online magazines, user forums, and fact pages now tap into a broad range of Web-related topics in exceeding detail. Reading these materials is an educational experience for just about anyone. Another interesting outlet of information, also available across the Web, are those pages posted by site administrators and Webmasters, entitled "About this Server." You might be amazed by the variety of platform/server combinations that hundreds of other organizations just like yours have successfully deployed. Fire up your favorite search engine and have a look!

So, how many platform/server combinations are there? Probably nobody knows. There are some good, up-to-date sources, however, on the most popular combinations. I should note here that when "platform" is used, it really means the "platform environment"—the hardware and operating system combination on which the Web server will reside. Five minutes of research will

reveal that hardware being used in server environments includes virtually every kind of machine still in use today—everything from DOS machines to Windows PCs, Macs, NT servers, Sun, Apollo and SGI workstations, IBM, VAX, DEC and HP minis and mainframes, as well as a hundred other types and brands not mentioned here. Basically any kind of computer with sufficient disk space, memory, and processing power can serve as a node on the Web. Your choice must obviously be based on your ultimate objectives for the Web server.

How about operating systems? Naturally, far fewer types of operating systems exist but, still, most of them are used. Using an often-cited survey from the WebCrawler, Steven J. Vaughan-Nichols, in his article entitled "At Your Service: The Right OS for Your Web Server" (*NetGuide*, 4/97), discusses the strengths and limitations of the most commonly preferred operating systems. Vaughan-Nichols' choice for high-end, high-volume sites is unequivocal: It's UNIX. For medium- to low-end sites, he prefers Windows NT. Looking at the WebCrawler survey, it appears that the rest of the world agrees with him.

Regenerated and generalized after the original data, the tally currently stands at

UNIX:	84%
Windows NT:	7%
MacOS:	5%
Windows 95, 3.1:	2%
All others:	2%

The percentages shown here represent the distribution of the most commonly used operating system platforms for servers across the Web.

UNIX

Most Internet aficionados probably could have made a good guess at the breakdown. UNIX far and away out-paces the rest of the pack for both historical and currently practical reasons. First of all, the Internet was born from UNIX-based parents, and for the first several years of its life, the only friends and family the youthful Internet knew were also of the UNIX persuasion. As years went by, the occasional odd operation system attempted to make its acquaintance, but even today, the vast majority of Internet-based activity takes place between UNIX machines.

UNIX (or UNIX derivatives) operate on a much wider variety of platforms than any other operating system. In fact, it is unlikely you could find a hardware platform in existence for which a UNIX-type OS hasn't been developed. Consequently, the greatest variety of Web server software has also been developed to run in the UNIX environment.

In addition to its sheer ubiquity, UNIX allows the greatest number of CPUs, the greatest number of IP addresses, and by far the most robust operating system, even in the *most* heavily trafficked networks. There is a downside, though.

The list of percentages shown in the previous section will probably begin to shift as "regular" people (non-computer types) become more and more attracted to the World Wide Web. The downside for UNIX is that in spite of its undisputed capabilities, it remains one of the *most* difficult operating systems to learn. You have to want to learn UNIX—really want to learn. With Windows or the MacOS, on the other hand, you hardly have to learn at all. You just sit down, point, click, and go to work. On top of this, UNIX-based Web servers traditionally have been difficult to install. Although this is changing, Windows-type servers, by comparison, almost install themselves.

Windows

The strength of the Windows-based operating systems as Web server platforms lies in the sea of nontechnical users who are just now becoming acquainted with the Internet. If you are installing a Web server for a low- to medium-impact site and expect to interface with other people in your organization who normally use PCs for other types of applications, chances are good that a Microsoft operating system will be your best choice.

According to the Vaughn-Nichols article, though a surprising 19 Web servers have been developed for the Windows 95 system, most people who are experienced with Windows 95 in heavy-duty application software would agree that this popular operating system is probably not stable enough to support a good Web server site. Windows NT, however, has proven itself to be an extremely hardy contender and appears to be gaining in widespread popularity.

Windows NT not only enjoys a rapidly growing number of suitors in the server development community but far outstrips other OS types in its capability to run the most popular suites of business software. Also a powerhouse in the area of easy database management, NT's only apparent drawback is its scalability. No weakling in this area, it still falls short of the scalable capabilities of UNIX and IBM's OS/2. For sites expecting huge amounts of traffic, these others might be more appropriate.

NOTE

Here's an interesting note: The busiest site on the Web is `Microsoft.com`, which is powered by an NT-based server. Microsoft's database server, however, which is responsible for transferring millions of product-related requests per day, runs on a mainframe powered by a UNIX-based server.

MacOS

The popularity of MacOS, not too far behind Windows with respect to UNIX, is a testament primarily to the number of Mac users who have discovered the Web. The MacOS has limited capability in heavy demand situations and is particularly clumsy in its interactions with

dynamic sites. Of the 20 or so servers that have been developed for MacOS, only one (StarNine) seems to enjoy much enduring popularity. Mac owners are a loyal bunch, as everybody knows. MacOS may not be the absolute best choice for a Web server OS, but you can probably expect the number of Mac servers to remain pretty constant from one year to the next.

OS/2

OS/2 is a great operating system for a Web server. It is robust, multithreaded, fast—all in all, a very stable platform. The only problem is that almost no one uses it anymore. Chalk another one up to IBM's marketing strategy. A half a dozen or so server packages have been designed for OS/2 systems. They work and will work for you if you already have an OS/2 shop. If you don't, you'll find many more software options and greater support with some other operating system.

NOTE

If you are planning to closely link your Web server to an internal intranet, you will want to read several good articles that have to do with the importance of picking the right NOS. OS/2 still makes a very strong showing in the intranet environment, a factor that might influence your Web server platform selection.

Moving on to the Web Server Software

You can find one of the most informative surveys on the Web at `http://www.netcraft.com/survey/`. The Netcraft site maintains a hotlinked directory of Web server home sites, as well as up-to-date survey information about server and platform use across the Web.

The Netcraft, April, 1997, survey results shown in Tables 37.1 and 37.2 are based upon responses from 1,002,612 sites.

Table 37.1. Top developers.

Developer	*Mar-97*	*%*	*Apr-97*	*%*
Apache	356500	42.79	429049	42.79
Microsoft	97077	11.65	154653	15.43
Netscape	108933	13.07	121870	12.16
NCSA	74446	8.94	73881	7.37

Table 37.2. Top servers.

Server	*Mar-97*	*%*	*Apr-97*	*%*
Apache	356500	42.79	429049	42.79
Microsoft-IIS	78414	9.41	131718	13.14
NCSA	74446	8.94	73881	7.37
Netscape-Communications	37656	4.52	39572	3.95
Netscape-Enterprise	28418	3.41	37853	3.78
Netscape-Commerce	35041	4.21	35129	3.5

There are other excellent sources of current server use information as well. Many magazines devote a large effort to tracking server and other Internet-related topics on a monthly basis.

You can find one of the best running summaries at Meckermedia's iWORLD site (`http://www.iworld.com`), where you can find WebCompare, ServerWatch, and the Web Servers Feature Chart. In the following section, I'll highlight the main features of some of the most popular Web servers currently at the top of all the lists. These are not the only options but merely the tip of the iceberg, so to speak. Check out the previously mentioned sites, as well as the monthly Network Features sections in magazines such as *PC Magazine* for descriptions of other Web servers and server-related products.

Features of Popular Web Servers

This section gives you an overview of the features of many Web servers that are currently in use.

Apache

The most popular Web server worldwide is Apache. Lending to its unquestioned dominance is that it was developed for the most commonly used platform, UNIX, and that it is free. It's not your ordinary freeware, however; the organization that has developed and maintained this powerful package (located at `www.apache.org`) has produced a sophisticated, easy-to-use product that is packed full of features and is constantly being improved. The Apache Development Group is an international organization of seasoned volunteers who have developed this product for noncomercial distribution in the public domain. Formed under the auspices of Apache Digital Corporation of Durango, Colorado, The Apache http project depends on donations by members and users for its continued work.

The current version is a plug-in replacement for NCSA 1.3. It is highly configurable and offers DBM-based authentication databases and content negotiation. Apache efficiently uses system memory, and you can easily upgrade it by various, optional modules. It's fast, installs easily, and best of all, includes the source code.

The only real drawback is one that doesn't usually cause people much of a problem anyway. There is no *official* technical support service for this product. That doesn't mean, however, that you can't find good help and advice. You can find a list of known bugs at the Web site, and you can find third-party support services through the Apache mailing list at `comp.infosystems.www.servers.unix`, or by commercial parties such as UK Web Ltd. or Cygnus (`http://www.cygnus.com/product/idk/apache/`). *Apache Week*, a weekly digest for Apache developers, is another good source of support information. To subscribe to the mailing list, send a message to `majordomo@apache.org` with the words "subscribe apache-announce" in the body of the message.

CERN httpd

The *CERN httpd*, also known as the *W3C httpd*, is an old standard for UNIX platforms that is in its final version with release of Version 3.0A. The W3C server is another of the well-known, public-domain programs that has been perfected by developers at MIT and CERN—the European Laboratory for Particle Physics near Geneva, Switzerland, which also happens to be the birthplace of the World Wide Web.

The CERN httpd is a full-featured, hypertext server that can serve equally well as a standalone HTTP server or as a proxy server within a firewall. A proxy server provides quicker document caching and sorting and can also provide access to the Internet from behind a firewall.

User guides, release notes, and installation and overview documents are all available from online sources (such as `www.w3c.org`), as are bug logs, utilities, and patches.

The main features of this server include the following:

- Functionality as a proxy server for HTTP, FTP, Gopher, WAIS, and news groups
- Server-side, executable scripting
- Established forms
- Clickable icon support
- A powerful index search interface
- Authorization checking and access protection
- Configurable file suffixes
- Capability of handling multiple file formats, encodings, and languages
- A number of highly flexible and configurable characteristics

Microsoft Internet Information Server (IIS)

A relative newcomer in the market place is Microsoft's Internet Information Server (HS), the commercial Web server integrated into NT Server 4.0. Though IIS's popularity accounts for only about 9.4 percent of the market (compared to Apache's 42 percent), the growing popularity of PC platforms and the growing number of PC users on the Internet seems to suggest that we'll be seeing more widespread use of this versatile Web server.

The IIS runs on the Windows 95 and NT platforms, although as I discussed earlier, NT provides much greater fault tolerance and overall stability. User feedback suggests that the overall performance of IIS is close to that of Apache in terms of reliability and overall satisfaction. For users already committed to the NT platform, the additional cost (free!) is just as compelling, because it comes bundled with the NT Server package.

Drawbacks include a lack of capability to support NFS drive security, given its dependence on Windows NT networking and security. Also, it is not easily customized for varied access by Web-specific users or groups.

Technical support is abundant, however, by way of a thorough Installation and Planning Guide, online tech support, and on-board utilities and help files.

NCSA

The next most popular server, whose popularity rivals Microsoft's IIS (at 8.9 percent of the market), is an old standard—NCSA HTTPD. Like its closely related cousin, NCSA HTTPD is also free (for noncommercial use). To access a copy on the Internet, go to `http://hoohoo.ncsa.uiuc.edu/docs/setup/Download.html`.

NCSA is another UNIX-based server that is known to be highly functional, reliable, and easily supported, but, like Apache, no commercial support is available. You can readily address most support issues through online information including complete user documentation, installation guides, and tutorials, as well as an active user discussion forum. Reports of known bugs are tracked at `httpd@ncsa.uiucc.edu`. The particular advantage to NCSA seems to be its capability to handle complex user-authentication issues at very large sites, where weekly accesses may number into the millions.

Netscape Servers

Netscape sells a wide range of server products, including Communications, Commerce, FastTrack, and Enterprise. The variety and versatility of these products helps account for the more than 12 percent of the market that Netscape has conquered.

Competing well with all the major servers, Netscape provides versions that run on Windows 95, NT, and UNIX systems. Free evaluation copies are available for trial use at Netscape's Web site, `www.netscape.com`, and licensed copies begin at $495.

Like the popular Netscape browser, the Netscape servers rank high in customer satisfaction, ease of use, and reliability. Their greatest achievement, however, may lie in their highly integrated design. Although it's sometimes hard to tell who's pushing who, the Netscape products exemplify the recent trend among server software developers to blend high-end user functionality into a single-user interface. Dovetailing Internet-related functions with normal operating system functions is another related characteristic. New and experienced users alike can enjoy such features as built-in search engines, remote editing capabilities, automatic cataloging, and built-in log analysis tools.

Netscape rounds out the program by offering a comprehensive support package that includes

- Ninety days of free support
- Incident-based support ($89 per incident, or three for $199)
- Annual support ($399 for 12 hours per 5 days, and $599 for 12 hours per 7 days)
- Online Help

WebSite

O'Reilly and Associates offers WebSite, the commercial server for Windows 95/NT, which is another of the new breed of servers offering multiple, integrated functionality. Chief among this package's capabilities are

- The capability for remote administration
- CGI 1.3 and Visual Basic toolkit
- HotDog editor
- Integrated indexing and search tools

WebSite Pro adds

- Enhanced crytographic security
- WebSite API (with Microsoft ISAPI-ISA compatibility)
- ODBC/SQL database access using API-integrated Cold Fusion 1.5
- API-integrated Perl 5
- Server-side Java SDK
- One-button publishing with Navigator Gold

WebSite is truly one of the most versatile of the server products currently available. You can download evaluation copies at the company's Web site: `www.ora.com`. Licensed copies are available from the vendor at the competitive price of $249 ($495 for the Pro version).

WebSite is highly regarded for its ease of installation and use, reliability, and overall functionality. Most notable are the very well-written user's manuals that are part of the package.

O'Reilly's tech support is also known to be of high quality, which lends credence to the old saying that you get what you pay for. The first 90 days of telephone support are free, but the same service will cost you $450 per year thereafter. You can also access online support and discussion groups and the newsletter, WebSite-talk.

Archive is available at `http://www.ora.com/archives/website-talk/`. Registered product owners are automatically entered on the mailing list.

WebSTAR

The leading Macintosh Web server is WebSTAR by StarNine. The product's easy installation, ease of use, and readily accessible (and eager to help) support team are among its greatest strengths. Other notable features include

- Enables scripting and recording with AppleScript
- Performs faster than MacHTTP 2.0
- Supports WebSTAR API
- Supports thousands of connections per hour
- Integrates with both Mac and SQL databases
- Enables remote administration from anywhere on the Internet
- Controls multiple servers from Mac

Fully licensed copies are available from Apple for $499, and 30-day trial copies are available at `keys@starnine.com`. A serial number is required to activate the software.

Sun Netra j Servers

Brand new in the market, and fully described at `www.sun.com`, is the Sun Microsystems Netra j server. The Netra j, according to a recent Sun press release, is designed to dramatically reduce the total cost of owning an IT infrastructure. The package is meant to complement the "zero-administration" JavaStation network computer. The Netra j server line features fully integrated software for Java computing, easy-to-use browser-based administration and management tools, secure access to data on other computers, and outstanding price/performance. The Netra j server is designed for transaction and process-oriented business applications. Netra j server configurations are available to support from tens to thousands of networked computers.

Designed exclusively to run on the Sun Solaris platform, this package is extremely pricey in comparison to virtually all other server packages. Sold as integrated hardware/software packages, the Netra j servers start at $7,695 and quickly go up to $200,000 and more. For those users who require the ability to do Java computing without jeopardizing previous investments in legacy applications and infrastructure, Sun and the Netra j server may offer some truly unique solutions.

This is a powerhouse product for existing Sun shops but is well beyond the range of what will probably remain the greatest majority of Internet users for quite some time.

Summary

In their role as the engines for the traffic on the Web, Web servers are becoming more and more sophisticated with each passing month. The brief review in the previous section illustrates the fact that the most popular servers, whether free or quite expensive, all now employ a wide array of utilities that are designed to more fully integrate with existing systems and provide a broader range of functionality.

The latest developments in server technology are becoming so tightly integrated with operating system and network operating system software packages that eventually they will become one and the same. All of this points to the fact that the role of the Web server has merged into mainstream computing. Yesterday's computer users discovered the Web and, for a brief time, explored it haphazardly with no guarantee of success. Tomorrow's computer users, however, will expect the Web to be paved and the onramps and offramps to be wide, clear, and abundant.

General advice for those considering a Web server purchase or upgrade:

> Spend adequate time defining your present and future needs. Discuss the costs, benefits, and challenges of setting up an in-house Web server with colleagues from other departments. Explore all possible applications for a married Internet/intranet system within your organization.

Specific advice for those considering a Web server purchase or upgrade:

> Spend a considerable amount of time perusing the bookstore shelves and the magazine racks and browsing the Web for information relating to your intended purchase. The options available to you are numerous. Be certain that you've nailed down your server requirements and then select the most compatible, stable, well-integrated, multifunctional, expandable, and affordable system you can find.

Finally, take an active role in educating others in the use of this marvelous tool. After all, putting the world within everyone's grasp has become the overarching purpose of the World Wide Web. As a Web development professional, you have a unique opportunity to help others glimpse what you've already seen.

VII PART

IN THIS PART

- The Emergence of Extensible Markup Language *781*
- Internationalizing the HTML Character Set and Language Tags *793*
- Point/Counterpoint: Pure HTML *813*
- HTML Beyond the Web *825*

The Future of HTML

CHAPTER 38

The Emergence of Extensible Markup Language

by Dmitry Kirsanov

IN THIS CHAPTER

- The Premises of XML *782*
- Well-Formed XML Documents *784*
- XML DTDs and Valid XML Documents *785*
- Linking Capabilities of XML *788*
- The Prospects of XML *790*

Everyone interested in the future of the Web must be curious—and pretty uncertain—about what may be the outcome of the HTML case. Browser wars and incompatible extensions tearing apart the language are not only bad by themselves, they're a sure sign that something's going wrong with HTML in the first place. It may sound like heresy that the tongue spoken by millions of Web pages is approaching the end of its useful life, but many serious observers cannot suppress exactly that feeling.

If we strip away for a moment the innumerable struts, crutches, and sophisticated gizmos that make the HTML golem walk and speak and look alive, what we'll see will be a pretty simple (not to say primitive) markup language designed for basic documents of a quite predictable structure. Just headers, paragraphs, block quotations, and the good old ADDRESS at the end. Does *this* sound like a model structure for the whole world of information out there?

Of course, now HTML has tables and fonts and so on. Indeed, visual HTML extensions (or inline images, as the last resort) enable you to emulate any document structure—that is, make the document *look* as if it is properly structured. But as a result, the internal structure of the text will inevitably become illogical, cumbersome, presentation-oriented (and with images, the text may cease to be text at all). This is very likely to prevent reusing the document in the future; it becomes difficult even to convert it into another visually oriented format, not to mention isolate its logical elements.

Fortunately, there's a new important language designed to address this issue. XML (eXtensible Markup Language) is a simple and compact subset of SGML designed specifically for use on the Internet in the way that HTML is currently used. This new project of W3C is gaining momentum at a surprising rate, and everybody seriously concerned with HTML may want to check it out. Maybe someday, you'll find yourself saying, "back in those days when everyone was using HTML..."

The Premises of XML

As with every new and promising technology, it is probably more important to explain what it isn't rather than what it is. XML, just like SGML, is not a page layout or graphics language. By itself, XML provides even fewer presentation tools than you have with HTML. Strictly speaking, it's not even a markup language, but rather a system making it possible to build such languages to match any conceivable document type.

Chapter 3, "SGML and the HTML DTD," explains the HTML document type definition (DTD), the specification of HTML tags and document structure written in SGML. Similarly, with XML you can build a DTD that exactly matches the structure of your document and introduces a set of self-explanatory, logically organized tags and attributes fine-tuned for your markup needs.

By attaching the DTD with the document sent over the network, you can ensure that the XML software reading the document can parse it correctly and thereby guarantee its correct formatting, conversion, adding to a database, or whatever the receiver will choose to do with the document. In short, with XML you can create your own HTML, or XYZML, or Whatever-You-Like-ML! (It's no surprise such a language was called "extensible" in the first place.)

It is important to understand that XML isn't better than HTML because it makes it easier to change fonts or position images. In the visual presentation realm, XML is not better than HTML (and some might say it's worse because it lacks all those neat Netscape enhancements—unless you've defined them in your DTD). It was the intention of the creators of the language that the visual presentation of an XML document can be (optionally) specified by an attached style sheet, which is an *external* mechanism for XML just as it is for HTML.

XML's visualization power is thus completely determined by the style sheet language you use—for example, Cascading Style Sheets (CSS) or Document Style Semantics and Specification Language (DSSSL)—and if you don't care about logical markup you can achieve exactly the same *visible* results by using this chosen style sheet mechanism with HTML. (Remember that you can use the neutral SPAN tag in HTML to apply any attributes, style names included, to arbitrary fragments of text.) It is when the proper internal structure of your data really matters that XML easily outshines HTML.

XML specification (found at `http://www.textuality.com/sgml-erb/WD-xml-lang.html`) defines the language in the terms of behavior of a *parser*, which is a piece of software whose sole purpose is to understand the element structure of your document and break it down into nested elements in accordance with the DTD. Another program (termed simply "application" in the XML specification) is supposed to obtain the document thus dissected from the parser and process it further. Exactly what the application performs on the document is outside the scope of XML; for instance, it may be a browser that displays the document using an appropriate style sheet.

Because XML is a subset of SGML, an XML document is almost always a valid SGML document; small discrepancies between these two languages are likely to be eliminated soon with the acceptance of certain amendments to the SGML standard. The relation between XML and HTML is more complex. With the capability to define new tags, XML documents are not likely to count as valid HTML very often; on the other hand, an HTML file is relatively easy to make XML-conformant on one of the two levels of conformance (described later in this chapter), depending on whether or not you provide a DTD for your document.

I don't attempt a real tutorial of XML in this chapter for two reasons. First, one chapter's space is surely insufficient to cover even the basics of the language, and second, the language itself is so young and unstable that it is probably untimely to start teaching it in a serious fashion. (A quote from the language specification: "Please be advised that the draft you are now reading is unusually volatile.") Instead, this chapter presents a couple of small examples that will help you to quickly grasp the "look and feel" of the language.

In a sense, XML is positioned somewhere in between SGML and HTML, with the intent of its creators being to combine the best features of these two languages. However, XML is much closer to SGML than to HTML, and although knowledge of HTML will help you understand the most obvious XML features, an acquaintance with SGML syntax and ideology would be of much more help. So I recommend that you brush up what you remember from reading Chapter 3 before proceeding to subsequent sections of this chapter.

Well-Formed XML Documents

If you're scared by the prospect of learning the art of writing document type definitions, there's good news for you: With XML, you can create a document even without DTD. If the lack of a DTD is the sole violation of XML requirements, such a document is called *well-formed,* as opposed to a *valid* document that has a DTD provided (or referred to). Thus, well-formedness is the lower of the two levels of XML conformance, but although it is inferior to validity, it is still very useful.

The permission to omit the DTD means, in essence, that you are free to use *any* tags that seem necessary for your document. You may just go wild and write in plain English what each part of your text is supposed to represent. For example, if you're a grammarian, you could try the following:

```
<SENTENCE>
  <SUBJECT TYPE="COMPLEX">
    <ARTICLE TYPE="INDEFINITE">A</ARTICLE>
    <ADJECTIVE>quick</ADJECTIVE>
    <ADJECTIVE>brown</ADJECTIVE>
    <NOUN>fox</NOUN>
  </SUBJECT>
  <VERB TYPE="INTRANSITIVE">jumps</VERB>
  <PREPOSITION>over</PREPOSITION>
  <ARTICLE TYPE="INDEFINITE">a</ARTICLE>
  <OBJECT>
    <ADJECTIVE>lazy</ADJECTIVE>
    <NOUN>dog</NOUN>
  </OBJECT>.
</SENTENCE>
```

Given this well-formed document, an XML parser will be able to break the text into the elements that you deemed essential in this case and pass each element to the application along with its name (derived from the tag name) and attribute information you provided. It is the application, not the XML parser, that must be programmed to perform some useful tasks with this structured information. Most often, you'll need to provide the application with a style sheet that associates certain formatting parameters with each element. (In our example, the style sheet might specify displaying different parts of speech in different colors.)

In fact, even "plain English" is not an obligatory requirement for your tags. Creators of XML intended that the language must be international from the very beginning. They painstakingly

identified all Unicode characters that may be called "letters" in some sense or in some language and included these characters in the set of characters allowed in element and attribute names. This means that you can write your tags in Russian or Chinese instead of English.

There are, of course, numerous restrictions that are imposed even on well-formed documents. Some of these requirements are even stricter than those of HTML and thus deserve special attention:

- In XML, *every* start tag must have a corresponding end tag (unless it is an empty tag that takes special form, as described in the next item). Tags should be properly nested; that is, you can't have an open tag within the scope of some other tag and the corresponding end tag outside that scope.
- If a tag is empty by its nature (in other words, the corresponding element can never contain any text), it must have a forward slash (/) before the closing greater than symbol (>), as in the following example:

  ```
  <IMG alt="XML logo" src="http://www.ucc.ie/xml/xml.gif"/>
  ```

  Such tags are the only tags that do not have corresponding end tags.
- All attribute values without exception must be enclosed in quotes—either single quotes ('') or double quotes ("").

In fact, the preceding requirements are the *only* ones that you must satisfy to make your HTML files well-formed XML. It doesn't matter which browser's HTML extensions you use or whether you "abuse" HTML tags or not. XML is a truly liberal language; it makes you a creator of your own universe whose rules you're unlikely to break simply because you established them.

Those familiar with the material in Chapter 3 may be wondering about another essential part of any SGML application—the SGML declaration that always accompanies a DTD. Because XML is pretty simple compared to SGML, its authors decided to omit this part of the language; XML parsers (or SGML software processing XML documents) should behave as if all XML documents were assigned the same generic SGML declaration that is listed in an appendix to the XML standard.

XML DTDs and Valid XML Documents

Although in many cases well-formed XML documents are sufficient for practical purposes, designing a DTD for your document has a number of advantages:

- First and foremost, a DTD allows an XML parser to *validate* your document (which is why such documents are called valid). When validating, the parser checks for misspelled tags or attributes, for errors in types of attribute values and in elements' content models (covered in Chapter 3), and so on. For HTML, similar validation services exist that will check your file against one of the existing HTML DTDs.

- For a human reader, a DTD is a convenient way to quickly learn the structure of the particular type of documents. Compared to SGML, the simplified DTD syntax of XML is very straightforward and unambiguous.
- With DTD, you can define not only elements and their attributes, but also *entities.* (See "Entity Declarations," later in this chapter.) Similarly to macros in word processors or `#define` preprocessor instructions in C, entities can be used to abbreviate text strings and markup instructions in an obvious and easy-to-modify manner. Also, you can use *external entities* to refer to other XML documents, DTDs, or binary data located in separate files.

Accessing the DTD

Let's examine an example of a valid XML document, namely a play by Shakespeare (*The Tempest*) marked up by Jon Bosak, one of the authors of XML. The package (`http://sunsite.unc.edu/pub/sun-info/standards/dsssl/egs/21_shaks/`) includes, besides the XML document and its DTD, a DSSSL style sheet that contains formatting instructions for each element and a Postscript output of a DSSSL processor that formatted the play. (See Chapter 21, "JavaScript Style Sheets and Other Alternatives to CSS.")

Here's the very beginning of the XML document `play.xml`:

```
<?XML version="1.0"?>
<!DOCTYPE play PUBLIC "-//Free Text Project//DTD Play//EN">
```

The first line here is an *XML declaration,* a special instruction that is XML-specific and would not be recognized by an SGML parser because of the `<?` delimiter (which is not used in SGML). Here, the XML declaration provides information about the version of XML standard that the document conforms to.

Next comes the `DOCTYPE` statement that, like its namesake in SGML, provides the DTD for the document to be parsed. In XML, a DTD may be in two parts: *internal* is contained in the document file itself, and *external* is referenced by its URL or *public identifier* (covered in Chapter 3), with the internal part taking precedence over the external one in a case of conflict.

In our example, only the external part of DTD is present, which is referred to by the public identifier preceded by the keyword `PUBLIC`. An XML parser is supposed to be able to retrieve the text of the DTD using its public identifier (that is, to translate the identifier into a URL or some other sort of physical address). If the DTD you're using is not assigned a well-known public identifier, you should provide a URL instead of it, with the `SYSTEM` keyword instead of `PUBLIC`. Here is an example:

```
<!DOCTYPE HTML SYSTEM "http://www.foo.com/myfiles/html3x.dtd">
```


Finally, to provide an internal part for a DTD, you must put it in brackets within the `DOCTYPE` declaration. Such a declaration may also contain a `SYSTEM` or `PUBLIC` external reference, as in the following example:

```
<!DOCTYPE HTML SYSTEM "http://www.foo.com/myfiles/html3x.dtd"
[
  <!-- your DTD goes here -->
]>
```

Element Declarations

The name right after the `DOCTYPE` keyword in the preceding statements is the name of the *root element* of your document type, the top-level element that encloses all other elements. In HTML, this element is named `HTML`, and in our Shakespearean example it is named `PLAY`. Here's how the `PLAY` element is defined in `play.dtd`:

```
<!ELEMENT play (title, fm, personae, scndescr, playsubt, induct?,
                                    prologue?, act+, epilogue?)>
```

You can see that the *content model* (discussed in Chapter 3) for this element is quite simple and immediately translatable into human talk: "A `PLAY` is formed by its `TITLE`, followed by the front matter (`FM`), followed by the list of dramatis `PERSONAE`, and so on." The question mark indicates optional elements, and the plus sign indicates the elements that may occur once or more. Note that the XML spec prescribes to drop the SGML minimization parameters that are useless in XML, which doesn't permit tag omission anyway.

One more excerpt from `play.dtd` shows a hierarchical set of related tags to mark a personage's speech:

```
<!ELEMENT speech   (speaker+, (line ¦ stagedir ¦ subhead)+)>
<!ELEMENT speaker  (#PCDATA)>
<!ELEMENT line     (stagedir ¦ #PCDATA)+>
<!ELEMENT stagedir (#PCDATA)>
<!ELEMENT subhead  (#PCDATA)>
```

Thus a `SPEECH` is constituted by one or more `SPEAKER` elements followed by at least one of the `LINE`, `STAGEDIR` (stage direction), or `SUBHEAD` elements, in no particular order. (The ¦ sign means that any one of the connected particles may occur.) The `#PCDATA` keyword has the meaning of "any character data without tags"; thus, the `SPEAKER`, `STAGEDIR`, and `SUBHEAD` elements are allowed to contain only text characters, while a `LINE` may have `STAGEDIR`s intermingled with text.

Note that nothing in the definition of `LINE` (except the name) suggests that what the element contains is really a line of verse. It is just implied to be so by the person who did markup, and it may be formatted as a line if an appropriate style sheet is used. However, XML only serves as an intermediator between the author and the formatter, and it is not intended to describe the nature of data elements that are marked up with it.

Here's a SPEECH element exemplifying these DTD provisions:

```
<SPEECH>
<SPEAKER>PROSPERO</SPEAKER>
<LINE><STAGEDIR>Aside</STAGEDIR> The Duke of Milan</LINE>
<LINE>And his more braver daughter could control thee,</LINE>
<LINE>If now 'twere fit to do't. At the first sight</LINE>
<LINE>They have changed eyes. Delicate Ariel,</LINE>
<LINE>I'll set thee free for this.</LINE>
<STAGEDIR>To FERDINAND</STAGEDIR>
<LINE>A word, good sir;</LINE>
<LINE>I fear you have done yourself some wrong: a word.</LINE>
</SPEECH>
```

Entity Declarations

Entities can be declared in a DTD as follows:

```
<!ENTITY me "Dmitry Kirsanov, St.Petersburg, Russia">
```

In the document, such an entity can be used similarly to mnemonic character entities of HTML:

```
This document was created by &me; on April 21, 1997
```

Another syntax is used to define entities that refer to external files or documents, as in the following example:

```
<!ENTITY mypage SYSTEM "http://www.symbol.ru/dk/index.xml">
<!ENTITY xml-logo SYSTEM "http://www.ucc.ie/xml/xml.gif" NDATA gif>
```

In the second declaration, `gif` is the name of a *notation* (similar to a data type), which must be declared somewhere in the DTD along with information on where an XML processor can access a helper software capable of handling data in this notation.

Now, `&mypage;` and `&xml-logo;` entities can be used in documents using this DTD. However, XML specification does not prescribe the exact behavior of the XML application on encountering such an entity. For example, it might incorporate it into the text of the current document, or it might present it as a link that the user can activate.

Linking Capabilities of XML

The first part of the XML specification referenced earlier in this chapter (`http://www.textuality.com/sgml-erb/WD-xml-lang.html`) fully describes the syntax of the language. However, a second part of the specification at `http://www.textuality.com/sgml-erb/WD-xml-link.html` is concerned with *linking capabilities* of XML documents similar to the hypertext features of HTML.

In some sense, this is beyond the scope of an SGML-like language, because the *semantics* of elements (including their capability to link) should be left to creators of SGML or XML applications. On the other hand, links are difficult to implement using common style sheet languages and therefore need support from a deeper level. For this reason (as well as, apparently, to make XML a better competitor to HTML), authors of the language chose to make linking provisions a part of the main body of the XML standard.

The linking model of XML (which can be used with SGML as well) is much more versatile than that of HTML. It is largely based on the efforts of other SGML projects such as HyTime and Text Encoding Initiative (TEI). I don't describe the detailed syntax here, because it is likely to change in subsequent drafts of the specification. It is worth noting, however, that all linking constructs are implemented on the level of reserved attributes and their values, not elements. This means that you can easily turn any element into a link by expanding its list of attributes.

A simple link intended to connect the XML document you're reading to some other Internet resource may be assigned a number of parameters, including the following:

- Strings describing the *role* of the link in the document and its associated *label* (for example, this information may be displayed by the browser in its status line)
- A parameter defining whether the linked resource should, upon activating the link, replace the current document, be opened in a new context (for example, in a new window), or be embedded into the current document in place of the link
- A parameter indicating whether the user can activate the link or it is activated automatically when encountered by the processing application

Very important, although perfectly backwards-compatible, enhancements are proposed for the syntax of URLs that are used in XML to address resources. Using search parts (separated by ?) or fragment identifiers (separated by #) of special form, you can access any part of another XML document without the necessity of attaching a label to it beforehand (such as `<A NAME="...">` in HTML).

The syntax of such *extended locators* in XML is capable of expressing formally such requests as "everything from the third paragraph of the first chapter to the end of the chapter" or "the whole section to which the last sidebar belongs." Of course "chapters" or "sidebars" are to be declared as corresponding elements in the DTD of the linked document.

Moreover, you may prescribe whether the necessary fragment is extracted right on the server where the linked document is stored (which may result in considerable bandwidth savings) or whether the server sends you the whole document and your application needs to winkle out the part you requested.

XML links can be not only inline but also *out-of-line*, which means that they do not need to reside in one of the documents connected by the link. Links can be grouped so that all the links in a group are activated at once. Finally, you can create collections of interlinked documents and inform the processing application about what other documents contain links to the current one.

The Prospects of XML

XML is really a quite recent development. Its basic principles were all worked out in 1996, and the first draft of the specification was presented to the public at the SGML 96 conference in November. A revised draft (still not final specification) was released in March 1997. As of this writing, software created for parsing XML files is all experimental and can fit on one 3.5-inch disk.

Nevertheless, this new development is likely to seriously impact the Web industry in the near future, and in the more distant future, it could completely change the landscape we're accustomed to. Here are several points that were chosen as the most important goals by the designers of the language. These could become the key advantages in the competition of XML with other technologies:

- **XML shall be straightforwardly usable over the Internet.** Web servers in use today require minimal configuration changes to be able to serve XML documents. The standard way to link and bind together XML documents and DTDs is via URLs that are understood by the majority of Internet software.
- **It shall be easy to write programs that process XML documents.** The experimental XML software mentioned earlier is all written in Java, with some of the experimental XML parsers being contained in class files of a few kilobytes in size.
- **XML documents should be human-legible and reasonably clear.** With the users of XML being able to create their own tags and attributes with self-explanatory names, an XML file is likely to be nearly as readable as (and in some cases, even more readable than) plain text.
- **The XML standard should be prepared quickly.** It is not yet finalized, but the amount of work done in such a short term is impressive.
- **The design of XML shall be formal and concise.** Syntax descriptions in XML specification use a formal grammar that is concise, easy to understand, and easy to translate into programming code.
- **XML documents shall be easy to create.** As you've seen in this chapter, the concept of well-formedness enables you to quickly mark up any document or translate it from HTML to XML.
- **Terseness in XML markup is of minimal importance.** Clear and unambiguous syntax was always given preference over the possibility of saving a few keystrokes.

The XML technology is in an embryonic state, so any attempts to augur its future are almost sure to not come true. However, the growing interest in XML shown by many people concerned with Web development is a clear indication that the Web is eager to try out something more powerful and elegant than the HTML of today.

Summary

In this chapter, you've become acquainted with an important new development, the extensible markup language, that, combined with a style sheet language, has a potential to eventually replace the heavily overused HTML of nowadays. Because it is a simplified subset of SGML, XML allows you to create customized markup tags and specify the corresponding document strucure for any type of documents you might need to store or disseminate. In particular, you learned about the following topics:

- What XML is and isn't, in what aspects it is better than HTML, and why it is important to mark up the logical structure of a document in the first place.
- What are well-formed XML documents, and how they can be easily created from HTML documents.
- What are the advantages of providing an XML document with a DTD, and how to refer to the DTD file from within an XML document.
- How elements and entities are declared in an XML DTD and then used in the documents.
- What linking capabilities are provided by the XML standard, and how they extend the HTML hypertext model.
- What were the main goals of the creators of XML, and in what ways its design principles help it compete with HTML and other technologies.

CHAPTER 39

Internationalizing the HTML Character Set and Language Tags

by Dmitry Kirsanov

IN THIS CHAPTER

- Character Encoding Standards *794*
- MIME *799*
- HTML Character Set *800*
- Language Identification *806*
- Language-Specific Presentation Markup *807*
- Font Issues *810*

No book on HTML is complete without a section on the ways to overcome the pronounced Western bias in the language and to provide for its fruitful application in the worldwide multilingual environment. This chapter covers the main approaches to this problem, both those used by practicing webmasters all around the world and those suggested by standard-setting bodies.

The primary problem related to HTML internationalization (or *i18n*, as it is often abbreviated: *i* plus 18 in-between letters plus *n*) is the correct rendering of characters used by other languages. This is why I start by examining different standards of character encoding (character sets). These standards are classified by the length of bit combinations they use, from 7-bit ASCII to Unicode and ISO 10646.

Various HTML internationalization issues were first crystallized in the important document RFC 2070 (`http://ds.internic.net/rfc/rfc2070.txt`). Then, RFC 2070 provisions were incorporated in the DTD for HTML version 4.0. However, I discuss, for the most part, the material of RFC 2070 and pay special attention to the cases where it is not identical to the declarations of HTML 4.0 DTD.

In the field of HTML proper, this chapter starts by investigating the new document character set as defined in RFC 2070 and HTML 4.0. You will be introduced to the important distinction between the document character set and external character encoding. You'll learn about existing methods of specifying external character encoding, proposed additions to handle multilanguage form input, as well as a number of real-world problems related to the HTML character set.

Another big part of the HTML internationalization problem is language markup, that is, specifying the language of a piece of text in order to help user agent software to render it, observing the typography conventions of that language. Some language-specific aspects of text presentation are also addressed in RFC 2070, which introduces tools to control writing direction, cursive joining, rendering of quotation marks, text alignment, and hyphenation. As a conclusion, I cover briefly the font issues related to HTML internationalization.

Character Encoding Standards

It so happened that the computer industry has been flourishing in the country whose language uses one of the most compact alphabets in the world. However, not long after the first computers had learned to spell English, a need arose to encode characters from other languages. In fact, even the minimum set of Latin letters and basic symbols has been for some time the subject of controversy between two competing standards, ASCII and (now almost extinct) EBCDIC; no wonder that for other languages' alphabets, a similar muddle has been around for much longer (in fact, it's still far from over).

As explained in Chapter 3, "SGML and the HTML DTD," a character encoding (often called *character set* or, more precisely, *coded character set*) is defined—first, by the numerical range of codes; second, by the repertoire of characters; and third, by a mapping between these two sets. You see that the term *character set* is a bit misleading because it actually implies *two* sets and a relation between them. Probably the most precise definition of a character encoding in mathematical terms is given by Dan Connolly in his paper "Character Set Considered Harmful" (`http://www.w3.org/pub/WWW/MarkUp/html-spec/charset-harmful.html`): "A function whose domain is a subset of integers, and whose range is a set of characters."

The range of codes is limited by the length of the sequence of bits (called *bit combination*) used to encode one character. For instance, a combination of 8 bits is sufficient to encode the total of 256 characters (although not all of these code positions may actually be used). The smaller the bit combination size, the more compact the encoding (that is, the less storage space is required for a piece of text), but at the same time, the fewer total characters you can encode.

It is quite logical to codify characters using bit combinations of the size most convenient for computers. Because modern computer architecture is based on bytes (also called *octets*) of 8 bits, all contemporary encoding standards use bit combinations of 8, 16, or 32 bits in length. The next sections survey the most important of these standards to see the roles they play in today's Internet.

7-Bit ASCII

The so-called 7-bit ASCII, or US ASCII, encoding is equivalent to the international standard named ISO 646 (`ftp://dkuug.dk/i18n/ISO_646`) established by the International Standards Organization (ISO). This encoding actually uses octets of 8 bits per character, but it leaves the first (the most significant) bit in each octet unused (it must be always zero). The 7 useful bits of ISO 646 are capable of encoding the total of 128 characters.

This is the most ubiquitous encoding standard used on the overwhelming majority of computers worldwide (either by itself or as a part of other encodings, as you'll see shortly). ISO 646 can be called international in the sense that there are precious few computers in the world that use other encodings for the same basic repertoire of characters. It is also used exclusively for keywords and syntax in all programming and markup languages (including SGML and HTML), as well as for all sorts of data that is human-editable but of essentially computer nature, such as configuration files or scripts.

However, with regard to the wealth of natural languages spoken around the world, ISO 646 is very restrictive. In fact, only English, Latin, and Swahili languages can use plain 7-bit ASCII with no additional characters. Most languages whose alphabets (also called *scripts* or *writing systems*) are based on the Latin alphabet use various accented letters and ligatures.

The first 32 codes of ISO 646 are reserved for *control characters*, which means that they invoke some functions or features in the device that reads the text rather than produce a visible shape (often called *glyph*) of a character for human readers. As a rule, character set standards are reluctant to exactly define the functions of control characters, as these functions may vary considerably depending on the nature of text processing software.

For example, of the 32 control characters of ISO 646, only a few (carriage return, line feed, tabulation) have more or less established meanings. For use in texts, most of these codes are just useless. The code space thus wasted in vain is a holdover from the old days, when these control characters used to play the role of today's document formats and communication protocols.

8-Bit Encodings

The first natural step to accommodate languages that are more letter-hungry than English is to make use of the 8th bit in every byte. This provides for additional 128 codes that are sufficient to encode an alphabet of several dozens letters (for example, Cyrillic or Greek) or a set of additional Latin letters with diacritical marks and ligatures used in many European languages (such as ç in French and ß in German).

Unfortunately, many more 8-bit encodings exist in the world than are really necessary. Nearly every computer platform or operating system making its way onto a national market without a strong computer industry of its own brought along a new encoding standard. For example, as many as three encodings for the Cyrillic alphabet are now widely used in Russia, one being left over from the days of MS-DOS, the second native to Microsoft Windows, and the third being popular in the UNIX community and on the Internet. A similar situation can be observed in many other national user communities.

ISO, being an authoritative international institution, has done its best to normalize the mess of 8-bit encodings. The ISO 8859 series of standards (which is available at `http://wwwwbs.cs.tu-berlin.de/~czyborra/charsets/`) covers almost all extensions of the Latin alphabet as well as the Cyrillic (ISO 8859-5), Arabic (ISO 8859-6), Greek (ISO 8859-7), and Hebrew (ISO 8859-8) alphabets. All these encodings are backward compatible with ISO 646; that is, the first 128 characters in each ISO 8859 code table are identical to 7-bit ASCII, while the national characters are always located in the upper 128 code positions.

Again, the first 32 code positions (128 to 159 decimal, inclusive) of the upper half in ISO 8859 are reserved for control characters and should not be used in texts. This time, however, many software manufacturers chose to disregard the taboo; for example, the majority of True Type fonts for Windows conform to ISO 8859-1 in code positions from 160 upward, but use the range 128–159 for various additional characters (notably the em dash and the trademark sign). This leads to the endless confusion about whether one can access these 32 characters in HTML (the DTD, following ISO 8859, declares this character range unused). HTML internationalization extensions resolve this controversy by making it possible to address these characters via their Unicode codes.

The authority of ISO was not, however, sufficient to position all of the 8859 series as a strong alternative to the ad hoc national encodings supported by popular operating systems and platforms. For example, ISO 8859-5 is hardly ever used to encode Russian texts except on a small number of computers.

On the other hand, the first standard in the 8859 series, ISO 8859-1 (often called ISO Latin-1), which contains the most widespread Latin alphabet extensions serving many European languages, has been widely recognized as *the* 8-bit ASCII extension. Whenever a need arises for an 8-bit encoding standard that is as international as possible, you're likely to see ISO 8859-1 playing the role. For instance, ISO 8859-1 served as a basis for the document character set in HTML versions up to 3.2. (In 4.0, this role was taken over by Unicode, as discussed next.)

16-Bit Encodings

Not all languages in the world use small alphabets. Some writing systems (for example, Japanese and Chinese) use *ideographs*, or hieroglyphs, instead of letters, each corresponding not to a sound of speech but to an entire concept or word. As there are many more words and especially conceivable ideas than there are sounds in a language, such writing systems usually contain many thousands of ideographs. An encoding for such a system needs at least 16 bits (2 octets) per character, which allows you to accommodate the total of 2^{16} = 65536 characters.

Ideally, such a 16-bit encoding should be backward compatible with the existing 8-bit (and especially 7-bit ASCII) encodings. This means that an ASCII-only device reading a stream of data in this encoding should be able to correctly interpret at least ASCII characters if they're present. This is implemented using *code switching*, or *escaping* techniques: Special sequences of control characters are used to switch back and forth between ASCII mode with the 1 octet per character and 2-octet modes (also called *code pages*). Encodings based on this principle are now widely used for Asian languages.

Code switching works all right, but one interesting problem is that the technique makes it ambiguous what to consider a coded representation of a character—is it just its 2-octet code or the code preceded by the switching sequence? It is obvious that the "extended" national symbols and ASCII characters are not treated equally in such systems, which may be practically justifiable but is likely to pose problems in the future.

In late 80s, the need for a truly international 16-bit coding standard became apparent. The Unicode Consortium (`http://www.unicode.org`), formed in 1991, undertook to create such a standard called Unicode. In Unicode, every character from the world's major writing systems is assigned a unique 2-octet code. According to the tradition, the first 128 codes of Unicode are identical to 7-bit ASCII, and the first 256 codes, to ISO 8859-1. However, strictly speaking, this standard is not backward compatible with 8-bit encodings; for instance, Unicode for the Latin letter *A* is 0041 (hex), while ASCII code for the same letter is simply 41.

The Unicode standard deserves a separate book to describe it fully (in fact, its official specification is available in book form from the Unicode Consortium). Its many blocks and zones cover all literal and syllabic alphabets that are now in use, alphabets of many dead languages, lots of special symbols, and combined characters (such as letters with all imaginable diacritical marks, circled digits, and so on).

Also, Unicode provides space for more than 20 thousand unified ideographs used in Asian languages. Contrary to other alphabets, ideographic systems were treated on a language-independent basis. This means that an ideograph that has similar meanings and appearance across the Asian languages is represented by a single code despite the fact that it corresponds to quite different *words* in each of the languages and that most such ideographs have country-specific glyph variants.

The resulting ideographic system implemented in Unicode is often abbreviated CJK (Chinese, Japanese, Korean) after the names of the major languages covered by this system. CJK unification reduced the set of ideographs to be encoded to a manageable (and codable) number, but the undesirable side effect is that it is impossible to create a single Unicode font suitable for everyone; a Chinese text should be displayed using slightly different visual shapes of ideographs than a Japanese text even if they use the same Unicode-encoded ideographs.

The work on Unicode is far from complete, as about 34 percent of the total coding space remains unassigned. Workgroups in both the Unicode Consortium and ISO are working on selection and codification of the most deserving candidates to colonize Unicode's as-of-yet wastelands. A good sign is that the process of Unicode acceptance throughout the computer industry is taking off; for example, Unicode is used for internal character coding in Java programming language and for font layout in Windows 95 and Windows NT operating systems.

ISO 10646

Although Unicode is still not widely used, in 1993 ISO published a new, 32-bit encoding standard named ISO/IEC 10646-1, or Universal Multiple-Octet Coded Character Set (abbreviated UCS and outlined at `http://www.dkuug.dk/JTC1/SC2/WG2/docs/standards`). Just as 7-bit ASCII does, though, this standard leaves the most significant bit in the most significant octet unused, which makes it essentially a 31-bit encoding.

Still, the code space of ISO 10646 spans the tremendous amount of 2^{31} = 2147483648 code positions, which is much more than could be used by all languages and writing systems that ever existed on Earth. What, then, is the rationale behind such a huge "Unicode of Unicodes?"

The main reason for developing a 4-octet encoding standard is that Unicode actually cannot accommodate all the characters for which it would be useful to provide encoding. Although a significant share of Unicode codes are still vacant, the proposals for new character and ideograph groups that are now under consideration require in total several times more code positions than are available in 16-bit Unicode.

Extending Unicode thus seems inevitable, and it makes little sense to extend it by 1 octet because computers will have trouble dealing with 3-octet (24-bit) sequences; 32-bit encoding, on the other hand, is particularly convenient for modern computers, most of which process information in 32-bit chunks.

Just as Unicode extends ISO 8859-1, the new ISO 10646 is a proper extension of Unicode. In terms of ISO 10646, a chunk of 256 sequential code positions is called a *row*, 256 rows constitute a *plane*, and 256 planes make up a *group*. The whole code space is thus divided into 128

groups. In such terms, Unicode is simply plane 00 of group 00, the special plane that in ISO 10646 standard is called the *Basic Multilingual Plane* (BMP). For example, the Latin letter A (Unicode 0041) is in ISO 10646 fully coded 00000041. As of now, ISO 10646 BMP is absolutely identical to Unicode, and it is unlikely that these two standards will ever diverge.

ISO 10646 specifies a number of intermediate formats that do not require using the codes in the *canonical form* of 4 octets per character. For example, the UCS-2 (Universal Character Set, 2-octet format) is indistinguishable from Unicode as it uses 16-bit codes from the BMP. The UTF-8 format (UCS Transformation Format, 8 bits) can be used to incorporate, with a sort of code switching technique, 32-bit codes into a stream consisting of mostly 7-bit ASCII codes. Finally, the UTF-16 method was developed to access more than a million 4-octet codes from within a Unicode/BMP 2-octet data stream without making it incompatible with current Unicode implementations.

Most probably, ISO 10646 will be rarely used in its canonical 4-octet form. For most texts and text-processing applications, wasting 32 bits per character is beyond the acceptable level of redundancy. However, ISO 10646 is an important standard in that it establishes a single authority on the vast lands lying beyond Unicode, thus preventing the problem of incompatible multioctet encodings even before this problem could possibly emerge.

MIME

MIME stands for Multipurpose Internet Mail Extensions. It is a standard developed originally to extend the capabilities of electronic mail by allowing e-mail messages to include virtually any type of data, not only plain text. However, the mechanisms of MIME proved so useful and well designed that they are now used in many other fields, including HTML. The latest MIME specification can be found in RFCs 2045 through 2049. (See `http://ds.internic.net/rfc/rfc2045.txt`, `http://ds.internic.net/rfc/rfc2046.txt`, and so forth.)

The existing e-mail transport systems, such as Simple Mail Transfer Protocol (SMTP) and Post Office Protocol 3 (POP3), do not accept anything but plain text in the body of a message. This means that a message should contain only printable (noncontrol) characters of 7-bit ASCII and the lines should not exceed some reasonable length. To overcome this limitation, MIME introduces methods to convert binary data or texts in more-than-7-bit encodings into "mail safe" plain ASCII text.

Special MIME header fields are added to such messages to specify what conversion method was used (if any) and what was the original type of the data sent in the message. For text messages, along with other parameters, the character encoding (character set, or *charset*) of the message body is specified. This mechanism is important for us because it is used not only in e-mail messages but also in HTTP headers for HTML documents transferred over the network. The `charset` parameter is a part of the `Content-Type` header field and takes the following form:

```
Content-Type: text/html; charset=ISO-8859-1
```

Here, `text/html` is the standard identifier of the "HTML source" data type, and `ISO-8859-1` indicates the character encoding used by the text of the HTML document. Both these values are taken from the official registry of content data types, character sets, and other MIME-related classifiers maintained by IANA (Internet Assigned Numbers Authority).

It is this official registry that makes MIME so useful beyond the e-mail realm. For our purposes, it is especially important that MIME has developed a standard way of communicating the character encoding of a document. The list of registered MIME `charset` values can be obtained from `ftp://ftp.isi.edu/in-notes/iana/assignments/character-sets`.

HTML Character Set

Now that you are acquainted with various character encoding standards and the MIME-supplied method to indicate the standard in use, it's time to get to HTML and see how it is tweaked in version 4.0 to handle multilanguage data.

Document Character Set Versus External Character Encoding

First of all, an important distinction should be made. Chapter 3 examines SGML declaration for HTML and, in particular, its `CHARSET` section. This section defines the single *document character set* to be used by all conforming HTML documents.

On one hand, this makes the choice of the document character set fairly obvious: It should be itself international, which means Unicode or, better yet, ISO 10646. Here's how the SGML declaration for HTML 4.0 defines the document character set (see Chapter 3 for syntax explanations):

```
CHARSET
      BASESET  "ISO 646:1983//CHARSET
                International Reference Version
                (IRV)//ESC 2/5 4/0"
      DESCSET  0   9   UNUSED
               9   2   9
               11  2   UNUSED
               13  1   13
               14  18  UNUSED
               32  95  32
               127 1   UNUSED
      BASESET  "ISO Registration Number 176//CHARSET
                   ISO/IEC 10646-1:1993 UCS-2 with
                   implementation level 3//ESC 2/5 2/15 4/5"
      DESCSET  128 32    UNUSED
               160 65375 160
```

Here, ISO 10646 is employed in one of its transformation formats, namely the UCS-2, which is a 2-octet format identical to Unicode.

RFC 2070 takes a more thoroughgoing approach and bases the document character set on the canonical 4-octet form of ISO 10646 only, without a reference to ISO 646 (which is a subset of ISO 10646 anyway) and with the upper limit of the code space raised to as much as 2147483646.

On the other hand, however, it is unrealistic to expect and fairly unreasonable to require that all HTML authors and browser manufacturers switch to Unicode in the next couple months (or years, for that matter). So how can we get the benefits of Unicode without making everybody change over to it?

RFC 2070 explains that this quandary is resolved by differentiating the document character set from the *external character encoding* of the document. The external encoding is applied to the document when it's stored on a server and transferred through the network; this encoding can be arbitrary, provided that it is sufficient to encode the character repertoire of the document and that both server and user agent software are capable of handling it properly.

Upon receiving the document, the user agent software should convert it from external encoding to the document character set, so that further SGML processing and markup parsing is performed in this character set only. Before displaying the parsed document, the user agent can recode it once again, for example, to comply to the encoding supported by the operating system (in order to call its display services) or to match the encoding of fonts that will be used for output.

Converting the document from external encoding to the SGML-specified document character set is done for two obvious purposes. First, it is necessary to ensure that all characters that have special meaning in HTML, such as letters in element names and < and > characters, are correctly recognized (although it is unlikely, external encoding could remap some of these characters to other bit combinations).

And second, remember that users can invoke characters using character references (as discussed in Chapter 3), such as `©` for the copyright sign. For these references to be unambiguous and not require changes when the document is recoded from one character set to another, it is declared that these explicit references always refer to the document character set—that is, Unicode.

This means that, for example, to access the `CYRILLIC CAPITAL LETTER EF` via a character reference, you should use its Unicode code, which yields `Ф`, *regardless* of what character encoding you work in when creating your document. It doesn't matter whether you use KOI8-R 8-bit Cyrillic encoding in which this letter is coded 230 (decimal) or ISO 8859-1 that has no Cyrillic alphabet at all. A compliant HTML parser should always resolve character references in terms of the document character set (Unicode) simply because, at the time of HTML-specific processing, the document should be already converted into the document character set.

Here are some advantages of using Unicode as the document character set and separating it from external encoding of a document:

- This solution is backward compatible to the previous versions of HTML standard. Indeed, character references in, say, HTML 3.2 were supposed to refer to ISO 8859-1, which is a proper subset of Unicode. (Whether a code is 8 or 16 bits makes no difference in this case because character references use decimal values of codes where padding zeroes can be dropped.)

- The solution is also fairly flexible. International HTML authors can continue using the character sets that are widely supported by software and that minimize overhead for their languages. At the same time, they acquire the capability to directly access the entire character space of Unicode via character references.
- Finally, implementing the technique should not be too bothersome for browser manufacturers. The RFC 2070 standard does not even require user agents (browsers) to be able to display *any* Unicode character, but offers instead a number of workarounds for the cases when a browser cannot generate a glyph for a particular Unicode code (for example, displaying the hexadecimal code or some special icon in place of the character).

Specifying External Character Encoding

For an HTML browser to correctly translate the received document from external encoding into the document character set, it must know the external encoding beforehand. As of now, MIME is the only standard mechanism capable of communicating such information. As described earlier in this chapter, the `charset` parameter is included in the `Content-Type` field that must be a part of any HTTP header, that is, must precede any document sent via HTTP protocol (See Chapter 5, "Behind the Scenes: HTTP and URIs.") This field should also be included in the header of an e-mail message containing an HTML document. Currently, there is no way to indicate character encoding for an HTML document retrieved via FTP or from a local or distributed file system.

Common browsers such as Netscape and Internet Explorer recognize the `charset` parameter and try to switch to the requested character set before displaying the document (in Netscape Communicator 4, for example, you can open the Options|Document Encoding submenu to see the list of supported character sets). If no `charset` parameter is specified, ISO 8859-1 is assumed, and if it's not what the author planned for the document, the user will have to guess which encoding to switch to manually in order to read the document. (It is not unreasonable to claim that the very possibility of manually switching character sets in common browsers is to blame for the abundance of Web servers that never care to declare the character encoding of the documents they deliver.)

However, there's something more to character set negotiation. HTTP protocol allows a client program to list a number of character encodings it can handle, in the order of preference, right in an HTTP request using the `Accept-Charset` field. This enables the server to select the appropriate version of the document among those available or translate it to a requested character set on-the-fly. The standard declares that if no `Accept-Charset` value is given in the request, the user agent thereby guarantees that it can handle *any* character set. Unfortunately, the only browser (at this time) that allows a user to specify the `Accept-Charset` value to be inserted in HTTP requests is Lynx (whose Web site is at `http://lynx.browser.org`).

One more method to indicate the external character encoding of a document is by emulating the `Content-Type` header field in a `META` element. For this, you should place the following tag within the `HEAD` block of your HTML document:

```
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=KOI8-R">
```

if you need to specify that your document is in KOI8-R Cyrillic encoding. This is a handy choice for those who are unable or unwilling to change setup of the server that the document is stored on, but it has an obvious down side: Such a document, if automatically converted from one encoding to another, requires manually changing the `<META>` tag attributes. The `META` encoding indication is supported by most browsers, but beware of a pitfall: Contrary to the standard stating that the `charset` value in the HTTP header, if present, should override its `META` emulation, some browsers give preference to the `META`-supplied value.

Forms Internationalization

When browsing on the Web, you not only download textual information, but sometimes upload it as well using the forms mechanism of HTML. Naturally, this mechanism needs adjustments to allow character set negotiation of the data submitted from the user agent software to the server. This section covers the new features introduced to meet this requirement.

In HTML 4.0, the `FORM` tag is given an additional `ACCEPT-CHARSET` attribute that is similar to the `Accept-Charset` HTTP field mentioned in the preceding section. The main difference is that the `ACCEPT-CHARSET` attribute in HTML works the other way around, specifying what character encodings the server is able to *receive* from the user. The value of the `ACCEPT-CHARSET` attribute is a list of MIME identifiers for character encodings the server can handle, in order of preference; usually this list contains at least the external character encoding of the document itself.

A browser could make use of the value of the `ACCEPT-CHARSET` attribute in several ways:

- A browser must configure the text input areas so that the text being typed in would display using appropriate glyphs. This is a minimal level of support (it is implemented, for example, in Netscape Navigator 3.0, although this browser uses the main document encoding for this purpose instead of the `ACCEPT-CHARSET` attribute value), because it leaves the user with the main problem of how to input text properly. If the operating system does not support the encoding, it might be necessary to use a specialized keyboard driver or copy and paste previously converted text. In certain cases, an HTML author could provide a clue right in the document as to which encoding is accepted in a particular input field.
- Better yet, a browser must take into account the character encoding supported by the operating system and convert, if it is possible (that is, if the encoding supported by the system and the encoding accepted by the server have identical character repertoires) and necessary (that is, if these two encodings are not the same), the text typed in by

the user before sending it out. This makes the preceding item unnecessary because the operating system itself takes care of the proper display of characters in text input areas, provided that they use the native encoding of the system. This level of support is implemented in Microsoft Internet Explorer 4.0 and Netscape Communicator 4.0 (but again, both of these browsers ignore the `ACCEPT-CHARSET` value and consider the form charset the same as the document charset).

- RFC 2070 suggests that a browser may restrict the range of characters that can be input in the text area in accordance with the encoding specified. In my opinion, this is rather useless if not accompanied by one of the other two provisions.

The second part of the form internationalization problem is how to submit the form data along with the information about its encoding. For the first of the two submission methods, `POST`, MIME is helpful once again. It is possible to add the `charset` parameter to the `"Content-Type: application/x-www-form-urlencoded"` header field that precedes any data sent with the `POST` method.

However, RFC 2070 gives preference to another technique that uses the `"multipart/form-data"` content type that was proposed in RFC 1867 (which is outlined at `http://ds.internic.net/rfc/rfc1867.txt`) for form-based file uploads. (RFC 1867 provisions are also incorporated into HTML 4.0.) With this method, form data is not encapsulated in the form of a URL, and each name/value pair may have its own `charset` parameter attached. Currently, this technique is not supported by common browsers.

With the other form submission method, `GET`, data is encapsulated right in the URL that the browser submits to the server. In principle, URLs can contain any bit combinations, provided that they are encoded using the `%HH` notation. (See Chapter 5.) However, quoting RFC 2070, "text submitted from a form is composed of characters, not octets," and there's no easy way to incorporate information about the encoding of text data into an URL (other than by providing an additional input field that the user will need to manually set, which is pretty awkward).

RFC 2070 suggests that even with the `GET` method, user agent software could send the data in the body of the HTTP request instead of the URL, although currently no applications support this technique. Another solution with URLs might be using one of the special formats of ISO 10646; in particular, the UTF-8 format (which is outlined at `http://ds.internic.net/rfc/rfc2044.txt`) preserves all 7-bit ASCII characters as is and encodes any non-ASCII characters using only the octets with the most significant bit sets—for instance, those outside the ASCII range. This makes UTF-8 completely backward compatible with the URL syntax. Because ISO 10646 is a superset of all other character encodings, a string in such a format doesn't require any further `charset` specifications (provided, of course, that the server is aware of using UTF-8).

Real-World Character Sets Problems

In fact, differentiating the document character set from external encoding is nothing really new in HTML. Any numerical character references in a document conforming to HTML 3.2 or an earlier version must refer to the characters from the Latin-1 (ISO 8859-1) set, regardless of the external character set of the document. Unfortunately, this convention is ignored by many contemporary browsers, which leads to undesirable (although, admittedly, not too serious in the case of HTML 3.2 without internationalization extensions) consequences.

For instance, the KOI8-R character encoding as defined in RFC 1489 (which is outlined at `http://ds.internic.net/rfc/rfc1489.txt`) specifies code 191 (decimal) for the `COPYRIGHT SIGN` character. In ISO 8859-1, the same symbol is coded 169. Ideally, when a mnemonic entity `©` or character reference `&169;` (which is what `©` expands to, as defined by HTML DTD) is used in a KOI8-R document, the browser must resolve it with regard to ISO 8859-1 character set and display the copyright sign (for example, by accessing code position 191 in a KOI8-R font).

However, as most browsers are incapable of remembering anything about the ISO 8859-1 character set after being switched to KOI8-R or whatever external encoding is used for a document, an HTML author cannot rely any more on the table of Latin-1 mnemonic character entities. These entities or numeric character references are guaranteed to work only if the document itself is created (and viewed) in ISO 8859-1.

As a sort of a workaround, creators of several KOI8-R Cyrillic fonts for use on the Web chose to move the copyright sign from the standard-prescribed code 191 to the Latin-1-inspired 169. As Alan Flavell of CERN has put it, "Breaking your font in order to help a broken browser is a bad idea." It is obvious that, with the internationalized HTML gaining wide recognition, the problem could become more severe, as Unicode character references in conforming documents are much more likely to go out of sync with the external character encoding of a document.

In fact, support for nonstandard document encodings in browsers such as Netscape Navigator 3.0 is reduced to the capability to switch display fonts, in response to either the `charset` parameter in HTTP header or the user's having selected a command—and little else. As a result, Netscape Navigator 3.0 cannot display Russian texts in KOI8-R without KOI8-R Cyrillic fonts installed, even if it's working under a Russian version of Windows that provides Cyrillic fonts in Windows encoding.

There are still more problems related to document character encoding that many common browsers are unable to cope with, and that HTML authors should therefore beware of:

- Even when the text of a document is correctly displayed, its title, if it contains encoding-specific characters, may appear broken in the window title bar (apparently because the font used in window title bars is determined by the operating system, which may be completely unaware of the encoding of the document).

- ALT texts in place of inline images, as well as button labels in forms, might not display correctly if they contain encoding-specific characters (again, the reason is that many browsers use a system-provided font for these purposes).
- Text-oriented Java applets in Java-enabled browsers may have problems with displaying text in a nonstandard encoding.

Language Identification

Character set problems constitute only a part of the whole HTML internationalization issue. Almost equally important is the problem of *language identification* of a document. Lots of aspects of document presentation depend not only on the character set, but also on the language of the text.

For example, as I've mentioned before, the same ideographs are used in many Asian languages, so that in each language they are rendered by slightly different glyphs and quite different sounds of speech. Also, different languages using the same character set may differ greatly in respect to hyphenation, spacing, use of punctuation, and so on.

To this end, HTML 4.0 introduces the new LANG attribute, which can be used with most HTML elements to describe the language of the element contents. A "language" in this context is defined as "spoken (or written) by human beings for communication of information to other human beings; computer languages are explicitly excluded." Here is an example:

```
<P LANG="fr">Ce paragraphe est en Français</P>
```

The LANG attribute can take as a value a two-letter abbreviated *code* (or *tag*) of the language. A list of these codes is defined by ISO 639 standard (outlined at `ftp://dkuug.dk/i18n/ISO_639`); these codes should not be confused with country codes (for example, `uk` as a language code means Ukrainian, not United Kingdom).

Also, *extended identifiers* can be used to designate different dialects or writing systems of a language, identify the country in which it is used, and so forth. These extended identifiers are based on two-letter codes with the addition of *subtags* separated by a hyphen (-). Here are some examples:

- `en-US`: English language of the USA (two-letter subtags are always interpreted as country codes)
- `no-nynorsk`: Nynorsk variant of Norwegian
- `az-cyrillic`: Azerbaijani language written in Cyrillic script

A registry of such extended language identifiers is maintained by IANA (at the address `ftp://ftp.isi.edu/in-notes/iana/assignments/languages/`). All LANG values are case insensitive; their complete syntax is defined by RFC 1766 (found at `http://ds.internic.net/rfc/rfc1766.txt`). Another useful resource is the document at `http://domen.uninett.no/~hta/ietf/lang-chars.txt`, where most known languages are listed along with the character sets they use.

Language-Specific Presentation Markup

In a multilanguage environment, a need may arise to specify in HTML some aspects of text presentation, such as the writing direction (left to right or right to left), punctuation peculiarities, and so on. These aspects usually can be derived from the language of the text (as mentioned in the preceding section), but sometimes one might need to specify this information without specifying the language or to override the language default values. Also, some presentation aspects (such as quotation marks) require additional markup even if a language is specified.

RFC 2070 introduces and HTML 4.0 adopts a whole bouquet of new HTML elements, attributes, and entities for this sort of presentation markup. These new features are summarized in the following sections.

Writing Direction

While most Western languages are written from left to right, languages such as Arabic and Hebrew are written from right to left. In situations when such text is intermingled with the text of the opposite direction (resulting in a bidirectional, or *BIDI*, text), a special markup might be necessary to resolve ambiguity.

Unicode standard has a number of direction-related provisions. Each Unicode character is assigned the *bidirectional category* parameter that may take a number of different values, such as left-to-right, right-to-left, number separator, or neutral (for example, whitespace). Some characters (such as parentheses) are marked as *mirrored* depending on the text direction (in right-to-left text, an opening parenthesis should take the appearance of a closing one and vice versa).

To support this behavior, RFC 2070 introduces directional markup tools of three types. The first type consists of the left-to-right and right-to-left marks that behave exactly as zero-width spaces having corresponding direction properties. These marks are taken directly from Unicode inventory, so in HTML they are implemented as entities expanding into corresponding Unicode characters:

```
<!ENTITY lrm  CDATA "&#8206;" -- left-to-right mark -->
<!ENTITY rlm  CDATA "&#8207;" -- right-to-left mark -->
```

Direction marks can be used when, for example, a double quote (which doesn't have a direction of its own, but is not a mirrored character either) sits between a Latin and a Hebrew character; in this situation, the actual place of the quote depends on whether it is assumed to belong to the left-to-right or the right-to-left text stream. By placing an invisible direction mark (`‎` or `‏`) on one side of the quote, you can ensure that the quote is surrounded by characters of the same directionality, thereby resolving the ambiguity.

The second type of direction markup is represented by the new DIR attribute that, like LANG, can be used with nearly all HTML tags to indicate the writing direction of the text in the element's contents. Sometimes you might need to indicate the basic writing direction of a piece of text; also, explicit direction markup is critical when there are two or more levels of nested contra-directional text (for an example, refer to RFC 2070).

The two possible values for the DIR attribute are strings rtl (right-to-left) and ltr (left-to-right). As is the case with CSS attributes, you can use the DIR attribute when no element is normally discriminated by using the SPAN element as a sort of a neutral container. If the DIR attribute is omitted, the element inherits the writing direction of its parent element. The entire HTML document's default direction is left to right.

For brevity, definitions of the DIR and LANG attributes are packed into one parameter entity in the HTML 4.0 DTD:

```
<!ENTITY % i18n
 "lang        NAME       #IMPLIED  -- RFC 1766 language value --
  dir         (ltr|rtl)  #IMPLIED  -- default directionality --"
  >
```

Later, the %i18n; entity is added to the ATTLIST declarations for the majority of HTML elements.

Finally, the third type of direction markup is represented by the new phrase-level BDO element (BDO stands for bidirectional override). It is used when a mix of left-to-right and right-to-left characters should be displayed in a single direction, overriding the intrinsic directional properties of the characters. For the BDO element, DIR is the only obligatory attribute.

Cursive Joining Behavior

In some writing systems (most notably Arabic), a letter's glyph may be different depending on the context—that is, on whether the letter is preceded or followed by some other letters. Arabic letters are modeled after handwritten cursive prototypes, so a letter in a middle of a word is drawn joined to its neighbors and therefore might look quite different than it does when it is isolated.

As a rule, software capable of displaying Arabic handles these differences automatically. But sometimes it's necessary to control the joining behavior, for example, to exemplify a standalone letter with cursive joiners. For this, Unicode provides two special characters, both being invisible and having zero width, the first to force joining of adjacent characters where normally no joining would occur, and the second to prevent joining that would normally take place. HTML 4.0 provides means to access these characters in HTML via the ‍ and ‌ mnemonic character entities:

```
<!ENTITY zwnj CDATA "&#8204;" -- zero width non-joiner -->
<!ENTITY zwj  CDATA "&#8205;" -- zero width joiner     -->
```

Quotation Marks

A number of different styles exist to render quotation marks around short, in-text quotations. Although the English language always uses quotes "like this," French has « comme ça », and German prefers „wie hier". Moreover, nested comments sometimes use different styles; for example, Russian tradition uses French quotes (without separating spaces) on the upper level and German quotes for quotations within quotations. Finally, it is desirable to be able to render the same text with "rich" quotes in a graphics environment but with plain double quotes of 7-bit ASCII in text-mode browsers.

To account for these differences, HTML 4.0 offers the new phrase-level Q element whose content is surrounded by a pair of quotation marks rendered in accordance with the language of the text, the level of nesting, and the display capabilities available. Here is an example:

```
<P LANG="en">The English language always uses quotes <Q>like this</Q>,
French has <Q LANG="fr">comme &ccedil;a</Q>,
and German prefers <Q LANG="de">wie hier</Q>.</P>
```

Unfortunately, this solution is not backward compatible; most existing software will just ignore Q tags without displaying even the plain ASCII quotes, which can often damage the meaning of the text. Thus, practical use of Q elements is not encouraged until the majority of user agent software provides support for the feature.

Alignment and Hyphenation

Traditions of using text justification modes in other languages may be quite different from those of English. That is why RFC 2070 introduces the optional ALIGN attribute that can be used with most block-level elements (namely P, HR, H1 to H6, OL, UL, DIV, MENU, LI, BLOCKQUOTE, and ADDRESS) with the values of left, right, center, and justify. RFC 2070 suggests that the default ALIGN value for texts with left-to-right writing direction should be left, and for right-to-left texts, right.

This is a significant improvement over HTML 3.2, where the list of elements supporting this attribute is shorter (only DIV, H1 to H6, HR, TD, and P) and the value "justify" is not allowed. Judging from the DTD, HTML 4.0 takes a halfway approach: It adopts the justify option but leaves the list of elements accepting the ALIGN attribute the same as in HTML 3.2.

As for hyphenation, user agents are supposed to apply language-dependent rules to break words if this is necessary for proper display. In complex or critical cases, RFC 2070 suggests that HTML authors use the mnemonic entity that invokes the SOFT HYPHEN character present in Unicode as well as all of the ISO 8859 family and other character sets.

This invisible character marks the point where a word break can occur; if the word is indeed broken, the character is visualized as a usual hyphen (-) character. Unfortunately, common browsers do not implement this behavior; what's worse, both Netscape Navigator and Microsoft Internet Explorer *always* display a - in place of a soft hyphen, thus preventing you from using this character whatsoever.

For better hyphenation control, the new HYPH element was proposed that is capable of handling complex cases when breaking a word is accompanied by a change in its spelling (for example, the German word *backen* becomes *bak-ken* when hyphenated). However, the HYPH element was not included in either RFC 2070 or HTML 4.0.

Font Issues

Fonts lie on the boundary between visual presentation aspects of HTML documents and the problems of HTML internationalization. It's of little use to have HTML supporting Unicode if you cannot display its character repertoire (or at least, the part of Unicode that your document makes use of).

Of course the majority of users interested in non-English Web content already have some fonts installed on their systems. Often these fonts are supplied with localized versions of operating systems or other software, and implement encodings that are popular for a particular language. Common browsers such as Netscape Navigator allow using such fonts for viewing Web pages.

However, what we need is a method to ensure proper display of multilanguage data on any given system. One solution might be creating and distributing a free (or inexpensive) multilanguage font pack or, alternatively, a single font with Unicode character layout. A free Unicode font named Cyberbit is available from Bitstream at `http://www.bitstream.com/cyberbit.htm`.

The big down side to the single font solution is that the file size of a typical Unicode font is several megabytes (even without ideographs area). Probably the most practical solution for the Web today is a *glyph server*, a proxy server that substitutes inline bitmaps for all non-ASCII characters on the page you're viewing. Intermediation of such a server is a quick way to read a foreign-language page without any font headaches. Glyph servers now available include `http://www.lfw.org/shodouka/` (Japanese only) and `http://baka.aubg.bg`.

Recently, Microsoft and Adobe announced a merger of their proprietary font formats, True Type and Type 1, to create a new format called OpenType (which you can learn about at `http://www.microsoft.com/truetype/fontpack/opentype.htm`). Besides improved typographic control, this new format will make fonts readily available for many platforms with much less software support than before.

For the subject of this chapter, it is particularly important that the two companies have submitted a proposal to W3C aiming at developing a scheme to include with the page sent over the Web all the fonts (in OpenType format) necessary to view it. Microsoft intends to implement a preliminary draft of this specification in one of the forthcoming versions of Internet Explorer.

This development is likely to have great impact on HTML internationalization. On the one hand, the possibility to ensure proper display of any characters in a document on any system capable of handling outline fonts is a big plus. On the other hand, however, there are a number of dangerous pitfalls along this path.

First, being able to rely on supplied fonts, some HTML authors (as well as browser manufacturers) might really go wild in the area of character sets support. In fact, a character encoding of a document needs only to comply to that of the accompanying font, which makes nearly all

HTML internationalization provisions described in this chapter redundant—and, as a result, puts them in danger of death of neglect without software support. Of course fonts for Web distribution can use Unicode, but there's no guarantee that this will always be the case.

Second, the HTML font support puts additional emphasis on the visual presentation of a document, which is the aspect being already overemphasized with the proprietary HTML extensions now widely used. There are many documents on the Web today that are created without any concern for portability or SGML compliance, and it is not very likely that font embedding in HTML documents will ever improve the situation.

The future of HTML internationalization is quite obscure now. The standards surveyed in the chapter have just been finalized, and their implementations in software (even experimental) are few. Also, the big software companies are particularly known for poor support of official standards and pursuing their own proprietary extensions instead.

However, most national webs are now growing at a much greater rate than the Web as a whole, so that Internet-related products without at least some international support are likely to become rare very soon. In view of this, the package of RFC 2070 and related standards has, in addition to its thoughtful design, the clear advantage of being open, independent, and stable.

Summary

This chapter examines the HTML internationalization provisions introduced in RFC 2070 and then adopted in HTML 4.0. This part of the HTML standard allows you to create and serve Web content in any language and using any writing system, maximizing the chances of your audience getting your message and not a mess of indecipherable characters. In particular, you learned about the following topics:

- What a coded character set is, what character set standards exist, and how they are used
- What MIME is, and how it helps to communicate information about the character set of a document
- Why it is important to differentiate the HTML document character set and the external character encoding of an HTML document
- What pitfalls of common browsers you should beware of when working with international HTML documents
- What tools are available in HTML 4.0 to specify the language of any document fragment and its language-specific presentation parameters (writing direction, cursive joining, rendering of quotation marks, text alignment, hyphenation)
- How the font accessibility problem might affect the future of HTML internationalization

CHAPTER 40

Point/Counterpoint: Pure HTML

by John Jung

IN THIS CHAPTER

- Definition of Pure HTML and Extensions *814*
- Pros and Cons of Pure HTML *815*
- Pros and Cons of Extensions *817*
- Which One to Use? *820*
- Working Around Popular HTML Extensions *822*

With the rush of the Internet and the Web, everything seems to be operating at a faster pace. Now, people think that any program six months to a year old is considered obsolete. Nowhere is this attitude more prevalent than in the software industry. Internet-related companies, in particular, feel pressure to keep releasing newer products faster. Indicative of this situation is the rapid release of newer versions of Web browsers. With these new releases come more, and newer, HTML extensions. But what do all these extensions give us? Are they helping or hurting Web authors?

Definition of Pure HTML and Extensions

With all these HTML tags floating around, how can anyone tell which ones are extensions? Truth be told, there's no easy way. It's almost impossible to keep track of which tag is an extension and which one isn't. From a purely technical perspective, the best approach to answer the question is to ask the World Wide Web Consortium (W3C). W3C's Web page (`http://www.w3.org/`) makes the official specifications for HTML available. The problem is that the information is often presented in a very dry manner.

Fortunately, most people don't create Web pages by hand anymore, and there's a much more convenient solution: HTML editors. These programs used to be simple text editors that gave immediate access to HTML tags. They've evolved to the point that they keep track of which tags are extensions, and which aren't. Some of the better ones will even tell you which attributes within a tag are extensions. The problem is, extensions are created at a much faster rate than the programs are released.

Where Do Extensions Come From?

Who's making up all these extensions? Anyone can make up any HTML extension and call it his own. The problem isn't creating the extension, but getting Web browsers to support it. Most Web browsers simply ignore any tag they don't recognize. As a result, even if you were to create your own extension, you'd never be able to see it. Chances are, no one would ever be able to see it. So the real power of creating extensions rests with the companies that make Web browsers. Because they create the browsers, they can create any extension they want and provide immediate support for it.

Why Are Extensions Made?

You might wonder why so many extensions are being made. There are, generally, two reasons why companies are creating so many extensions. One is because they find a definite lack within HTML before they create their extension. Typically, extensions created out of this motive have been largely accepted. Usually, these extensions have offered better management of the HTML body itself than what was available. Sometimes these extensions are based on proposed standards, and so eventually get support.

But probably the main reason software developers create so many extensions is to be different. With the fast software development pace, each new version of the browser has to offer a new feature of some sort. The quickest way to accomplish this is to create an extension. It gives the browser an immediate feature that others lack, and it's easy. What needs to be done gets worked out inside the company. Best of all, if the extension gets very wide support, the company is hailed as a pioneer.

How Does a Tag Become Pure?

A pure HTML tag is one that has its behavior and syntax formally defined by the W3C. For a tag to become pure, its purpose and syntax must be proposed to the W3C. In turn, the W3C debates the merits of the proposed tag and determines whether the tag is viable. After the proposed tag has been scrutinized, the entire W3C decides whether it should become a standard tag.

Pros and Cons of Pure HTML

As mentioned, a pure HTML tag is one that is officially sanctioned by the W3C. But does a tag having this official sanction really offer any benefits? If extensions generally offer more capabilities than pure HTML, why not use them all the time? Also, are there any negatives to using pure tags?

Pros of Using Pure HTML

There are several reasons for using pure HTML tags in your Web page. Almost all of them benefit, in some way, by being part of a standard. Because they must go through a rigorous review, standard tags are generally slower to come by. This means that when the tag is finally released, you can be sure that it wasn't thrown in as just a gimmick. The tag didn't show up, just because a new release of the browser dictated its existence. Also, standard HTML tags tend to provide support for all systems and configurations. HTML was initially created to disseminate information, and as a result, tries to reach the widest audience.

Immediate Wide Support

One of the biggest benefits to using pure HTML tags in your Web page is immediate wide support. There is currently no known major Web browser company that isn't committed to the HTML standard. As a result, the standard tags almost guarantee that everybody will be able to see them. Part of the reason browser companies can support standard tags so quickly is because they are reviewed by the W3C, and the major browser vendors are members of the W3C. As a result, they know which tags are coming along and which ones will likely pass. Further, they can pass along such information to their own companies so that when a tag does become a standard HTML tag, everybody will have support for it.

Extensions, on the other hand, usually are created by a single company, so nobody else knows about them. Other Web browser companies find out about the extension the same time everybody else does. As a result, the other Web browser companies either have to wait for others to create extensions and support them or risk releasing a product that won't support the extensions of a different company. Thus, the other Web browser company either appears to lack initiative or is alienating potential customers.

Platform Independence

The W3C is an independent standards body; that is, no one person dictates how it should operate. This is another definite reason to use pure HTML, because such tags tend to be *platform independent*—you don't need to have one particular system, or environment, to use the tag. It might seem unlikely that a company would introduce a tag that it couldn't widely support. After all, wouldn't that company be shooting itself in the foot? Absolutely. The problem is that it has already happened.

The DYNSRC attribute is probably the best example of an extension that was entirely too platform dependent. This attribute, proposed by Microsoft, was to allow video playback using the <IMG> tag. Unfortunately, it would only support AVI files. Not surprisingly, AVI files are most often found on Windows-based operating systems. There were no provisions in the extension to support other multimedia file formats. As a result, few people used it, preferring, instead, to use the <EMBED> tag. Ironically, the <EMBED> tag was another extension—introduced by Netscape.

How many problems did this problem cause for Microsoft? It's hard to say, but they introduced the extension, so it was up to them to support it. Although they supported it fine on the Windows operating systems, that was the whole extent of it. Internet Explorer, the Microsoft browser, was very slow in migrating to other operating systems. Compare that with Netscape Navigator, which was, and still is, available on numerous platforms. Not only was the DYNSRC attribute a failure for Microsoft, it probably cost them some business. Because Microsoft's browser was only on one platform, it was not a solution for many large businesses. Netscape, however, rapidly became the de facto Web browser in many large companies.

Did one extension cause all these problems? Probably not, but it probably didn't help, either.

Feedback

Another advantage of using pure HTML tags in Web pages is that HTML tags tend to be more fleshed out. When a single company releases an extension, it might not have heard differing, or even helpful, opinions. As a result, if a shortcoming or flaw exists, it'll only be found after the extension is released. At least with pure HTML tags, there's the advantage of having more people look at the idea, increasing the likelihood that problems will be identified before the tag is released.

Cons of Using Pure HTML

Although pure HTML tags are widely supported, there are some drawbacks to using them exclusively. In particular, many of the standard HTML tags are, relatively speaking, quite tame. The primary purpose of pure HTML is to get the text out—to present the content—not necessarily how it looks. Extensions are designed to present information well. Recent developments from the W3C have helped in this regard, but it's still far behind extensions.

Creativity

People who make fun of committees often say that they eliminate creativity. Although this isn't really true, standards committees certainly do stifle creativity a bit. The problem is that standards bodies, such as the W3C, face very few pressures, which means that they have little motivation to create new tags. Contrast that against the Web browser companies such as Microsoft and Netscape. They almost have to create new HTML tags just to be different. They need to carve out some sort of niche so that people will choose their browser over the competition's.

Consider some popular extensions that have provided creativity that HTML tags didn't. Extensions such as the SIZE attribute for the <HR> tag have given us the capability to control the length and thickness of the horizontal rule. Extensions such as the ALIGN attribute for the <IMG> tag have also provided far better control over how text flows in relation to images. Similarly, the ability to specify different font sizes, which the SIZE attribute for the <FONT> tag accomplishes, was also largely overlooked by the W3C, and extensions filled the gap. Although many of these extensions have since gone on to become standards, they certainly didn't start out that way. They started out as extensions created by Netscape.

Availability

Another shortcoming to using only pure HTML is that it takes so long for pure tags to become available. Because pure HTML tags must go through a long evaluation process, it can take several months before a tag actually becomes standard. Extensions, on the other hand, can be released whenever a new Web browser is released. There's no need to wait for an approval process—the extension is created, and support can be instantaneous.

If we simply waited for the W3C to create new standards, many popular features wouldn't be supported. For example, nobody would be able to create any Web pages with frames until now, when the W3C made them officially available. All Java applets wouldn't have existed until about the beginning of 1997, when the <APPLET> and <PARAM> tags became standards. In short, Web pages in general would have a lot less spunk and appeal if everybody just used pure HTML.

Pros and Cons of Extensions

Anybody can create an HTML extension, so there are a lot of nonstandard tags out there. Extensions are a double-edged sword. On the one hand, they generally offer more features than pure HTML tags. On the other hand, there's no way of guaranteeing extensions will be supported by others. These are just some of the points discussed in the following sections.

Pros of Using Extensions

Because extensions can be created on-the-fly, they offer a number of advantages over pure HTML. Extensions can respond quickly to what people want, providing immediate access to cutting-edge technology. Although a standards committee is still debating the viability of the

technology, extensions are using it. Furthermore, some browser companies release extensions based on upcoming standards. This makes it possible to remain somewhat ahead of the game while ensuring that others will soon catch up. Finally, extensions also have the capability to direct a debated standard in a certain direction. Although it might not become a true standard, it comes close enough for most people.

Cutting-Edge Technology

One of the best aspects of using HTML extensions is that you can use them right away. There's no need to listen to people debating the merits of a tag, or the technology it's based on. Simply use the tag and forget about it. For example, two years ago, Java was an emerging technology with some promise. With its potential to liven up Web pages and the right marketing, it caught on quickly. This was helped along by Netscape's introduction of the `<APPLET>` extension, which enabled Web authors to link their Java applets with their Web pages.

However, while Netscape's extension was taking over, the W3C committee was largely quiet. Even during the HTML 3.0 DTD proposal, almost a year later, there was no mention of Java. That's because they were still taking a "wait and see" attitude with it. Although this is an understandable position to take, it certainly wasn't what Web authors were looking for. They wanted to spice up their Web pages immediately, and they wanted to use Java.

Future Standards

Another advantage of extensions is that they can be precursors to standards. Some browser companies are part of the W3C, and they know, in advance, what tags are coming. They also know how much support or opposition those tags are facing. As a result, they can often reasonably predict which tags will become standards and which ones won't. They can use this knowledge if they're releasing a new browser before the standard has been finalized. This, in turn, gives Web authors the assurance that they will eventually get support. Meanwhile, they can impress others and get their point across more effectively. It's just not a standard yet.

Directing a Standard

Using HTML extensions can influence pure HTML. Standards committees can argue all they want about what should be a standard. However, if an HTML extension is in tremendously wide use, there's little reason to oppose the extension. It's far better to make the extension a pure HTML tag than to have so many Web pages be nonstandard. There might be a desire to fix certain aspects of, or even enhance, the extension. However, popular extensions will probably become standards.

An example of an extension controlling where the standard goes is the `<FRAME>` tag. Frames were introduced years ago, with Netscape Navigator 2.0. Presently, with Netscape Communicator soon to be released, the W3C is still debating whether or not it should support frames, and if so, how best to implement them. Only now, after Netscape Communicator, has the W3C decided to make frames offical. The W3C frame syntax is nearly identical to Netscape's

extensions, which have been around for so many years. The only major difference between the W3C frames and Netscape's is the addition of inline frames. That is, frames that are contained within the body of the Web page. Did the grassroots support for `<FRAME>` help it gain stature? Possibly, but such support certainly didn't hurt the tag.

Cons of Using Extensions

Although HTML extensions can give you capabilities you've never had before, using them also has its share of pitfalls. Some problems are small, such as modified syntax. For example, an extension might have a proposed attribute that ultimately gets changed when the tag becomes a standard. Other problems are much more serious, such as not getting support anywhere else. Such is the case with poorly thought out ideas for HTML tags. Most of these problems deal with issues of support and maintenance of Web pages.

Change of Syntax

One problem with using extensions, particularly those expected to become standards, is that their syntax could change. Although a browser company might expect that its proposal will become standard, there are never any guarantees. As a result, if you follow the company too closely, you may create nonstandard Web pages.

An example of such a situation involves the `ALIGN` attribute of the `<IMG>` tag. Netscape proposed numerous additional values for it, such as `texttop` and `absmiddle`, and encouraged people to use them. However, the extensions were never approved. The W3C rejected those values, saying that the current ones were sufficient. Although this probably won't break any Web pages entirely, it might cause some inconvenience.

Lack of Support

Another problem facing people who use extensions is that some extensions never get supported. Typically, this happens when extensions are poorly thought out and created for the sake of being different. The history of HTML extensions is rife with examples of failed extensions. Interestingly, many of the failed extensions came from Microsoft. In an effort to be different from Netscape, Microsoft created numerous extensions for its Web browser. Almost all of them were rejected by the W3C.

Among the more prominent HTML extensions that Microsoft proposed is the `<BGSOUND>` tag. A sound file had to be specified, and when the page was accessed, the sound file would be played. There were numerous problems with this simplistic approach, not the least of which was that there was no guarantee that everybody could hear the sound. Further, there were few Web sites that even had a desire, let alone a need, to use such a tag. Additionally, Netscape's `<EMBED>` extension was already in wide use as a general-purpose method of including files. As a result, this tag, and the `DYNSRC` extension attribute for the `<IMG>` tag, both failed. There was no reason to use two separate tags when one was sufficient.

Which One to Use?

So which should you use, pure HTML or extensions? On the one hand, with pure HTML, you're pretty sure that your message will get out. The problem is that it might not look as nice as someone else's Web page. On the other hand, extensions can help you create those better looking, more dynamic pages. But if the extensions never get wide support, your message will never get out. There really is no easy answer because there are compelling arguments for both sides.

So what's the best solution for you? I don't know. Because I don't know you, I don't have a good answer just for you. The answer for the best approach to solving the HTML versus extensions debate lies entirely with you. If you're working on a corporate Web page and it is distributing public memos, you probably don't need to use extensions. Or, if you're trying to get a message out to as many people as possible, standard HTML tags are probably the best approach. On the other hand, if you have a personal Web page or one that is appropriate for your corporate image, you should probably opt for extensions. Also, you must remember that because HTML is constantly evolving, the answer is really a moving target. If you used a tag or attribute that was an extension, it might suddenly become a standard before you know it. Therefore, you would actually be using standard HTML tags to get high-impact Web pages. A number of different approaches to designing Web pages will fit most needs. Which one you choose depends on which one you feel is appropriate for your need.

Multiple Web Pages

One way to get around the problem is to have multiple Web pages. You can have a Web page full of extensions and all the cutting-edge features you want, but if you do, be sure to always have a similar page for those who might not have the latest browser. Some people don't have a thirst to always have the latest piece of software, and you have to accommodate them as much as possible. For them, you can create a Web page that uses few or no images, and uses nothing but pure HTML code. Just be sure to put a link to the generic Web page on your main Web page. It's best to put such a link either at the top or the bottom, of the page. You should also provide links to your more sophisticated Web page on the generic page. (See Figure 40.1.) This method makes it possible for all visitors to your site, regardless of what they're using, to enjoy it.

Common Extensions

Another way to get around the problem is to use only common extensions. Some extensions are in incredibly wide use even though they aren't standard. You can probably use such tags without worrying about breaking too many browsers. What makes an extension a common extension? That's hard to say because the HTML specifications are always changing.

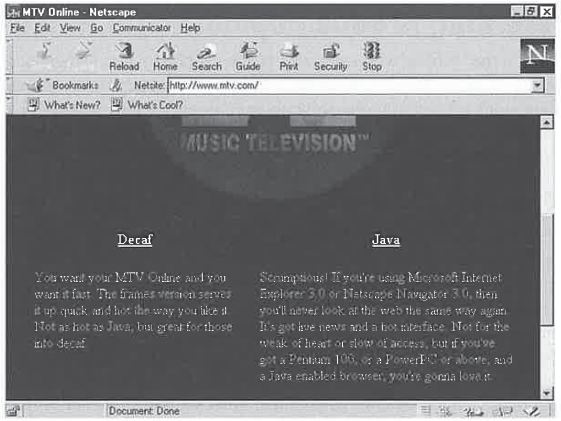


FIGURE 40.1.
MTV takes an interesting approach to distinguish between its various Web pages.

One way you can find common extensions is to create your Web page as normal and then, before you publish it, be sure to check how the page looks. Don't just use one browser, use two or three. Try to find a slightly older browser, or one that supports the most basic of features. Most browsers come with a free evaluation period, so you don't have to pay to try them out. Be sure that the information generally looks the same on all the browsers. Although the content might not appear the same, make sure that all the content is there for everybody to see.

Try Using Pure HTML

A sure way of getting around the problem of using extensions is to use only pure HTML. If it sounds like I'm discouraging the use of extensions, that's because I am. Too often, when people aren't sure how to make a page look just right, they run straight for extensions. They immediately jump to JavaScript or Java applets and, consequently, alienate some users.

For various reasons, some people might not be able to access these extensions. For example, security warnings have been issued about possibly dangerous Java and JavaScript Web pages. You can be sure that some people have completely disabled Java and JavaScript on their modern browsers. And not everybody is using Netscape or Internet Explorer. There is still a handful of users with text-only browsers.

Therefore, I do recommend that you try to code your page in pure HTML. Although you might have to make a few compromises, such as not having a message scroll across the screen, it might not be so bad. You might actually be pleasantly surprised at what you can do simply by using tables and animated GIFs. They might not be as devastatingly impressive as extensions, but they can look pretty good just the same.

Working Around Popular HTML Extensions

Part of the problem with extensions is that people simply use them because they're there. Extensions, while often quite useful, aren't the only methods of solving a problem. There are numerous creative ways that you can accomplish a similar look using only pure HTML tags. This is worth the effort because if you can find the right combination that works, you can reach more people.

Frames

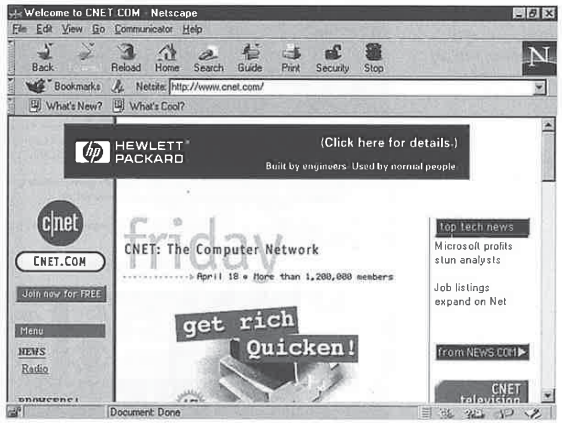
Frames is one of the most widely used extensions around. People like to use them because they give strict control over the look of an entire window. You can have a menu on the left side of the window and a content window on the right. When the user clicks on an item in the left, the right frame automatically updates. It looks cool and makes it easy for people to navigate around. Best of all, frames can give an immediate look and feel to the site, such as using a frame for a navigation tool. Additionally, it's not that much work to provide support for frames. After the `<FRAMESET>` layout is complete, just make sure all the links point to the right `TARGET`. The problem is, many older browsers either don't support frames or support them poorly.

Although frames offer such a nice method of navigation, there are easier alternatives—not as snazzy looking as frames, but reasonable alternatives, nonetheless. One approach is to have a common page layout. This gives your site a common look and feel without frames. If you absolutely must have a common navigation tool, you can use tables. Have a row or column where the navigation buttons reside. (See Figure 40.2.) Although this requires you to duplicate some HTML code, it does make your site very accessible.

Multimedia Files

I know what you're thinking, "Multimedia files aren't HTML tags!" You're right. Unfortunately, people use the `<EMBED>` extension to put a multimedia file on their sites. The file may be anything from a sound file that plays in the background to a full-fledged multimedia file. Although some file types need an extension to properly function, such as for Shockwave files, QuickTime movies, and other plug-ins, not all of them do. For what most people do with multimedia files, it's an unnecessary risk of possibly alienating users. There's also the problem that sufficiently large multimedia files could take a while to retrieve—so long that some people might give up on your site. In fact, some corporate sites that initially had multimedia files as part of their Web pages soon took them out. Chances are, people complained either because of inaccessibility or download time of the files.

FIGURE 40.2.
CNET, a popular Web site, uses tables instead of frames for navigation.



The best way to get around this problem is to simply not use multimedia files. I know you're going to complain, claiming that they're essential to your site. But are they really? Do you really need the music from Rocky playing when someone first accesses your site? Is it worth losing people who don't have sound capability or the latest browser? If you really need to have multimedia files in your Web page, there are two good alternatives for you.

One alternative is the multiple Web page solution mentioned earlier. You can have a list of available Web pages with duplicated contents. Furthermore, you can list the desired configuration for accessing each page. Users would simply click the one that best suits their conditions. Another alternative is to provide links to the multimedia files themselves. (See Figure 40.3.) Although your Web page won't look nearly as dynamic or exciting as it could, you will be reaching more people. With this solution anybody, with or without a recent browser, can experience your Web page. Additionally, with downloadable multimedia files, you also have another means of advertising. You might be able to put in a simple ad for your site or company at the beginning or end of a file.

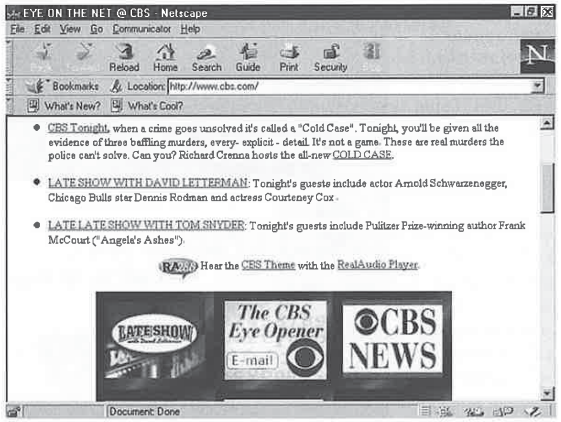


FIGURE 40.3.
Unlike some of its competitors, CBS lets you download its theme music.

Summary

This chapter explains what pure HTML and HTML extensions are and covers the benefits and drawbacks of using them. Ultimately, of course, the decision about which HTML tags to use is up to you. But perhaps the best piece of advice is to not work on the bleeding edge. HTML extensions can look really neat and give cool effects, but you won't be reaching the widest audience. Because not everyone will automatically upgrade to the latest version of a Web browser, you can't be sure how wide your audience will be. However, if you don't care about reaching everybody, then go nuts. Use all the extensions you want, and don't worry about anyone who complains about not being able to see your Web page.

CHAPTER 41

HTML Beyond the Web

by Dick Oliver

IN THIS CHAPTER

- From Calculators to Communicators *827*
- From the Desktop to the Conference Table *828*
- HTML as the New User Interface *829*
- The Digital Media Revolution *830*
- Unity in Diversity *831*
- HTML as a Programming Language *833*
- HTML Applicaitons of the Future *834*
- What You Can Do Today to Be Ready for Tomorrow *836*

If you've read even a few chapters from Parts III, IV, and V of this book, you have probably gotten the idea that HTML isn't just for Web pages any more. But you may not yet see how all these new extensions and technologies fit together into a cohesive future for HTML. This chapter steps back a bit and offers a big-picture view of where HTML is heading and the new role of HTML in today's high-tech world. You'll find out why most of the HTML pages you create in your lifetime will probably not be Web pages, and why the intimate familiarity with HTML you have gained from this book will be one of the most important (and profitable) skills that anyone can have in the next few years.

To pull that big picture together, first consider some of the pieces that you've seen in previous chapters:

- In Chapter 22, "Dynamic HTML," you learned that all future versions of the Microsoft Windows operating system will use HTML as a fundamental part of the user interface.
- In Chapter 17, "Introducing Cougar," and Chapter 38, "The Emergence of Extensible Markup Language," you discovered that HTML is being extended to give you precise control over the appearance and functionality of virtually any textual and graphical information.
- In Part V, "Associated Technologies and Programming Languages," you saw how all major programming languages, interactive media, and database formats can also be seamlessly integrated with HTML.
- And in Chapter 39, "Internationalizing the HTML Character Set and Language Tags," you found out that HTML can be used to communicate in the native script of any human language in the world.

New data security standards are also making it practical to carry out financial and other sensitive transactions with HTML, and the proposed Platform for Internet Content Selection (PICS) standard provides a highly flexible way for the content of any HTML page to be rated according to any criteria that a rating authority or individual user might select. Restricting access to adult-oriented or confidential information is one of many applications.

All this adds up to a very near future where HTML will without a doubt play a central role—it may even be accurate to say *the* central role—in the display and exchange of almost all information across all computers and computer networks on Earth. If this sounds important, well, it is. But this chapter makes a case that HTML will have an even more important role than that to play. To understand how that can be so, we need to take another step back to see an even bigger picture: the changing role of the computer itself in our society.

From Calculators to Communicators

The computer was once considered a device for accounting and number crunching. Then it evolved into a device for "crunching" all types of information, from words and numbers to graphics and sounds. Today and tomorrow, the computer is above all a communications device; its primary use is the transmission of information between people.

Two major trends are dragging HTML out of its niche as a Web page language:

- People from a wider variety of social, economic, and educational backgrounds are using information technology on a daily basis. Digital information is playing a greater role in all our lives.
- Physical location is no longer the primary factor in the market that most businesses serve. To find customers and do business in a global marketplace, even the smallest companies need to be able to gather and distribute many different types of information.

These large social trends may not seem, at first glance, to have much to do with HTML, except that they cause people and businesses to access the Web more often. But they have directly led to two corresponding trends in technology:

- Entry-level information technology is getting—and must continue to get—both easier to use and less expensive.
- The distinctions between an individual computer, a local network, and the global Internet are blurring.

HTML is proving to be the dream technology that is enabling both of these trends to accelerate faster than anyone anticipated.

In many workplaces today, these two trends are already well established. You can use a computer to access business information every day without knowing much more than how to click links and scroll down through long pages. And you can do so without being at all sure which information is coming from your computer, which is coming from the server down the hall, and which is coming from other servers, perhaps thousands of miles away.

The direction that these trends are taking us is also already clear: Users who become used to seeing highly readable and attractive pages of information (such as Figure 41.1) on their computer screens will lose the tiny bit of tolerance they have left for cryptic icons, unadorned text messages, and idiosyncratic menu mazes. (See Figure 41.2.) They will soon expect their computer screens to *always* be as easy to read and interact with as the Web.

Those who make their millions supplying computer software are well aware of that expectation and are expending an unprecedented amount of research and development effort toward fulfilling it. Along the way, the central metaphor for interacting with computers has changed from the "window" of the 1980s "desktop" to the "page" of the 1990s "World Wide Web." It is in the process of changing even more radically, to a metaphor focused on direct communication instead of old-fashioned paper shuffling.