Designed for Microsoft® Windows® 95

CD-ROM Included

Hot Tools for Cool Web Sites

HTML In Action

- Advanced techniques
- ActiveX™ tools
- CGI scripting hooks
- Java

Bruce Morris

Microsoft® Press

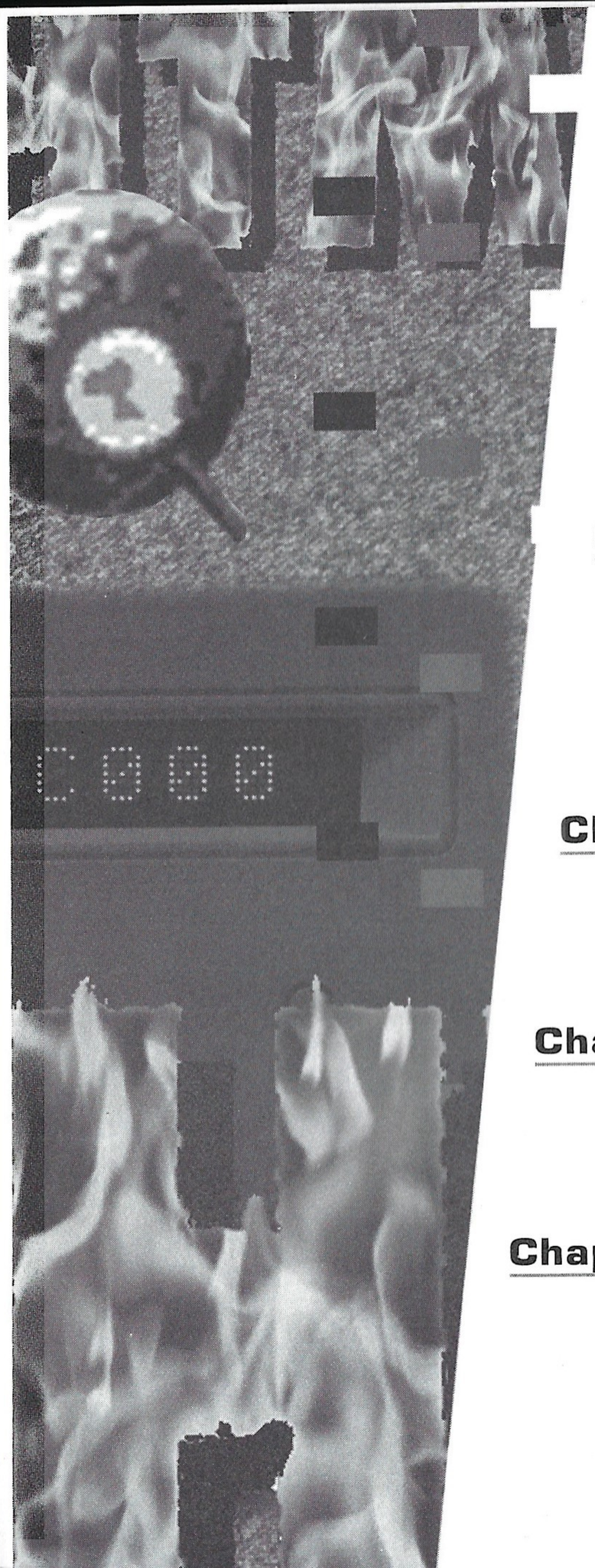
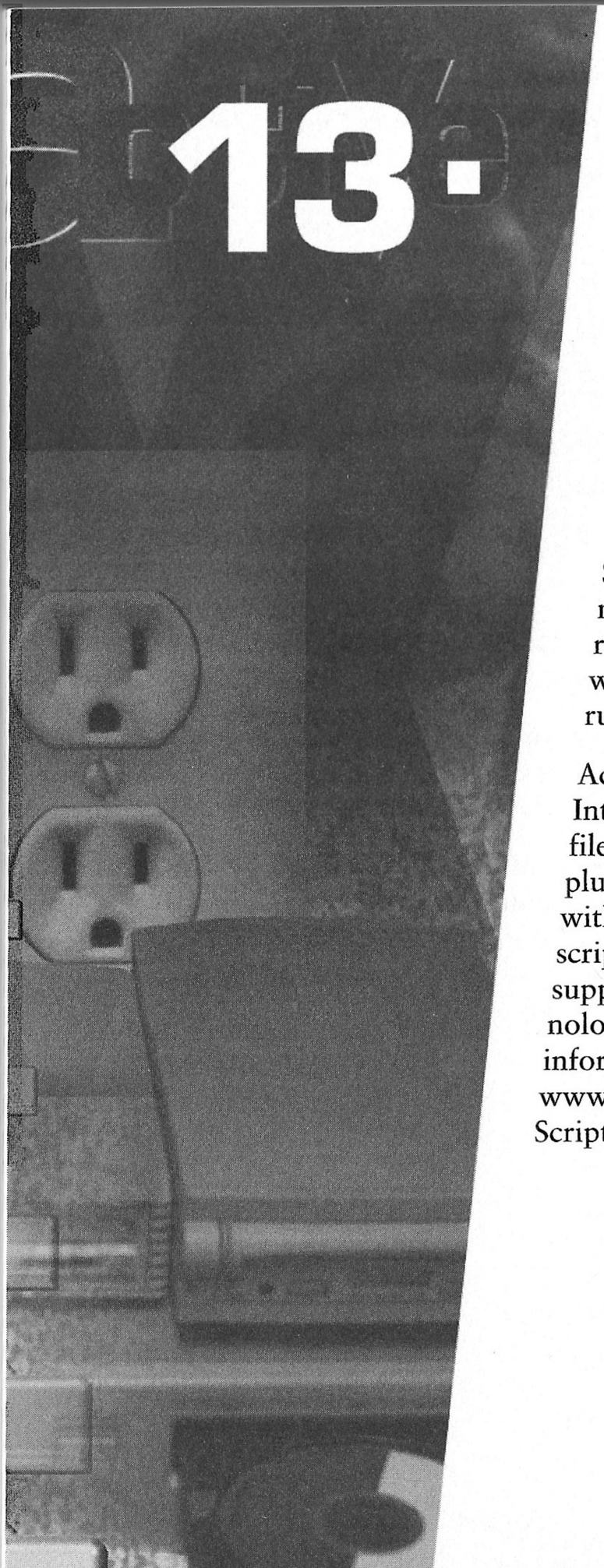


TABLE OF CONTENTS



13 ACTIVEX

by Lin A. Laurie

How do you harness the energy and innovation that abound on the World Wide Web? How do you add animation, sound, video, interactivity, and customized content to your Web pages? Microsoft's answer to this question is in using ActiveX; Microsoft Visual Basic, Scripting Edition (VB Script); and Internet Explorer 3.0.

So, what do you get when you take OLE (an object-oriented technology designed for creating, managing, and accessing object-based components across process and machine boundaries) and repackage it to download automatically from a server, function device-independently, operate without requiring that the object's authoring program be installed on each user's machine, and run in HTML code? In a nutshell, that's ActiveX.

ActiveX is an API built on Win32 and OLE that lets developers enable their applications for the Internet. Supported by Internet Explorer 3.0, ActiveX adapts Visual Basic–like objects (in .OCX files) to the Web and throws in its own Internet Server API (ISAPI). Using OLE Automation and a pluggable scripting engine such as VB Script, you can add interactive features to your Web pages with just a little programming. (In this usage, "pluggable" means you can insert the code for the script inside your HTML code and identify the code, and it will run as if it were one of the codes supported by your browser.) This chapter is for the more advanced Web page developer. The technology covered is so new that it is still evolving as this book is being printed. For the most current information on ActiveX, VB Script, and Internet Explorer, access Microsoft's Web site at http://www.microsoft.com. This chapter covers ActiveX technology, looks at how to implement it using VB Script and HTML, and discusses some of the ActiveX objects and controls currently available.

What ActiveX Means to You

Imagine a future where depositories of independent and innovative objects (in .OCX files) exist, scattered across the Internet. Various independent object-authoring vendors own these depositories and sell an inventory of pluggable objects from them over the net as they are needed. Let's say that you want to include an interactive order-processing capability in your Web page or add a killer 3-D movie to your application. You would purchase the ActiveX program controls and their associated usage rights to either function from one of these object depositories. You would then write the HTML and VB Script code that would transfer variables to and from the ActiveX object and then use the HTML code to create your own Web page. Then you would install the HTML program for your Web page on a network server, and it would run whenever a user clicked on the hyperlink that activated the ActiveX object.

For instance, let's say that your 3-D movie object was created using Microsoft's SoftImage (a 3-D authoring tool). But your users don't own a copy of SoftImage. You could embed the VB Script code to call the object (3-D movie) in your HTML code, install the object on the network server that has your Web page with the code that activates the object, enable it to download automatically to the user's machine (unless it already exists there from an earlier use), and then have it execute, without requiring that SoftImage be located anywhere (on the server or the client).

Or, for an online ordering system, you would purchase the ActiveX object from the object-authoring vendor. Along with the object, you'd receive a list of parameters (data) that the object needs in order to perform its function (order processing). You'd need to use VB Script or another scripting tool to pass the needed parameters to the object. You could validate form data, automatically generate custom Web pages, and even write games as objects that run using VB Script to control them. The object would use the data to output information back to the Web page. Information such as the item amount, order total, tax, and so on could be calculated by the object and transferred back to the VB Script, which would then pass it to the HTML code, which would then display the calculated information on a Web page. The scripting language would take care of the input and output tasks to the object, passing them back to the HTML code, which would display them on the Web page, while the object would handle the calculations and any other internal processing needed to calculate the fields you'd need for valid order processing. Since most accounting

applications, such as order processing, can only be done a finite number of ways, these applications would make good use of the pluggable-object technology and leave developers free to become involved in the more creative aspects of application development such as adding multimedia interactive experiences to their Web pages with Virtual Reality Modeling Language (VRML).

As a developer, you'll find abundant opportunities to purchase from these futuristic code depositories controls (custom sliders, list boxes, dialog boxes, buttons, and so on) that you can plug into ActiveX objects created in any number of languages. (You can create ActiveX objects in C++, or Java applets in JavaScript, for instance.) Your objects can have a unique or a uniform look, depending on the pluggable controls you use when you develop them. Then, when plugged into an HTML-coded Web page and viewed with Internet Explorer 3.0 or another browser that supports them, they can be used in ways you might not have imagined when creating them.

Developing controls and objects is a business opportunity you might consider. In the future, expect to see an increased need for these pluggable components (controls and objects). Since coding these components is more labor-intensive than plugging them into HTML code, this is where the challenge will exist and your innovation can truly be called into play.

Internet Explorer 3.0 and ActiveX

Internet Explorer 3.0 has a key role to play because it supports ActiveX, JavaScript, VB Script, and other pluggable scripting engines. By enhancing your Web pages for Internet Explorer 3.0, you can treat your users to state-of-the-art interactivity.

Internet Explorer 3.0 has an open and extensible architecture so that everyone can plug their components into it, regardless of the language, tool, data format, or protocol used to create them. It accomplishes this plugability using custom controls made available through ActiveX.

Microsoft has big plans for Internet Explorer 3.0—incorporating it into Windows 95 and using it to replace its Windows Explorer as a navigation tool. The Microsoft concept is to use Internet Explorer 3.0 as an interface for the Internet, for a corporate intranet, and for the contents of your hard drive.

Microsoft is also adding support for the HTML extension for frames. Frames allow Web page developers to segment the screen real estate into functionally independent panes. Internet Explorer 3.0 brings some new standards to the arena by introducing its own HTML extensions, including a FACE parameter for the existing <FONT> tag to let Web designers control the look of their Web page typography by specifying one or more alternative fonts.

Plug-ins, ActiveX, and Java are three of a kind—interactivity tools. Microsoft plans to provide support for all three of these tools by disguising Java applets and plug-ins (with wrappers) to imitate ActiveX controls.

Internet Component Download

The mechanism for downloading and installing ActiveX objects on an Internet user's system is called Internet Component Download technology and is used internally by Microsoft Internet Explorer to handle ActiveX components inserted in HTML pages.

TIP

ALTHOUGH ACTIVEX IS INSTALLED WITH INTERNET EXPLORER 3.0, ALL OF ITS CONTROLS ARE NOT.

The following description of this technology is not inclusive, but it will help you understand what happens on both the server and the client when a user clicks on a Web page that contains an ActiveX component.

Microsoft has created a new system API named GoGetClassObjectFromURL to safely download ActiveX components. Other safety needs can be met by using the WinVerifyTrust API or by using a high-level "Setup" ActiveX control. Currently the Internet Component Download will not download anything except ActiveX objects. It also downloads components into a permanent storage directory, where they sit. However, in a future release Microsoft plans to convert this storage directory to a temporary cache that discards unpopular components (when you've closed the browser).

To package an ActiveX object for downloading, you must use the <OBJECT> tag in HTML and choose one of three packaging schemes: Portable Executables (as .OCX or .DLL files), Cabinet files (as .CAB files), or stand-alone (as .INF files).

The Portable Executable Packaging Scheme

This approach uses a single executable file (such as an .OCX or a .DLL) that is downloaded, installed, and registered all at once. It is the simplest way to package a single-file ActiveX object. This packaging scheme does not use file compression, and it is not platform-independent except when used with the HTTP control.

The Cabinet Files Packaging Scheme

This approach uses a file that can contain one or more files (thus it is called a cabinet), all of which are downloaded together in a compressed cabinet file. It also is not platform-independent except when it is used with the HTTP control.

The Stand-Alone .INF Files Packaging Scheme

This approach uses a file that contains the specifications for other files that need to be downloaded and set up in order for an .OCX to run. The .INF file contains URLs pointing to files that should be downloaded. This packaging scheme can provide platform independence by listing files for various platforms, and it does not require the HTTP control. You should include an .INF file in your cabinet file specifications.

TIP

WHEN YOU EMBED AN ACTIVEX CONTROL ON A PAGE, USE THE **OBJECT** ATTRIBUTE TO SPECIFY AN ALTERNATIVE OBJECT TO DISPLAY WHEN THE CONTROL IS NOT FOUND.

Creating a Package

After defining your package scheme, you will need to create your package using a script or one of the tools in the Microsoft ActiveX Developer's Kit (MADK).

The download process is complicated, but it consists of the following general steps:

1. Download the necessary files (.CAB, .INF, or executable) using URL monikers.
2. Call the WinVerifyTrust API to ensure that all downloaded files are safe to install. Part of this API will ask the user whether he or she wants to install the files before proceeding to install them on the user's computer.

TIP

DON'T ASSUME THAT EVERYONE WHO VIEWS YOUR PAGE WILL HAVE THE ACTIVEX CONTROL ON THEIR MACHINES. USE THE **CODEBASE** ATTRIBUTE TO POINT TO A LOCATION WHERE THE ACTIVEX CONTROL CAN BE FOUND.

3. Use the /regserver command-line argument for .EXE files and DLLRegisterServer() for other executables (such as .DLL and .OCX files) to self-register all ActiveX components.
4. Add registery entries to track downloaded code.
5. Call GoGetClassObjectFromURL to get the *rclsid* (Class ID) of the object to be installed.
6. Store the downloaded files in the windows\system\occache directory. Currently the path is hard-coded. In future releases of Internet Explorer, you will be able to use a registry setting or a Control Panel applet to choose the directory. Also, note that at the current time, downloads are stored permanently. In future releases, you will be able to specify which components should be deleted at the end of your Web session or when you turn off your computer.

Downloading ActiveX code from the Internet also is complex. This section can provide only an overview of the process that will give you a general understanding of what occurs. For additional information about this process, review the most recent postings on this subject at http://www.microsoft.com or on MSN (The Microsoft Network).

Security and ActiveX

Although .OCXs let users enjoy and interact with their computers flexibly and powerfully, they also provide a method of destroying data on your server. Unfortunately, there are people just waiting to write the next great virus using .OCX technology. Fortunately, Microsoft has taken steps to help ensure that what you're downloading is safe.

Because .OCXs can access your hardware directly, an .OCX control has the potential to do nasty things, such as formatting your hard drive. To help prevent such malicious vandalism, Microsoft is establishing a certification program that will enable you to have your .OCX certified as safe.

Before you download an .OCX that has been certified, you'll be prompted with a message saying something like, "The application you are about to download has been certified by Vendor ABC and is compliant with the Microsoft Certification guidelines. Continue?" Conversely, when you try to download an .OCX control that is not certified, a warning message will appear, telling you that what you are about to do may be unsafe. You will be able to cancel before any damage can occur.

OLE Document Objects

Although HTML code provides a single standard for documents and navigation support, it does not provide the ability to publish on the Web in any format, regardless of tools, a document that is visible to a user who doesn't own the application used to author it. Enter ActiveX documents, objects that let you publish documents in your favorite format and ensure that the viewing code for the object is available to your reader. Using hyperlinks inside your ActiveX documents, you can add navigation capability as well.

In technical terms, OLE Document Objects are a set of OLE interfaces built on top of the standard OLE in-place activation interfaces that are used in OLE objects today. A document server owns the entire client area of the document container in which it is running. What this means in layman's terms is that when an ActiveX-enabled browser such as Internet Explorer 3.0 tries to view one of these documents, it downloads the document's server container to the user's computer (after asking permission) and runs it. The container then takes over the browser's entire client area and displays the file. To end users, the process is seamless, and they continue to use the same tool to view the document that they used to browse the Internet.

OLE Scripting

Microsoft's goal from the very beginning of OLE development has been to allow software developers to become more productive by creating reusable software modules. With ActiveX, Microsoft has expanded that goal from including OLE technology across the enterprise to the largest network of all—the Internet.

With OLE Scripting, you can call reusable software modules written in any supported language and not need to learn a lot about programming. Let's say you want to run a mapping program to map a URL to an Automation object (components that allow you to extend the functionality of your script but don't require a Windows interface) with thread-safe communication to any automation server with simple syntax. Thread-safe communication is a safe instance of execution of an object, meaning that while one user's object is being executed by the server, a second client cannot interrupt the execution to use the same object. When multithreading occurs, separate instances of the object are created, each with a separate thread to the object from the client. For

example, suppose you had an Automation object named myobject, a method named mymethod, and two parameters, param1 and param2. The following code demonstrates how you would execute the method through a URL name to return data in the form of a particular Web page. Because it is going through an Internet information server (IIS), the example uses the HTTP protocol and extends it by simply pointing the URL at the object. The example shows the simplicity of communicating with your OLE object using OLE Scripting.

```
http://machine/path/myobject.mymethod?param1=data1&param2=data2
```

You can use any language to create OLE applications so long as the language complies with the OLE Scripting interface. The OLE Scripting specification has the following characteristics:

- Applications can easily locate a script engine and run a script, but they don't need to know anything about the syntax, grammar, and execution model of the script itself.
- Events that are triggered from components within the application (such as OLE controls) are caught by the script engine and can have scripts associated with them.
- The script engine runs the application's space as a service, not the other way around.
- Script engine overhead should be negligible.
- The script engine should be able to customize base application objects. (Brett Morrison, *PPI Technology White Paper: OLE, Java, or Both?* [1996 Pinnacle Publishing, Inc.] p. 10.)

VB Script and ActiveX

The VB Script language is an upwardly compatible subset of Visual Basic that adds scripting capabilities to Web pages viewed with Internet Explorer 3.0 or other browsers that support the same standards. It will run in HTML without requiring the host to create or support other integration code, as opposed to languages such as CGI, which require integration code in order to execute. In technical terms, we say it is a component that is used by a host such as a browser or other application without requiring the presence of other scripting components.

VB Script is a safe language, having had any "unsafe" operations removed—that is, it does not include Visual Basic development components such as an editor, a debugger, a project manager, or a source code controller. It can be used on any hardware platform as a batch-automation language to provide scripting capability.

The classic example of VB Script uses HTML's <INSERT> tag—currently under review by the World Wide Web Consortium for inclusion into the official HTML specification. <INSERT> allows you to embed any kind of object in a browser. In the following example, we insert an .OCX control:

```
<INSERT>
    CLSID = {"ACME OCX control"}
    OLEcontrol.forecolor = true
    OLEcontrol.animate
    javaapplet.forecolor = olecontrol.forecolor
<\INSERT>
```

This example shows how easy it is to write a few lines of VB Script code that can communicate with your external program. The <INSERT> tag might seem to be a replacement for the IMG tag, but IMG is still useful for backward compatibility because the parameters to it are dynamic instead of being built into the definition. For example, the following code inserts an .AVI file if the browser supports that format; otherwise, it uses the backward-compatible IMG tag to insert a compatible image into a Web page:

```
<INSERT DATA=interview.avi
TYPE="application/avi">
<PARAM NAME=LOOP VALUE=INFINITE>
<IMG SRC="still.gif" ALT="Interview with Customer">
</INSERT>
```

Here's an example of an <INSERT> tag that loads a Java applet by referencing a Java class:

```
<INSERT CODE="BounceItem.class" WIDTH=500 HEIGHT=300>
</INSERT>
```

You can also use the <INSERT> tag by specifying an application name in its CLASSID section. Although we have become accustomed to thinking of CLASSID as an integer, it can actually be anything. The browser can then interpret CLASSID to locate the code for the class by interrogating the OLE registry or downloading it somehow by using the URL name:

```
<INSERT
CLASSID="Word.Basic"
CODE="http://ole.acme.com/apps/Word"
WIDTH=400
HEIGHT=75
ALIGN=BASELINE
>
<PARAM NAME=TEXT VALUE="This is the OLE Viewer">
</INSERT>
```

The key advantage of this technology is that the browser is only the front end of the application. In other words, there aren't any particular hoops to jump through for Internet applications. A well-designed OLE application doesn't care who it's communicating with. When you build an application that uses Internet Explorer (or any other browser supporting ActiveX) as its user interface, you're building an application that can have Visual Basic, Visual C++, Access, or any other OLE and Visual Basic–enabled hosting development tool as its front end.

ActiveX Controls

The controls described in this section are the only ones currently supported by ActiveX and released by Microsoft with the ActiveX SDK. At some future point, more controls might be included in the package. In addition, these controls are included in the Internet ActiveX Control Pack and are designed to assist you in creating powerful Internet applications based on the current base Internet protocols.

File Transfer Protocol (Client)

File Transfer Protocol (FTP) is a popular standard network protocol that is used to transfer files over networks. The FTP Client control allows developers to use Microsoft programs such as Access, Visual Basic, and Visual FoxPro to easily implement FTP in their applications.

HTML Control (Client)

The HTML control provides parsing and layout of HTML data as well as a scrollable view of the selected page.

HTTP Control (Client)

The HTTP (Hypertext Transport Protocol) control implements the HTTP client based on the HTTP specification.

NNTP Control (Client)

The NNTP (Network News Transfer) client control allows you to connect to a news server, retrieve a list of available newsgroups and their descriptions, enter a newsgroup, get lists of articles, and get any article. This control implements the basic client NNTP.

POP (Post Office Protocol) Control (Client)

The POP control provides access to Internet mail servers using the POP3 protocol. Internet mail developers and system integrators can use this control to retrieve mail from UNIX servers and other servers supporting the POP3 protocol.

SMTP (Simple Mail Transfer Protocol) Control (Client)

The SMTP control provides a reusable component that gives applications access to SMTP mail servers and mail posting capabilities.

WinSock TCP (Transmission Control Protocol) Control (Client and Server)

The WinSock TCP control is a connection-based control and is analogous to a telephone on which the user must establish a connection before proceeding.

WinSock UDP (User Datagram Protocol) Control (Client and Server)

The WinSock UDP control is a connectionless control and is analogous to a radio. The computer sending data can only "broadcast" without establishing a connection, and the receiving computer need not respond.

NOTE: Both the WinSock TCP and the WinSock UDP controls allow data to be exchanged in both directions.

Using VB Scripts to Run ActiveX Objects

This section describes additional controls created by Microsoft or its partners for use with Windows 95 and Internet Explorer 3.0. These objects add sound, video, and interactivity to your Web pages. Some of these controls are actually authoring controls that let you create objects. Others are controls that allow you to pass parameters to them before displaying an outcome. For instance, you can define the type, color, legend text, and other elements for a pie chart. ActiveX receives the parameters you pass it and uses the .OCX to create a chart of the type you specified.

ActiveMovie

ActiveMovie Stream is an add-on toolkit that works with Internet Explorer 3.0. You can use it to play ActiveMovie streaming format (.ASF) files over low bandwidth networks such as the Internet. These are files that send (stream) sound, pictures, and URLs over the Internet in real time. The ActiveMovie streaming format is an open and extendable data-independent format used to archive, annotate, index, and transmit synchronized multimedia content. More specifically, an .ASF file allows multiple data objects (objects such as audio objects, video objects, still images, events, URLs, HTML pages, and programs) to be combined and stored in a single synchronized multimedia stream. You can incorporate these encapsulated objects into your HTML code so that they are viewable and executable from any HTTP server. In addition, the format permits progressive rendering and image stacking for rapid replay.

ActiveMovie Stream's encapsulation allows you to efficiently synchronize and store existing popular media types and formats such as MPEG, .AVI, .WAV, and Apple QuickTime on a variety of servers. You can transmit .ASF data over a variety of protocols and networks, including TCP/IP, IPX/SPX, UDP, RTP, and ATM, with some limitations. For instance, because of bandwidth limitations and Internet data loss, you might find that the playback quality is adversely affected. However, you can use error-correction information in your files to compensate for network data loss and "jitter." Another limitation is that .ASF files cannot stream through corporate firewalls.

ActiveMusic

This control offers you the ability to create music within and across pages in a Web site. The ActiveMusic control provides an interface between the AudioActive engine and Internet Explorer 3.0. It manages file transfers, can sense page transitions, and can be called only from VB Script programs. It is not an interface for compiled Visual Basic, C, or C++ programs.

Think of AudioActive as an intelligent music system that knows how to deliver the techniques of a musician in software. In a sense, it's like being able to put a living, breathing composer inside a computer and being able to ask that composer to write music that responds to programs as they unfold without subjecting the composer to the often cramped, dusty conditions and sharp, pointy things inside a computer.

Currently the following functionality is being added to the ActiveMusic control:

- Continuous controller support
- Polychord support
- Optimization of 32-bit DLL
- COM implementation of 32-bit DLL
- OCX implementation for online usage
- Integration of the RenderActive synthesis engine

For an example of how ActiveMusic works, see the Sonic Pizza sample in the ActiveX Gallery at http://www.microsoft.com.

ActiveVRML

ActiveVRML (Active Virtual Reality Modeling Language) is a new control that lets you create interactive animation in a simple and intuitive manner. For instance, ActiveVRML frees you from complex programming of events, multithreading, sampling, and frame generation. It supports the following media types:

MEDIA TYPE	DESCRIPTION
3_D Geometry	Supports importation, aggregation, and transformation. Also supports texture mapping of interactive animated images, manipulation of color and opacity, and embedding of sounds and lights.
Images	Provides infinite resolution and extent images. Supports importation, 2-D transformation, opacity manipulation, and image overlaying. Also supports rendering an image from a 3-D model and rendering an image from rich text. Even geometric and image renderings have infinite resolution and extent since discretizational and cropping are left to the display setup, which is always left implicit.
Sounds	Provides rudimentary support for importing, manipulating, and mixing sounds. Also, provides sonic rendering of 3-D models (you can listen as well as watch). Conceptually, ActiveVRML supports infinite sampling rate and sample precision.
Montages	Supports composite 2½-D images in multilayered cel animation.
2-D and 3-D Transforms	Supports translate, scale, rotate, shear, identity, composition, inversion, and matrix-based construction. Support can be extended to nonlinear deformations.
Colors	Supports various constants, construction and deconstruction in RGB, and HSL color spaces.
Text	Supports rudimentary text, including formatted text—color, font family, and optionally bold and italic.
Miscellaneous	Supports numbers, characters, and strings.

You can use existing material, available on the Internet, as a starting point in developing interactive animations because ActiveVRML supports so many media types. Then type in definitions, texture mapping characteristics, scaling requirements, color requirements, behaviors, sound requirements, reactive behaviors, competing events, repetition, event varieties, and so on. For instance, the code to import an existing item into an earth.gif image is:

```
sphere = first(import("sphere.wrl"));
earthMap = first(import("earth-map.gif"))
```

To apply an earth-type texture to earthmap, you would type in the following code:

```
unitEarth = texture(earthmap, sphere);
```

As you can see from these examples, the coding conventions for ActiveVRML are simple. The naming and composing processes are completely independent, so the author can choose how many names to introduce and where to introduce them, based on individual style and reusability plans.

To embed the ActiveVRML viewer control on an HTML page, you need to use the <OBJECT> tag and know the GUID value of the viewer control's CLASSID value. The following is an example of how you would do this. Note that you would use the CLASSID value as shown here in your own example.

```
<OBJECT CLASSID="{389C2960-3640-11CF-9294-00AA00B8A733}" WIDTH=50 HEIGHT=50>
</OBJECT>
```

Properties

When you add the viewer control to a form, you must set properties on the control that indicate which ActiveVRML script, or .AVR file, is to be used and which expression is to be displayed. For example, assume "http://www.MyStore.com/Merchandise/shelf.avr" contains the following ActiveVRML script:

```
myGeo, ptMin, ptMax = import("cube.wrl");
model = renderedImage(myGeo, defaultCamera);
```

The ActiveVRML control supports the DataPath and Expression properties. (For this example, DataPath="http://www.MyStore.com/Merchandise/shelf.avr" and Expression="model".) You must set these two properties before the ActiveVRML viewer control will display anything.

When you embed the viewer control in a Visual Basic form, you can set the viewer's properties by using the Visual Basic properties window or by right-clicking on the viewer control and selecting ActiveVRML Viewer Properties. In the latter case, you'll see a property page dialog box for the viewer control.

DataPath property: This property indicates the URL location of the ActiveVRML file to be used. This is a string value. The following are examples:

```
http://www.MyStore.com/Merchandise/shelf.avr
c:\merchandise\shelf.avr
\\myserver\merchandise\shelf.avr
```

Expression property: This property indicates which expression exposed by the ActiveVRML script is to be displayed by the viewer. This is a string value.

Environment property: This property sets a name space for the model. This is a string value and can be left empty.

Border property: This Boolean property indicates whether the viewer control should display an inset border. The default value is True. If the value is set to False, no border is drawn.

Frozen property: This property indicates whether subsequent changes to the DataPath, Expression, or Environment properties will cause the viewer to refresh the display. The default value of this Boolean property is set to False. This property is useful in cases where you want to modify one or more of the DataPath, Expression, or Environment properties while an ActiveVRML script is being viewed. Because the viewer normally refreshes the display when any one of these properties is changed, setting the Frozen property to True will keep the viewer from attempting to display the new item. When all the desired properties have been set, changing the Frozen property to False will force the browser to redisplay the new item.

Setting Properties Under Internet Explorer

With the ActiveVRML control embedded in Internet Explorer, you need to use the <PARAM> tag to set the control properties. The following example sets some control properties:

```
<OBJECT CLASSID="{389C2960-3640-11CF-9294-00AA00B8A733}" WIDTH=50 HEIGHT=50>
<PARAM NAME="DataPath" VALUE="http://www.ASite.com/avrml/myown.avr">
<PARAM NAME="Expression" VALUE="myImage">
<PARAM NAME="Border" VALUE=False>
</OBJECT>
```

Animated Button Control

The Animated Button control displays various frame sequences of an .AVI file, depending on the button state, and uses the Windows Animation Common Control. The .AVI file must be RLE-compressed or 8-bit compressed. RLE (Run-Length Encoding) is a standard form of bitmap compression used by many graphics tools in Microsoft Windows and consumes significantly less memory than ordinary bitmaps without significant display delay. Another form of bitmap compression is 8-bit compression. You need to make sure that the starting frame of any particular sequence is a keyframe (a limitation based on Windows Animation Common Control). The palette of the file must match the palette used by Internet Explorer. Currently this means that the Windows halftone palette is returned by the CreateHalftonePalette API. In the future, more palettes may be supported.

The Animated Button control can be in any of the following four states:

- DOWN, when the control receives a left mouse button click.
- FOCUS, when the control gets focus. (Focus is the ability to receive user input through the mouse or the keyboard. When an object has focus, it can receive input from a user.)
- MOUSEOVER, when the mouse pointer moves over the control.
- DEFAULT, when both the mouse cursor and the focus are not on the control.

All the states are mutually exclusive. This can be confusing because the mouse pointer can move over the control but can still have focus. However, there is a hierarchical precedence. DOWN has precedence over all the other states, and DEFAULT has the least precedence. MOUSEOVER has precedence over FOCUS. So if a control has a FOCUS state and the mouse moves over it, the state changes to the MOUSEOVER state.

The Animated Button control has the following properties associated with it:

URL: The URL location of the AVI file to be used.

DefaultFrStart: The start frame for the DEFAULT state.

DefaultFrEnd: The end frame for the DEFAULT state.

MouseoverFrStart: The start frame for the MOUSEOVER state.

MouseoverFrEnd: The end frame for the MOUSEOVER state.

FocusFrStart: The start frame for the FOCUS state.

FocusFrEnd: The end frame for the FOCUS state.

DownFrStart: The start frame for the DOWN state.

DownFrEnd: The end frame for the DOWN state.

The Animated Button control has the following method associated with it:

AboutBox: Displays the About box.

The Animated Button control has the following events associated with it:

ButtonEvent_Click: Fired when the button is clicked.

ButtonEvent_DblClick: Fired when a double click occurs.

ButtonEvent_Focus: Fired when the button gets focus.

ButtonEvent_Enter: Fired when the mouse pointer enters the button area.

ButtonEvent_Leave: Fired when the mouse pointer leaves the button area.

The HTML code to insert the Animated Button control in this page is as follows:

```
<OBJECT
CODEBASE="http://ohserv/ie/download/activex/ieanbtn.ocx#Version=4.70.0.1085"
    ID=anbtn
    CLASSID="clsid:0482B100-739C-11CF-A3A9-00A0C9034920"
    WIDTH=300
    HEIGHT=200
    ALIGN=CENTER
    HSPACE=0
    VSPACE=0
```

(continued)

```
>
<PARAM NAME="defaultfrstart" VALUE="0">
<PARAM NAME="defaultfrend" VALUE="7">
<PARAM NAME="mouseoverfrstart" VALUE="8">
<PARAM NAME="mouseoverfrend" VALUE="15">
<PARAM NAME="focusfrstart" VALUE="16">
<PARAM NAME="focusfrend" VALUE="23">
<PARAM NAME="downfrstart" VALUE="24">
<PARAM NAME="downfrend" VALUE="34">
<PARAM NAME="URL" VALUE="http://ohserv/win95.avi">
</OBJECT>
```

Chart Control

The Chart control (IECHART) lets you draw and define various chart types. You can use Chart to define the following chart types:

- Pie chart
- Point chart
- Line chart
- Area chart
- Bar chart
- Column chart
- Stocks chart

The Chart control supports only one method—AboutBox—and no events. To insert the Chart control into a Web page, follow the examples listed later in this section. Pass the parameter values for chart elements, such as color, grid, and legend, to the object, and watch as the Chart is created using ActiveX. For more complex chart usage, you can use VB Script to add timing values to your chart.

The following code inserts a chart into HTML code. Both objects are ActiveX objects. Notice how the CODEBASE command indicates the site of the control object (.OCX file).

```
<OBJECT
          CLASSID="{FC25B780-75BE-11CF-8B01-444553540000}"
          CODEBASE="http://www.microsoft.com/ie/download/activex/
iegrad.ocx#Version=4,70,0,1082"
          ID=chart1
          WIDTH=300
          HEIGHT=150
          ALIGN=center
          HSPACE=0
          VSPACE=0
   >
   <PARAM NAME="_extentX" VALUE="300">
   <PARAM NAME="_extentY" VALUE="150">
   <PARAM NAME="ChartStyle" VALUE="0">
   <PARAM NAME="ChartType" VALUE="0">
   <PARAM NAME="hgridStyle" VALUE="0">
   <PARAM NAME="vgridStyle" VALUE="0">
   <PARAM NAME="colorscheme" VALUE="1">
   <PARAM NAME="rows" VALUE="4">
   <PARAM NAME="columns" VALUE="4">
   <PARAM NAME="data[0][0]" VALUE="30">
   <PARAM NAME="data[0][1]" VALUE="60">
   <PARAM NAME="data[0][2]" VALUE="20">
   <PARAM NAME="data[0][3]" VALUE="40">
   <PARAM NAME="data[1][0]" VALUE="31">
   <PARAM NAME="data[1][1]" VALUE="61">
   <PARAM NAME="data[1][2]" VALUE="21">
   <PARAM NAME="data[1][3]" VALUE="41">
   <PARAM NAME="data[2][0]" VALUE="32">
   <PARAM NAME="data[2][1]" VALUE="62">
   <PARAM NAME="data[2][2]" VALUE="22">
   <PARAM NAME="data[2][3]" VALUE="42">
   <PARAM NAME="data[3][0]" VALUE="33">
   <PARAM NAME="data[3][1]" VALUE="63">
   <PARAM NAME="data[3][2]" VALUE="23">
   <PARAM NAME="data[3][3]" VALUE="43">
```

(continued)

```
   </OBJECT>
<OBJECT ID=timer CLASSID="{59CCB4A0-727D-11CF-AC36-00AA00A47DD2}">
<PARAM NAME="TimeOut" VALUE="750">
<PARAM NAME="enable" VALUE="1">
</OBJECT>
<SCRIPT LANGUAGE="VBS">
sub timer_time
    i = chart1.chartstyle
    chart1.chartstyle = chart1.chartstyle + 1
    if i = chart1.chartstyle then
        chart1.chartstyle = 0
        i = chart1.charttype
        chart1.charttype = chart1.charttype + 1
        if i = chart1.charttype then
            chart1.charttype = 0
            chart1.chartstyle = 0
        end if
    end if
end sub
</SCRIPT>
```

The boldface lines show how VB Script is incorporated into the HTML code. In the example, the script executes a subroutine named timer_time.

Comic Chat

Comic Chat is a new kind of graphical chat program developed by Microsoft. In recent years, a number of graphical chat programs have appeared for online services and the Internet. Comic Chat's approach varies from other graphical chat programs in that it uses a visual representation of conversations based on comic conventions. A comic strip unfolds, showing the various participants in the conversation as comic characters and their utterances in word balloons, as users type in text. To keep the users focused more on chatting and less on controlling their characters, Comic Chat automates a number of processes. It automatically chooses a default gesture and expression based on the text that is typed. It determines who should be in each panel and where they should be. There is no need to wander around looking for people to participate in a conversation. If somebody speaks to a user, the user need not turn around and try to find the speaker.

Comic Chat runs on Windows 95. It uses the Internet Relay Chat (IRC) protocol, the most popular chat protocol on the Internet, and it runs on standard IRC servers.

Gradient Control

This control shades the specified area with a range of colors, making a gradual transition from a specified color to another specified color. You can choose two different colors or a range of shades within a color. You can also specify the direction in which the gradation should occur and the gradation starting and ending coordinates.

Properties

The Gradient control has the following properties.

StartColor: The StartColor property determines the color with which the transition starts.

EndColor: The EndColor property determines the color with which the transition ends.

Direction: The Direction property determines the direction in which the color moves. It takes one of the following values:

VALUE	DIRECTION
0	The color transition is horizontal.
1	The color transition is vertical.
2	The color transition is toward the center.
3	The color transition is toward a corner.
4	The color transition is across, diagonally down.
5	The color transition is across, diagonally up.
6	The color transition is around the point specified in the StartPoint property.
7	The color transition is across the line joining the StartPoint and EndPoint properties.

StartPoint: The StartPoint property determines the coordinates of a starting point, in the format (x,y).

EndPoint: The EndPoint property determines the coordinates of an ending point, in the format (x,y).

Methods

The Gradient control has the following method.

AboutBox: This method produces the About Gradient Control box.

Events

The Gradient control has no events.

The code to insert a graded color in a square block is as follows:

```
<CENTER>
<FONT SIZE=5><B>Gradient Control</B></FONT>
</CENTER>
<BR>
This control shades the area with a range of colors, making the transition from
a specified color to another specified color.
<HR>
The following is a Gradient object.
<BR>
<OBJECT
ID=iegrad1
TYPE="application/x-oleobject"
CLASSID="clsid:017C99A0-8637-11CF-A3A9-00A0C9034920"
CODEBASE="http://www.microsoft.com/ie/download/activex/
iegrad.ocx#Version=4,70,0,1082"
WIDTH=50
HEIGHT=50
>
<PARAM NAME="StartColor" VALUE="#0000ff">
<PARAM NAME="EndColor" VALUE="#000000">
<PARAM NAME="Direction" VALUE = "4">
</OBJECT>
```

Intrinsic Controls

There are several intrinsic, or common, controls that come with Internet Explorer 3.0. Controls such as buttons, check boxes, combo boxes, list boxes, and passwords are included in the ActiveX SDK and are installed during Internet Explorer 3.0 setup. These controls are found in HTMLCTL.OCX, which is copied to the hard drive and registered during setup. You use standard HTML forms tags and values to implement these controls.

The following controls are found in HTMLCTL.OCX:

CONTROL	TAG
ButtonCtl Object	INPUT TYPE=BUTTON
CheckboxCtl Object	INPUT TYPE=CHECKBOX
ComboCtl Object	SELECT MULTIPLE
ListCtl Object	SELECT
PasswordCtl Object	INPUT TYPE=PASSWORD
RadioCtl Object	INPUT TYPE=RADIO
TextAreaCtl Object	TEXTAREA
TextCtl Object	INPUT NAME

Label Control

The Label control displays text at a given angle and supports the Click event.

The HTML code to insert the Label control into a Web page at 270 degrees is as follows:

```
<OBJECT
        CLASSID="clsid:{99B42120-6EC7-11CF-A6C7-00AA00A47DD2}"
        ID=sprlbl1
        WIDTH=150
```

(continued)

```
        HEIGHT=500
        VSPACE=0
        ALIGN=left
>
<PARAM NAME="_extentX" VALUE="14">
<PARAM NAME="_extentY" VALUE="14">
<PARAM NAME="angle" VALUE="270">
<PARAM NAME="alignment" VALUE="2">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="caption" VALUE="Properties">
<PARAM NAME="FontName" VALUE="Times New Roman">
<PARAM NAME="FontSize" VALUE="130">
</OBJECT>
```

New Item Control

You can use the New Item control to highlight new items on a Web page. Up to a specified date, it displays the selected image. After that date, it does not continue to display itself or the item.

The HTML to insert the New Item control in a Web page is as follows:

```
<OBJECT
        ID=ienewb
        TYPE="application/x-oleobject"
        CLASSID="clsid:{642B65C0-7374-11CF-A3A9-00A0C9034920}"
        WIDTH=20
        HEIGHT=10
>
<PARAM NAME="date" VALUE="5/1/1996">
<PARAM NAME="image" VALUE="/ie/images/new.gif">
</OBJECT>
```

The "date" parameter specifies the ending date—the date after which you don't want the item displayed. The image parameter defines the image that you want to display.

Popup Menu Control

The Popup Menu control displays a popup menu whenever it is enabled. The Popup Menu control is invoked by calling the method PopUp.

PARAM Tag

The Popup Menu control uses the <PARAM> tag with the following attribute.

Menuitem[]: This tag produces the menu item to be displayed.

Methods

The Popup Menu control has the following methods.

AboutBox: This method displays information about the menu.

PopUp([in] int x, [in] int y): This method pops up the menu. If no value is passed, the current mouse pointer position is used for displaying the popup menu.

Clear(): This method clears all the menu items.

RemoveItem([in] int index): This method removes the specified item. If the menu item does not exist, nothing is done.

AddItem([in] String, [in/optional] int index): This method adds the passed menu item to the menu at the specified index. If no index is passed, the item is appended to the end.

Events

The Popup Menu control has the following event.

Click(int item): The item clicked is one of the parameters passed.

HTML Code

The HTML code to insert the Popup Menu control in this page is as follows:

```
    <OBJECT
        CODEBASE="http://www.microsoft.com/ie/download/activex/
iemenu.ocx#Version=4,70,0,1082"
        ID=iemenu1
```

(continued)

```
        CLASSID="clsid:0482B100-739C-11CF-A3A9-00A0C9034920"
        WIDTH=1
        HEIGHT=1
        ALIGN=left
        HSPACE=0
        VSPACE=0
    >
    <PARAM NAME="Menuitem[0]" VALUE="This is the first item">
    <PARAM NAME="Menuitem[1]" VALUE="This is the second item">
    <PARAM NAME="Menuitem[2]" VALUE="This is the third item">
    <PARAM NAME="Menuitem[3]" VALUE="No way this is the fifth item">
    <PARAM NAME="Menuitem[4]" VALUE="This is the fifth item">
    </OBJECT>

<SCRIPT LANGUAGE="VBScript">
sub Iepop1_Click(ByVal x)
    Alert "Menu click on item:  "&x
    Randomize
    call Iepop1.RemoveItem(x)
    call Iepop1.AddItem("Added Me", x)
    call Iepop1.PopUp(Rnd*640,Rnd*480)
end sub
sub timer1_timer
    Alert "Got timer"
    timer1.Enabled = False
    call Iepop1.PopUp()
end sub
</SCRIPT>
```

Preloader Control

Preloader is an invisible control, in that the user does nothing and the code controls what image downloads behind the image the user is currently viewing. Preloader lets you specify an image to be downloaded to the Internet Explorer cache after a user decides that he or she wants to load that image. Downloading to the cache takes less time than downloading to disk, especially with large images, giving your Web page the appearance of more efficient operation.

Here is the code to use to preload an object using Preloader.

```
<OBJECT
            ID=movie
            CLASSID="clsid:16E349E0-702C-11CF-A3A9-00A0C9034920"
            CODEBASE="http://ohserv/ie/download/activex/ieprld.ocx"
            WIDTH=1
            HEIGHT=1
>
<PARAM NAME="URL" VALUE="http://ohserv/win95.avi">
<PARAM NAME="enable" VALUE="1">
</OBJECT>
```

Label Control

You can use the Label control to display text at a given angle and along curves that you define. It supports CLICK, CHANGE, MOUSEDOWN, MOUSEOVER, and MOUSEUP events. This control operates like WordArt.

Caption

This specifies the text to be displayed.

Angle

This specifies in degrees the counterclockwise rotation of the given text.

Alignment

This specifies the alignment of the text in the control. Alignment takes the following values.

0: Alignment is to the left.

1: Alignment is to the right.

2: Alignment is centered.

3: Alignment is to the top.

4: Alignment is to the bottom.

BackStyle
This controls the background. Backstyle takes the following values.

0: The background is transparent.

1: The background is opaque.

FontName
This is the name of a TrueType font.

FontSize
This is the size of the font.

FontItalic
This is a flag for italicized text.

FontBold
This is a flag for bold text.

FontUnderline
This is a flag for underlined text.

FontStrikeout
This is a flag for strikeout text.

Mode
This specifies which mode the text should be rendered in. Mode takes the following values.

0: The mode is normal (the same as the Label control).

1: The mode is normal text with rotation.

2: This value applies the user-specified lines while rendering without rotation.

3: This value applies the user-specified lines while rendering with rotation.

To specify the two lines along which the text will be shown, use <PARAM> tags such as TopPoints, TopXY, BotPoints, and BotXY. You can use this feature in VB Script because it supports properties such as TopPoints, TopIndex, TopXY, BotPoints, BotIndex, and BotXY. You can also use a

property page (an OLE function that occurs when a Properties window in Visual Basic is encountered) with this control.

HTML Code

The following HTML code inserts the Label control (IELABEL.OCX) into your Web page in a vertical rendering of 270 degrees.

```
<OBJECT
    CLASSID="clsid:99B42120-6EC7-11CF-A6C7-00AA00A47DD2"
    CODEBASE="http://ohserv/ie/download/activex/ielabel.ocx#version=4
        ID=sprlbl1
        WIDTH=150
        HEIGHT=150
        VSPACE=0
        ALIGN=LEFT
>
<PARAM NAME="Angle" VALUE="270">
<PARAM NAME="Alignment" VALUE="2">
<PARAM NAME="BackStyle" VALUE="0">
<PARAM NAME="Caption" VALUE="Properties">
<PARAM NAME="FontName" VALUE="Times New Roman">
<PARAM NAME="FontSize" VALUE="60">
</OBJECT>
```

Timer Control

The Timer control evokes events on a periodic basis and is an invisible control at run time.

The following HTML code inserts the Timer control (IETIMER.OCX) into your Web page so that when you click on a button, you toggle between enabled and disabled states. When enabled, you can use the Timer control to wait before executing a command, or to automatically execute a command or run an application at a designated interval.

```
<OBJECT
    CLASSID="clsid:59CCB4A0-727D-11CF-AC36-00AA00A47DD2"
    CODEBASE="http://ohserv/ie/download/activex/ietimer.ocx#version=4,70,0,1085"
```

(continued)

```
    ID=timer1
    ALIGN=MIDDLE
>
<PARAM NAME="Interval" VALUE="200">
<PARAM NAME="Enabled" VALUE="True">
</OBJECT>
<OBJECT
    CLASSID="clsid:59CCB4A0-727D-11CF-AC36-00AA00A47DD2"
    CODEBASE="http://ohserv/ie/download/activex/ietimer.ocx#version=4,70,0,1085"
    ID=timer2
    ALIGN=MIDDLE
>
<PARAM NAME="Interval" VALUE="1000">
<PARAM NAME="Enabled" VALUE="True">
</OBJECT>

<SCRIPT LANGUAGE="VBSCRIPT">
Sub BtnToggle_OnClick
    Timer1.Enabled = Not Timer1.Enabled
    Timer2.Enabled = Not Timer2.Enabled
End Sub

sub timer1_timer
    label.Angle = (label.Angle + 5) mod 360
end sub
sub timer2_timer
    cool.forecolor = rnd() * 16777216
end sub
</SCRIPT>
```

ActiveX and VB Script Examples

The following information provides you with some dos and don'ts for using VB Script with ActiveX components. It also provides working examples of the VB Script code in its current implementation. The examples are based on two working ActiveX demo applications from Microsoft:

Volcano Coffee Company and Sonic Pizza. These examples are packaged with the ActiveX SDK code, and the most up-to-date version can be downloaded from MSN or at http://www.microsoft.com. Before we get to the samples, however, let's look at some common misunderstandings that developers have when using this new technology.

Some Common Misunderstandings

Contrary to what you might have heard, VB Script will not be able to perform file I/O (input and output) or access hardware directly, and it will live and run within a browser. In the current version, in order to be "safe," any code that could potentially cause damage to a user's computer has been disabled. When security on the Internet is no longer a big issue, this safety feature may no longer be necessary.

Many developers have trouble understanding that objects are not simple variables. In all flavors of Visual Basic, including VB Script, you must use special operators when dealing with objects. For example, you will get an error when you try to execute the following code:

```
<SCRIPT LANGUAGE="VBSCRIPT">
Sub Button1_OnClick
      if x = empty then
        msgbox "x is empty"
      end if
      if addrbook = empty then
        msgbox "no addrbook"
      end if
End Sub
Sub Button2_OnClick
      if x = empty then
        msgbox "x is empty"
      end if
      if addrselector = empty then
        msgbox "no addrselector"
      end if
End Sub
</SCRIPT>
```

As a comparison, use the Is operator, and to perform an assignment, use Set as in the following code:

```
Dim X
    Set X = SomeOtherObj
    If X Is Nothing Then
        'The assignment didn't work
    Else
        X.method
    End If
```

If X is an object, use a line of code like this:

```
If X = empty Then
```

This amounts to saying the following:

```
If X.Defaultproperty = 0 Then
```

This is because "empty" has no meaning. (Visual Basic thinks it is a variable you've just created.) If the object in question does not have a default property, you will get an error stating that the object doesn't support this property or method.

Coding with ActiveX

Now let's look at some sample code. The following section uses the Volcano Coffee Company demo program to illustrate how to combine HTML 3.0, VB Script, and ActiveX objects into a working Web page that serves as an online catalog and ordering system. The opening screen is shown in Figure 13-1.

The Volcano Coffee Company Web page is built using frames. Each frame has been coded into a separate HTML file and pulled together from a master HTML file. The following code is in the master HTML file (VOLCANO3.HTM). This file describes each frame, names the frames "lefty" and "righty," and codes the sources of the other frame files so that they can be run from the master program. In addition, it loads a dynamic source, which loops an infinite number of times.

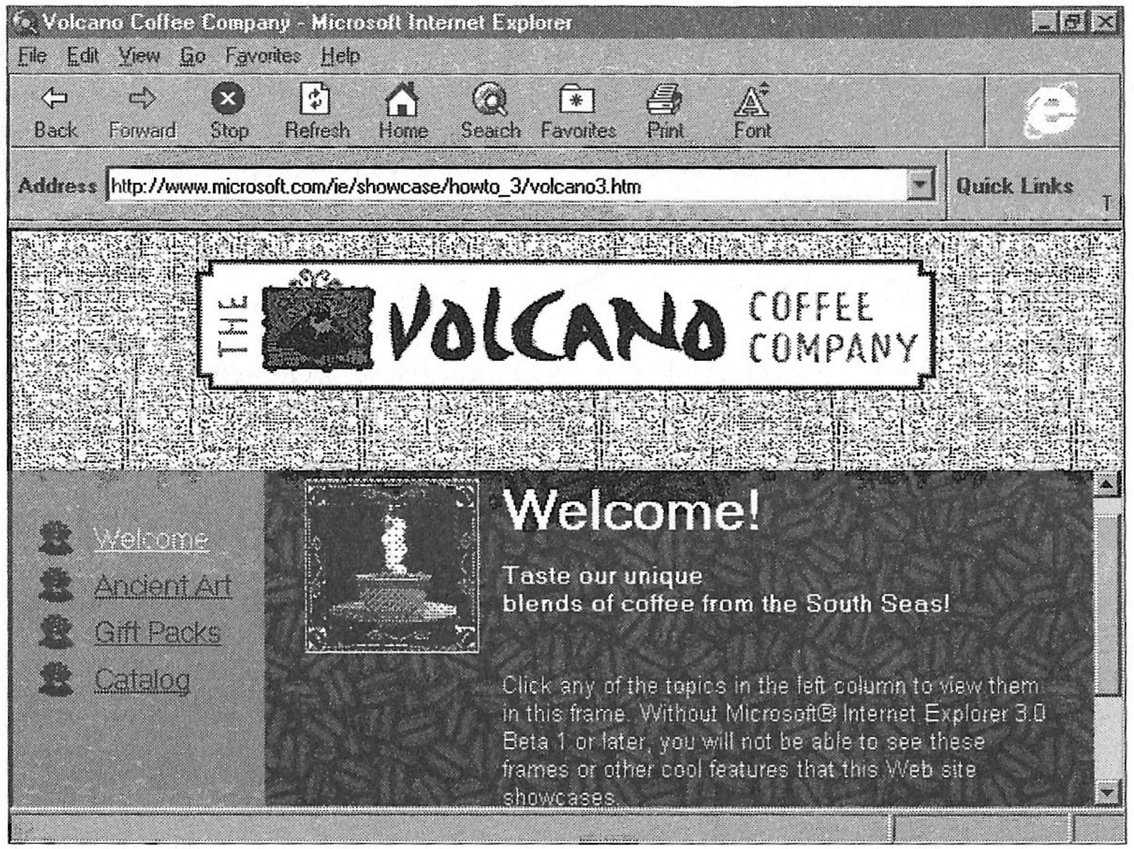


FIGURE 13-1 ***The Volcano Coffee Company Welcome screen.***

This code displays the cup in the upper left corner of the Welcome screen and causes the cup to loop continuously. Notice that the code that indicates the sound loop is not used until later, when the Welcome frame (righty) is actually created.

If you have Internet access, you can display the most current version of this demo at http://www.microsoft.com/ie/showcase/howto_3/volcano3.htm.

```
<HEAD>
<TITLE> Volcano Coffee Company</TITLE>
</HEAD>
<FRAMESET ROWS="137,65%" FRAMEBORDER=0 FRAMESPACING=0>
        <FRAME SRC="/ie/showcase/howto3/masthead.htm" SCROLLING="no" NORESIZE>
        <FRAMESET COLS="145,*">
        <FRAME NAME="lefty"
SRC="/ie/showcase/howto_3/contents.htm"
SCROLLING="no" NORESIZE SCROLL=NO>
```

(continued)

```
        <FRAME NAME="righty"
SRC="/ie/showcase/howto_3/welcome.htm"
SCROLLING="yes" NORESIZE>
        </FRAMESET>
</FRAMESET>

<NOFRAMES>

<BODY TOPMARGIN="0"  BACKGROUND="/ie/images/canvas2.gif" BGPROPERTIES=FIXED
LINK="#000000" VLINK="#000000">
<BGSOUND SRC="/ie/audio/drmshort.wav" LOOP=3>
<CENTER>
<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR VALIGN="top">
<TD><IMG SRC="/ie/images/masthead.gif" HSPACE=5 ALT="The Volcano Coffee Company"
WIDTH=424 HEIGHT=75>
</TD>
</TR>
</TABLE>
<P>
<P>
<CENTER>
<TABLE BORDER=1 BORDERCOLOR="BLACK" BGCOLOR="#FFFCC" WIDTH=90%  CELLPADDING=3
CELLSPACING=3>

<TR BGCOLOR="#D36D32" ALIGN=LEFT VALIGN=TOP>
<TD ROWSPAN=2><IMG DYNSRC="/ie/AVI/cup.avi"
SRC="/ie/avi/cupalt.gif" HSPACE=2 LOOP=INFINITE WIDTH=100
HEIGHT=100 ALIGN=LEFT></TD>
<TD><FONT FACE="HELV" SIZE=6 COLOR="#F3D2A6"><B>Welcome!</B></FONT>
</TD>
</TR>

<TR BGCOLOR="#D36D32">
<TD VALIGN=TOP><FONT FACE="HELV" SIZE=3 COLOR="#F3D2A6"><B>Taste our unique</B>
<BR>
blends of coffee from the South Seas!</FONT>
</TD>
</TR>
```

```
<TR BGCOLOR="#D36D32">
<TD COLSPAN=2><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24
HSPACE=6><A HREF="/ie/showcase/howto_3/welcome.htm"><FONT FACE="HELV"
SIZE=3>Welcome</FONT></A>
<BR>
</TD>
</TR>

<TR BGCOLOR="#D36D32">
<TD COLSPAN=2><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24
HSPACE=6>
<A HREF="/ie/showcase/howto_3/ancient.htm"><FONT FACE="HELV" SIZE=3>Ancient
Art</FONT></A>
<BR>
</TD>
</TR>

<TR BGCOLOR="#D36D32">
<TD COLSPAN=2><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24
HSPACE=6>
<A HREF="/ie/showcase/howto_3/catalog.htm"><FONT FACE="HELV" SIZE=3>Catalog
</FONT></A>
<BR>
</TD>
</TR>

<TR>
<TD COLSPAN=2><FONT FACE="HELV" SIZE=2 Color="#black">Because you are viewing
this page in Internet Explorer 2.0, you cannot see how the browser window is
divided into frames or see other cool features of Microsoft Internet Explorer
3.0. You can:
<BR>
<P>
<IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24 HSPACE=6>View
the showcase page for <A HREF="/ie/showcase/howto/iedemo.htm">Microsoft Internet
Explorer 2.0</A>
<BR>
```

(continued)

```
<P>
<IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24 HSPACE=6>View
the demo that explains the <A HREF="/ie/showcase/howto_3/pdcdemo.htm">Microsoft
Internet Explorer 3.0 showcase pages.</A>
<BR>
<P>
<IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24
HSPACE=6>Download a beta version of <A HREF="/intdev/sdk/"><B>Microsoft Internet
Explorer 3.0. including the Microsoft ActiveX(tm) Developers Kit.</B></A></FONT>
</TD>
</TR>
</TABLE>
</CENTER>
</CENTER>
</BODY>
</NOFRAMES>
</HTML>
```

The masthead is called into place by some additional code. The following code shows how the masthead was placed as part of the Welcome screen:

```
<HTML>
<HEAD>
<TITLE>The Volcano Coffee Company</TITLE>
</HEAD>
<BODY BACKGROUND="/ie/images/canvas2.gif"  BGPROPERTIES=FIXED>
<BGSOUND SRC="/ie/audio/drmshort.wav" LOOP=3>
<CENTER>
<TABLE BORDER=0 CELLSPACING=0 CELLPADDING=0>
<TR VALIGN="top"><TD>
<IMG SRC="/ie/images/masthead.gif" HSPACE=5 ALT="The Volcano Coffee Company"
WIDTH=424 HEIGHT=75>
</TD>
</TR>
</TABLE>
</CENTER>
</BODY>
</HTML>
```

Here is the code to create the table of contents frame. Of particular note are the boldface lines that describe where to place the next frame ("righty"), where to get the next frame's .HTM file, and what to print on its masthead.

```
<HTML>
<HEAD>
<TITLE>Contents - Volcano Coffee Company </TITLE>
<META NAME="GENERATOR" CONTENT="Internet Assistant for Microsoft Word 2.0z Beta">
</HEAD>
<BODY BGCOLOR="#D36D32" LEFTMARGIN=10 TOPMARGIN=10 LINK="#000000" VLINK="#F3D2A6">
<BASEFONT SIZE=2 FACE="HELV">
<BR>
<TABLE WIDTH=140 CELLPADDING=0 CELLSPACING=3>
<TR>
<TD><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24></TD>
<TD><A TARGET="righty" HREF="/ie/showcase/howto_3/welcome.htm">
<FONT SIZE=3>Welcome<BR></TD></FONT></A>
</TR>
<TR>
TD><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24></TD>
TD><A TARGET="righty" HREF="/ie/showcase/howto_3/ancient.htm">
ONT SIZE=3>Ancient Art<BR></TD></FONT></A>
/TR>
TR>
TD><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24></TD>
TD><A TARGET="righty" HREF="/ie/showcase/howto_3/giftpack.htm">
ONT SIZE=3> Gift Packs<BR></TD></FONT></A>
/TR>
TR>
TD><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24></TD>
TD><A target="righty" HREF="/ie/showcase/howto_3/catalog.htm">
FONT SIZE=3>Catalog<BR></TD></FONT></A>
/TR>
TR>
TD><IMG SRC="/ie/images/volbttn.gif" ALIGN=MIDDLE WIDTH=24 HEIGHT=24></TD>
TD><A target="righty" HREF="/ie/showcase/howto_3/coffee.htm">
FONT SIZE=3>Brew A Cup!</FONT></A><BR></TD>
```

(continued)

This is the first 40 pages of a 70-Page chapter on ActiveX. I included it to show my versatility as a technical writer.