Tutorial

Visual InterDev 101

Visual InterDev is a comprehensive development tool that allows you to build Web sites and Web applications. You can use Visual InterDev to build individual HTML pages, Active Server Pages, as well as large scale Web Applications that involve hundreds of pages designed to be used by thousands of people. In addition, the tool has features allowing you to do complete site design and to debug the pages within your site.

Microsoft has positioned Visual InterDev to be its premiere tool for Web developers. As such, the company has built in advanced development features that are part of a proprietary framework. This framework, the *Scripting Object Model*, provides a layer of automatic coding enabling you to create very powerful, event-driven scripts that work with *Design Time Controls* that are unique in the Visual InterDev environment. In addition, Microsoft has integrated its flavor of Java, Visual J++, into the development environment. You can use Visual J++ and its Windows Foundation Classes (WFC) to write Java applications that go beyond the “run anywhere” mantra of Java inter-platform development. You use WFC to work indirectly with the Windows API to create Windows specific Java programs. Although it is true that Visual InterDev’s bells and whistles try to persuade you to use the product for all of your Web development, it is quite possible to use it to make a robust, powerful Web application and yet not be bound to it as an exclusive tool, should you wish to migrate to other development environments later on. Visual InterDev is a powerful tool that can save you time and labor no matter how deeply you plan to commit yourself to it.

The intent of this tutorial is to give you a functional overview of Visual InterDev. We are not going to take a detailed look at any one feature.

Hopefully, this tutorial will provide enough interest that you'll want to follow up on the topics presented here in other books and periodicals addressing the features of Visual InterDev in detail. In addition, Visual InterDev ships with detailed online documentation you can use to extend your knowledge of the topics presented here.

In this tutorial you will learn

- About the different features of Visual InterDev
- About the Visual InterDev Integrated Development Environment
- How to create Web Pages using Visual InterDev
- How to debug Web pages using Visual InterDev

Creating Web Pages Using Visual InterDev

We'll start the tutorial by making a simple Web page using Visual InterDev to get a feel for the product. Then, we'll look at the Integrated Development Environment. Finally, we'll create a Visual InterDev project and learn how to use the product's debugging features. We'll omit covering the Scripting Object Model and Visual J++. These are broad topics and well outside the scope of this tutorial.

The best way to learn about Visual InterDev is to make a Web page using it. So let's get on with it.

Making a Web Page

The first thing we are going to do is to learn how to make a simple Web page using Visual InterDev.

1. First, invoke Visual InterDev by selecting it from the Programs, Microsoft Visual Studio 6.0 menu of the Start menu (see Figure 1).
2. When Visual InterDev starts, it presents you with a New Project dialog box. For now, we are going to Cancel the dialog. We'll learn about Visual InterDev projects at another time (see Figure 2).
3. Instead, we are going to add an HTML page to the development environment. You add an HTML page to the development environment by selecting New File from the File menu.

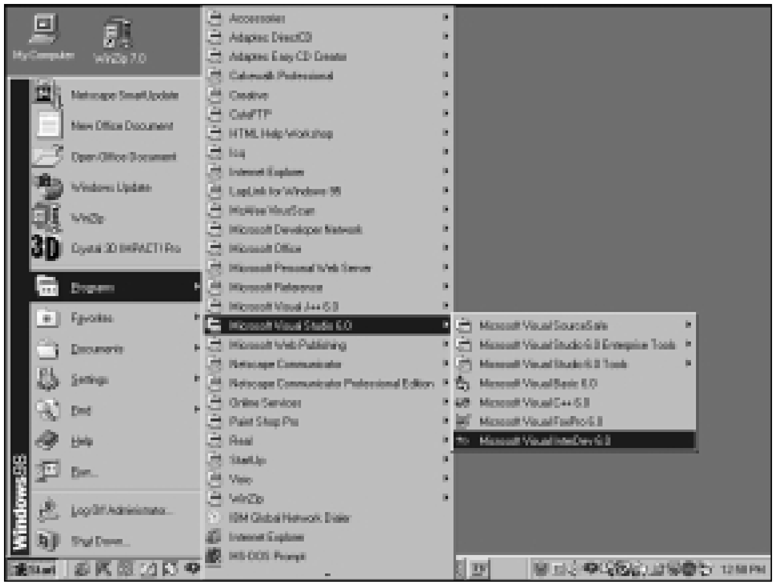
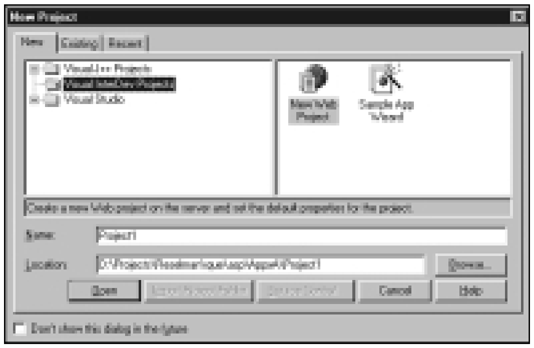


Figure 1: *Visual InterDev is part of the Visual Studio family of products.*



Create a new Web project on the server and set the default properties for the project.

Name: Project1

Location: D:\Projects\Readmar\queAsp\Apps4\Project1

□ Don't show this dialog in the future

Figure 2: *Visual InterDev shows you the New Project dialog box when it starts up.*

4. The New File dialog allows you to choose from many file types. In this case, we are going to choose an HTML Page (see Figure 3).

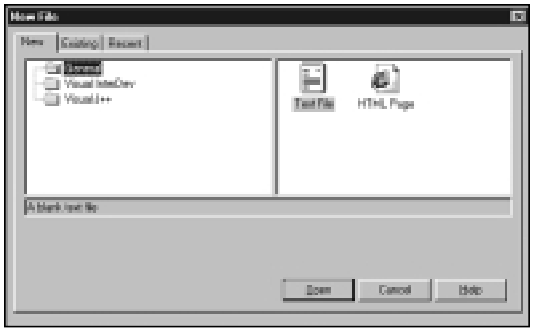


Figure 3: *In addition to HTML pages, you can add many other types of files to the IDE, including, but not limited to, text files, Java files, or ASP files.*

5. When Visual InterDev loads an HTML page into the development environment, it activates the HTML editor in the central development pane. You can type text directly into the Design View pane and format the text you enter. (Notice that the Design tab is selected at the bottom of the page in Figure 4.) Visual InterDev translates your data into HTML behind the scenes.

6. You can see in the example in Figure 4 that we have entered text directly into the HTML editor.

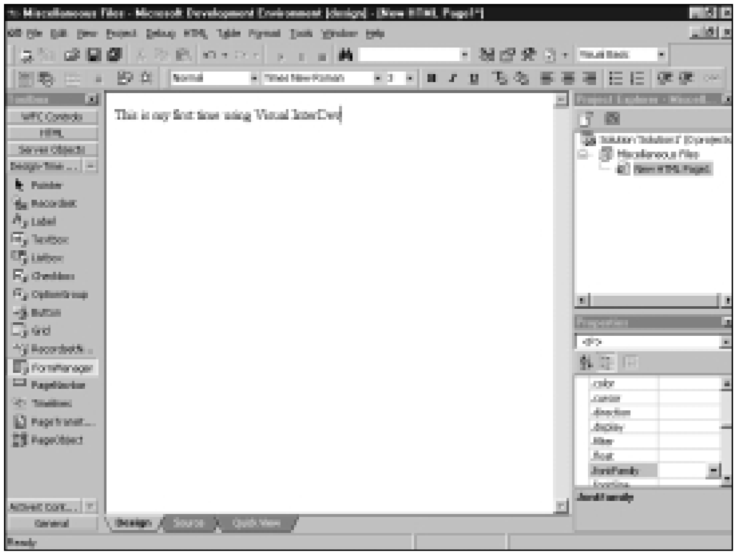
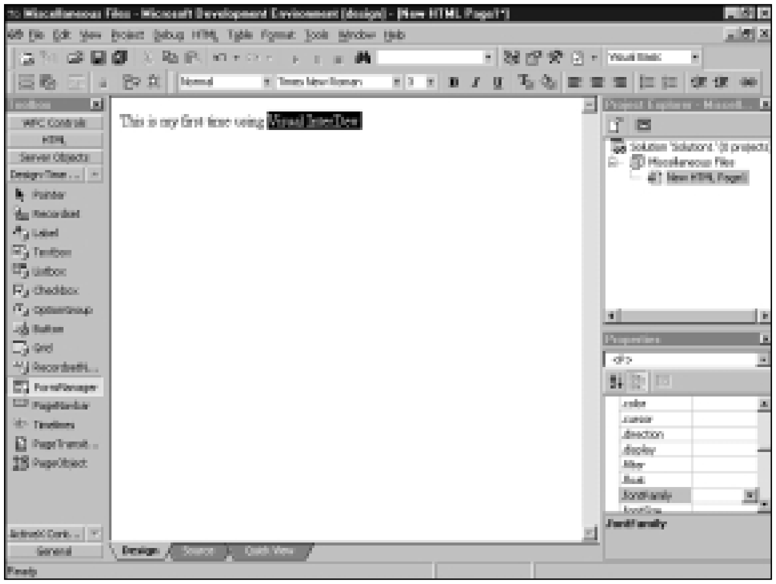

This is my first time using Visual InterDev

Figure 4: *Visual InterDev ships with a WYSIWYG HTML editor.*

7. Visual InterDev allows you to highlight text and adjust the text's format, just as we would in a standard word processor (see Figure 5) .

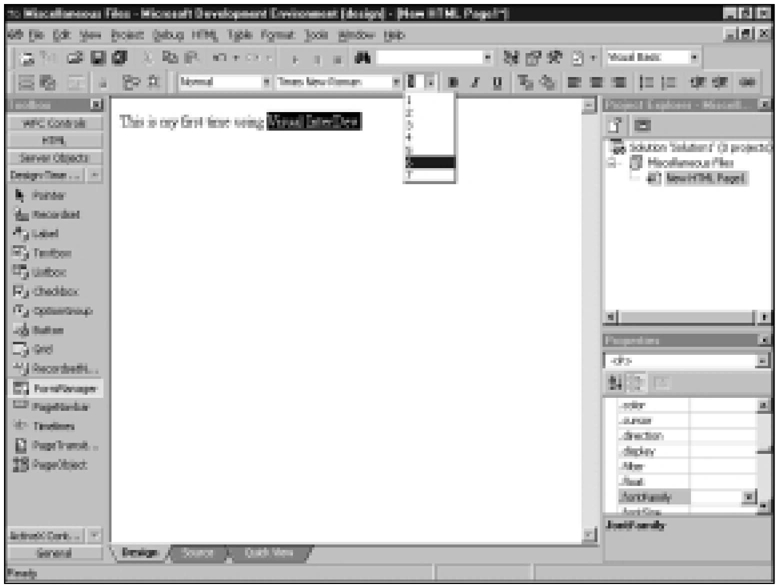


This is my first time using Visual InterDev

Figure 5: *You can change the format of lines, words, or characters in the Visual InterDev HTML editor. All you need to do is highlight the text you want to change.*

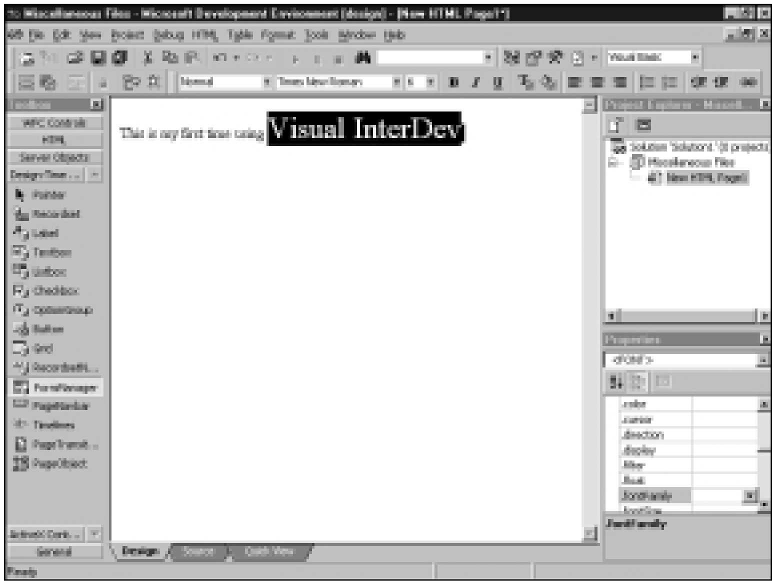
8. Visual InterDev allows you to change the size of a character's font. You select font size from the Font Size drop down list located on the toolbar at the top of the IDE. Notice that the font size numbers indicate relative sizing as opposed to fixed point sizes. HTML sizes fonts relatively. In other words, a font of size 5 is larger that a font of size 4. The actual size of the font is determined by the browser (see Figure 6).

9. When you select a new font size, the change automatically appears in the Visual InterDev HTML editor (see Figure 7).


Miscellaneous Files - Microsoft Development Environment [design] - [New HTML Page1*]
File Edit View Project Debug HTML Table Format Tools Window Help
Normal
Times New Roman
Toolbox
Project Explorer
Properties
Design
Source
Quick View
Ready

This is my first time using Visual InterDev

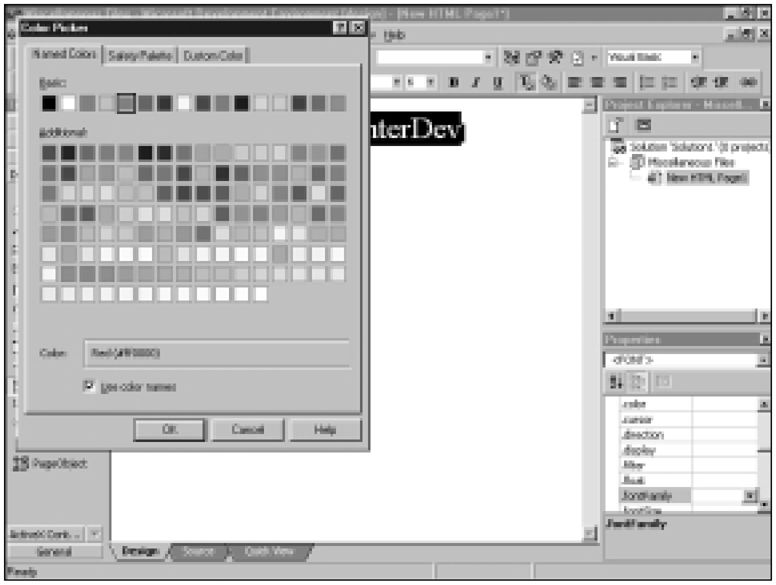
Figure 6: *Visual InterDev uses relative sizing when defining a font size. This is a standard practice in HTML.*


Miscellaneous Files - Microsoft Development Environment [design] - [New HTML Page1*]
File Edit View Project Debug HTML Table Format Tools Window Help
Normal
Times New Roman
Toolbox
Project Explorer
Properties
Design
Source
Quick View
Ready

This is my first time using Visual InterDev

Figure 7: *Visual InterDev allows you to see the changes you make in your Web pages immediately.*

10. Visual InterDev allows you to change a font's color. All you need to do is select the font and then click T in the IDE's HTML toolbar, which is situated underneath the menu bar. Clicking the T displays the Color Picker dialog (see Figure 8) .



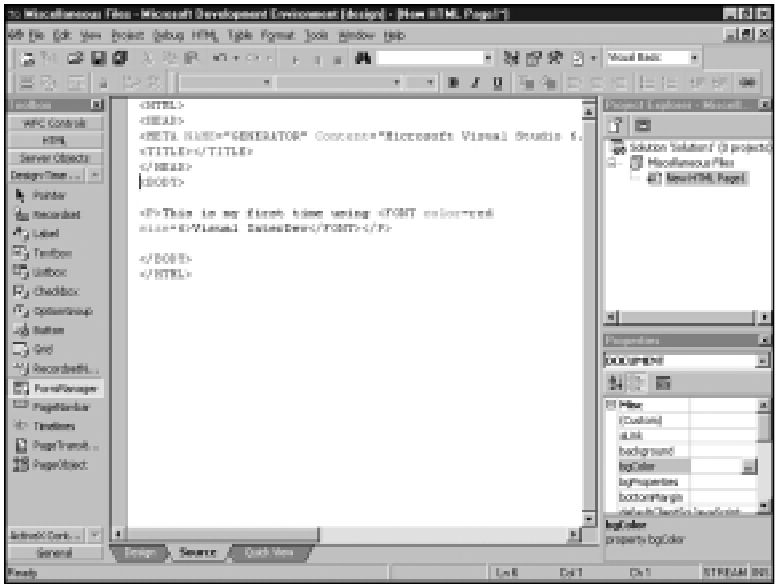
Color: Red (#FF0000)

☑ Use color names

Figure 8: *Visual InterDev allows you to change Text color and page color by using a Color Picker dialog box.*

12. One of the most powerful features of the Visual InterDev HTML editor is that it translates your design ideas in HTML transparently. If you want to view the HTML behind your page's layout, all you need to do is click the Source tab at the bottom of the Visual InterDev HTML page editor (see Figure 9).

As you can see, making a page of HTML in Visual InterDev is simple, yet powerful. The visual layout aspects of Visual InterDev do not stop at text; you can add images, HTML elements, and create hypertext links with nothing more than the drag of a mouse. In addition, Visual InterDev allows you to add ActiveX controls and Java applets to your page if you need to.



```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.
<TITLE></TITLE>
</HEAD>
<BODY>

<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>

</BODY>
</HTML>
```

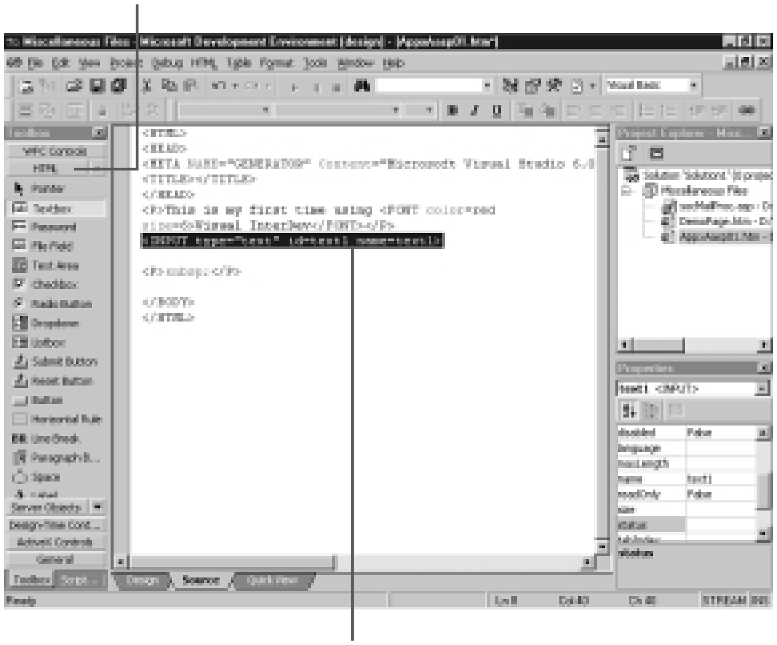
Figure 9: *The Visual InterDev HTML editor allows two way editing—changes you make in the HTML Source will be displayed in the Design view and vice versa.*

Adding an HTML Element to Your Page

Adding an HTML element to your Web page requires no more than dragging a mouse when you use Visual InterDev. When you look at Figure 9, notice the button-like tab with the caption “HTML” in the toolbox on the left hand side of Visual InterDev. When you click this tab, a toolbox full of HTML elements appears (see Figure 10).

Visual InterDev offers you a few ways of adding these HTML elements to your Web page. If you are in Source view (as in Figure 10), all you need to do is place your cursor within the code where you want to add an HTML element. Then, double-click the HTML element in the toolbox or drag it into position from the toolbox to the Source window. Visual InterDev adds the HTML element to the Web page automatically. Figure 10 shows you the code that appears in the Source window when you select a Text element in the toolbox and double-click the element’s icon.

Click here to display the HTML elements.



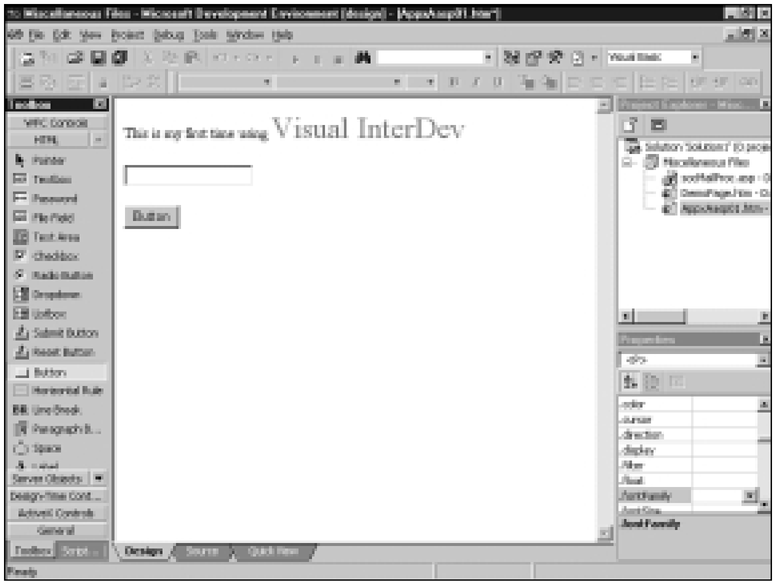
```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0
<TITLE></TITLE>
</HEAD>
<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>
<INPUT type="text" id=text1 name=text1>

<P>&nbsp;</P>

</BODY>
</HTML>
```

You can drag and drop HTML elements from the Toolbox to the code window.

Figure 10: *The HTML tab of the toolbox displays a list of HTML elements that you can drag onto your Web page.*

When you switch to Design view, as shown in Figure 11, Visual InterDev displays the HTML element as it appears in a Web browser.

Visual InterDev allows you to work in Design view to add HTML elements to a Web page. All you need to do is click the mouse cursor at the position in the HTML document where you want the element to appear. Then, you double click the element’s icon in the toolbox or select the icon and drag it over to the document (see Figure 11) .



This is my first time using Visual InterDev

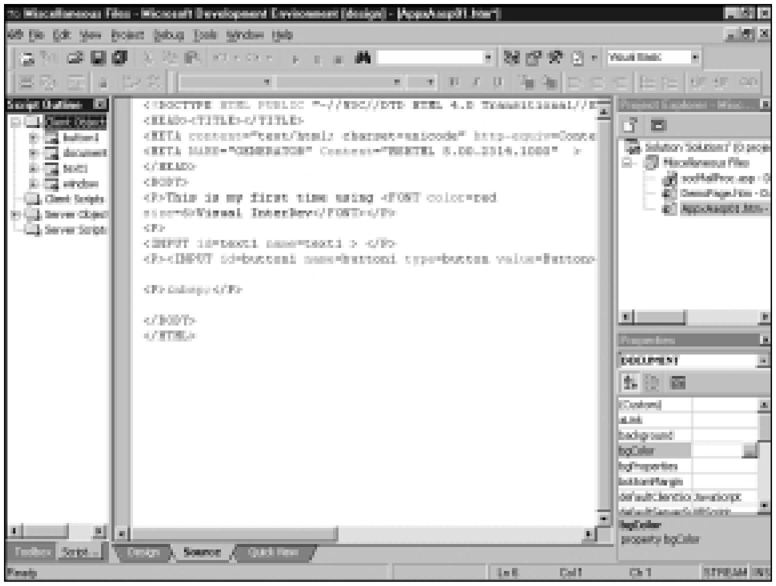
__Figure 11:__ Visual InterDev allows you to add HTML elements to your HTML documents in Design view by dragging and dropping or by double-clicking element icons in the toolbox, just as you did in Source view.

Adding Events to Your Page Using the Script Outline

After you have HTML elements in place, you can program event behavior into your HTML document. Notice in the bottom left-hand corner of the toolbox there is tab captioned “Script…”. Clicking this tab displays the Script Outline.

The Script Outline shows the various elements in your HTML document. These elements coincide with objects in the *Document Object Model*. The Script Outline displays these items when Visual InterDev is in Source view. In Design view, the Script Outline appears blank.

Notice in Figure 12 that Visual InterDev is in Source View and the Script Outline is displayed. Also notice that the Script Outline displays outline nodes for the window and document objects of the page as well as the Text element named `text1` and the Button element name `button1`. We added the Text and Button elements earlier in this demonstration. The window and document objects appear by default. All Web pages have window and document objects. They are part of the Document Object Model.

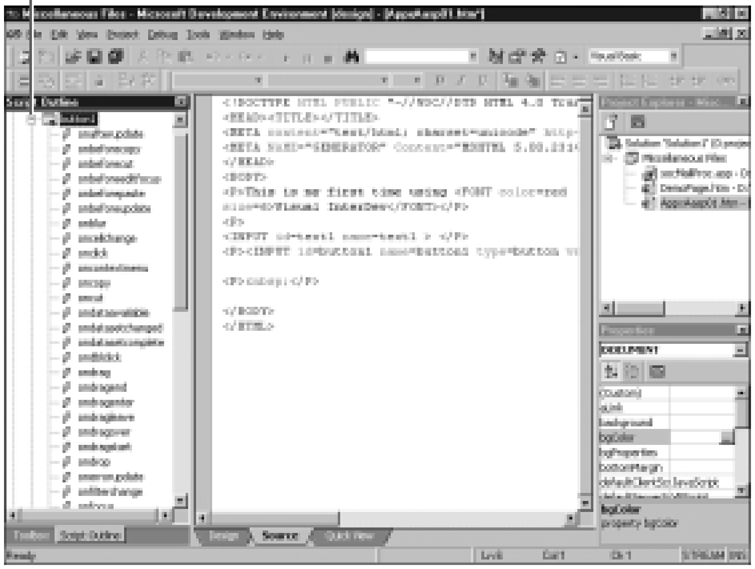


```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//E
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-equiv=Conte
<META NAME="GENERATOR" Content="MSHTML 5.00.2314.1000" >
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
<P><INPUT id=button1 name=button1 type=button value=Button>

<P>&nbsp;</P>

</BODY>
</HTML>
```

Figure 12: *The Script Outline on the left side of Visual InterDev allows you quick access to the elements and objects of a Web Page.*

All elements and objects in the Script Outline have event procedures you can program. All you need to do is click the plus sign to the left of the element or object. When we click the plus sign to the left of the button1 node, as shown in Figure 13, the node expands to show the events procedures we can program.

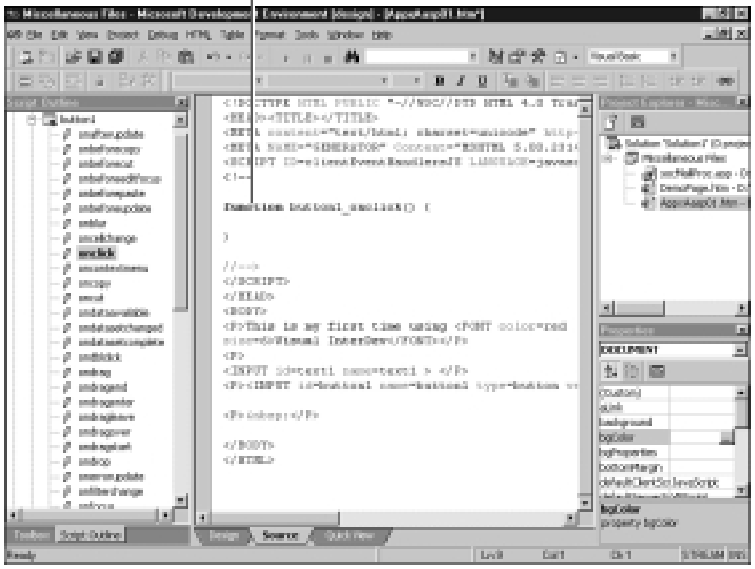
Should you want to program a particular event procedure, double-click the event procedure's icon in the Script Outline. Visual InterDev adds the event procedure's script block to the page's source code automatically. Figure 14 shows the code that results when double-clicking `button1`'s `onclick()` event procedure listed in the Script Outline. Notice that the code is in JavaScript. The default language for the client-side script generated by Visual InterDev is JavaScript.



Script Outline

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Tra
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-
<META NAME="GENERATOR" Content="MSHTML 5.00.231
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
<P><INPUT id=button1 name=button1 type=button va

<P>&nbsp;</P>

</BODY>
</HTML>
```

Figure 13: *The Script Outline allows you to access the event procedures of a given element or object.*



JavaScript

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Tra
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-
<META NAME="GENERATOR" Content="MSHTML 5.00.231
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javasc
<!--

function button1_onclick() {

}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
<P><INPUT id=button1 name=button1 type=button va

<P>&nbsp;</P>

</BODY>
</HTML>
```

Figure 14: *When you use Visual InterDev to add event procedures to your source code automatically, it creates the event procedure in JavaScript.*

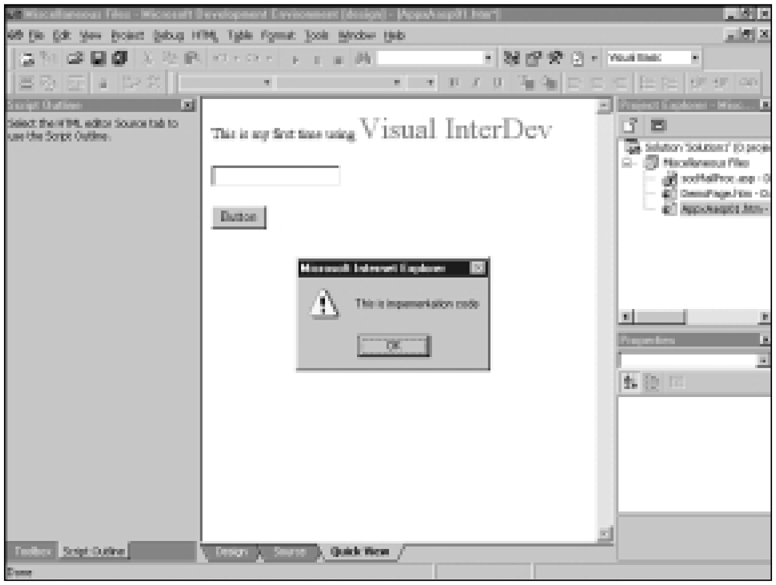
As you see in Figure 14, Visual InterDev creates the event procedure framework for you. All you need to do is fill in the actual implementation code. After you have some working code entered, you can test the code by selecting the Quick View tab at the bottom of the central Visual InterDev pane. Figure 15 shows the result of a button click event procedure in Quick View. We added implementation behavior to the event procedure. The behavior that we added displays an Alert dialog box. The code for the event procedure is shown in Listing 1.

EXAMPLE

Listing 1: The Button Click Event and the Alert Dialog

```
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!—

function button1_onclick() {
    alert("This is impementation code");
}

//—>
</SCRIPT>
```

OUTPUT



Select the HTML editor Source tab to use the Script Outline.

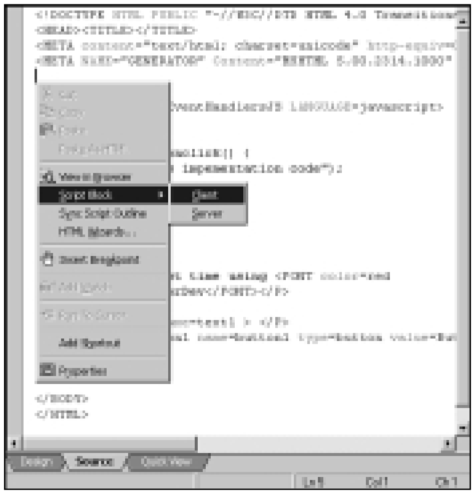
This is my first time using Visual InterDev

This is impementation code

Figure 15: *You can run event procedure scripts that you created in Visual InterDev's Design View.*

Adding Client-Side Script

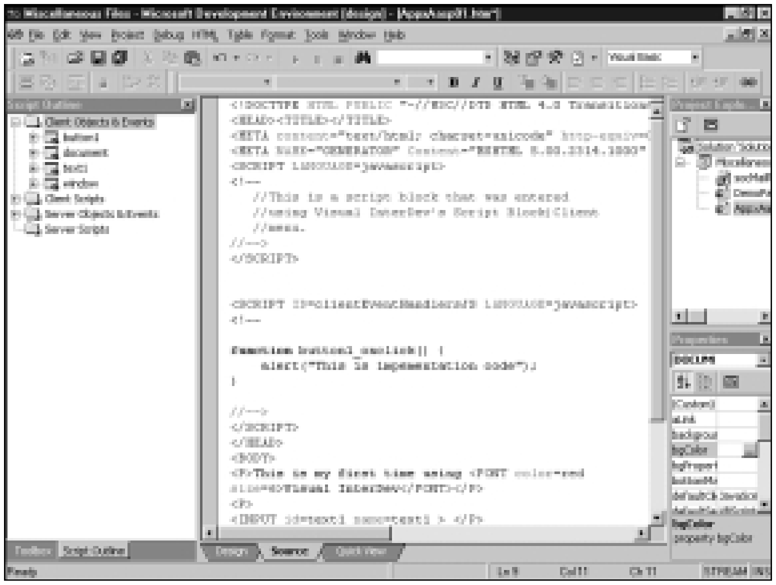
Visual InterDev allows you to add both client- and server-side scripts to your pages easily. To use Visual InterDev to add a script block to your Web page, you must be in Source view. Place your mouse cursor in the portion of your source code where you want to add script and then right-click. Visual InterDev displays a context menu (see Figure 16).



```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional"
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-equiv=
<META NAME="GENERATOR" Content="MSHTML 5.00.2014.1000"

             ventHandlers?5 LANGUAGE=javascript>

             onclick() {
             implementation code");

             t time using <FONT color=red
             rDev</FONT></P>

             me=text1 > </P>
             ml name=button1 type=button value=Bu

</BODY>
</HTML>
```

__Figure 16:__ Visual InterDev allows you to create either client-side or server-side `<Script></Script>` tags by selecting Script Block from the context menu.

When you pick Script Block, Client script from the context menu, Visual InterDev enters a set of `<SCRIPT></SCRIPT>` tags within the source code. By default, the client-side script tags are set to respect the JavaScript language. However, Visual InterDev can be reconfigured to generate VBScript script blocks. After a script block is generated, you add script statements or procedures within the script tags (see Figure 17).



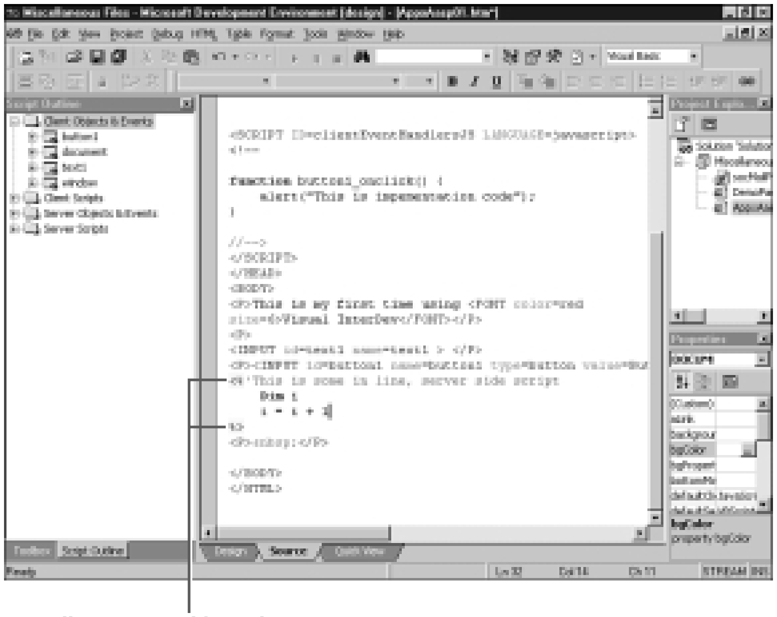
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-equiv=
<META NAME="GENERATOR" Content="MSHTML 5.00.2314.1000"
<SCRIPT LANGUAGE=javascript>
<!--
   //This is a script block that was entered
   //using Visual InterDev's Script Block|Client
   //menu.
//-->
</SCRIPT>


<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!--

function button1_onclick() {
    alert("This is implementation code");
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=4>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
```

__Figure 17:__ The default language for client-side script is JavaScript and VBScript for server-side script.

Using Visual InterDev with Server-Side Script

Visual InterDev has the ability to create server-side script blocks for you automatically. The process is very much the same as creating a client-side script block, except that you select Server from the Script Block context menu (refer to Figure 16).

You need to create inline, server-side code manually. When you create server-side script inline, Visual InterDev highlights the `<% %>` tags in yellow. Highlighting these tags allows you to keep track of your inline code blocks easily (see Figure 18).



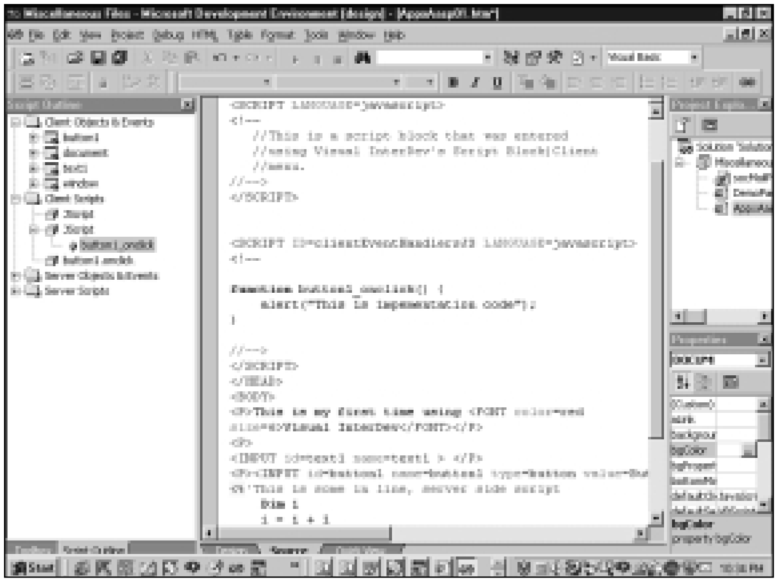
```
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!--

function button1_onclick() {
    alert("This is implementation code");
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
<P><INPUT id=button1 name=button1 type=button value=Bu
<%'This is some in line, server side script
    Dim i
    i = i + 1
%>
<P>&nbsp;</P>

</BODY>
</HTML>
```

Inline, server-side script tags

Figure 18: *Visual InterDev highlights the inline, server-side script tags,* `<% %>`*, in yellow.*

Lastly, Visual InterDev keeps track of your page's scripts and scripted functions in the Script Outline. See Figure 19 for an example.

Notice that the Client Scripts folder in the Script Outline on the left side of the Visual InterDev work area has been expanded. Also, the expanded folder has sub nodes with the captions "JScript" and "button1.onlick". The "JScript" captions indicate the presence in your page of `<SCRIPT>` tags that are written in JavaScript. The "button1.onclick" caption indicates a directive in the `<INPUT Type=Button>` tag to use the `onclick` method when button1 is clicked. (Don't worry too much about the `button1.onclick()` directive. Visual InterDev makes entries such as this in the `<INPUT>` tags automatically when you double-click an object or element's event procedure in the Script Outline. Visual InterDev is satisfying a syntax requirement that allows JavaScript to assign an event procedure to the button click.)



```
<SCRIPT LANGUAGE="javascript">
<!--
   //This is a script block that was entered
   //using Visual InterDev's Script Block|Client
   //menu.
//-->
</SCRIPT>


<SCRIPT ID=clientEventHandlersJS LANGUAGE="javascript">
<!--

function button1_onclick() {
    alert("This is implementation code");
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=4>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
<P><INPUT id=button1 name=button1 type=button value=But
<% 'This is some in line, server side script
    Dim i
    i = i + 1
```

Figure 19: *Visual InterDev lists <SCRIPT> tags, function references, and functions within <SCRIPT> tags in the Script Outline.*

When you look at the second JScript node in the Script Outline, you'll notice that the `button1_onclick` event procedure, which Visual InterDev created earlier, is listed beneath the node. `button1_onclick` is a function. The Script Outline lists all the functions within a given `<SCRIPT>` tag. All of these listings in the Script Outline relate to either a `<SCRIPT>` tag or a function within a `<SCRIPT>` tag within the source code. If you double-click a `<SCRIPT>` tag or function entry in the Script Outline, Visual InterDev places the cursor at the position of the `<SCRIPT>` or function you selected.

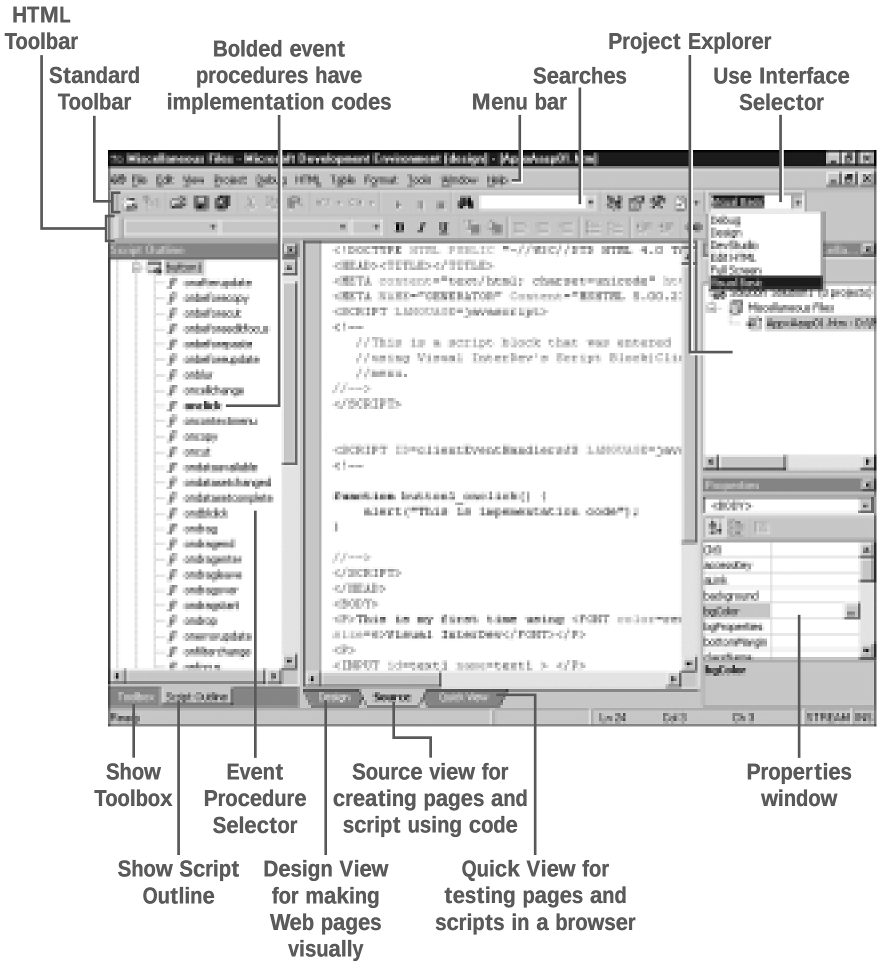
Now that you have become acquainted with Visual InterDev by seeing some operational work, let's look at some of the details of the Visual InterDev Integrated Development Environment.

Understanding the Integrated Development

Visual InterDev provides a programming workbench you use to create standard HTML pages as well as Active Server Pages. The formal name for the workbench is the Integrated Development Environment (IDE). You use the IDE to create new pages and add work pages. The IDE provides Project Wizards to create multi-page Web applications and a set of tools to work with ActiveX components in your page's scripts. In addition, the IDE provides debugging tools that help find and fix errors in client-side and server-side scripts.

To say that the Visual InterDev programming environment is comprehensive is an understatement. Unlike other development environments that are dedicated to one language such as Visual Basic or C++, Visual InterDev is designed to support many aspects of Internet development. Thus, you can use it to do HTML, Javascript, VBScript, and standard Java programming. In addition, Visual InterDev has the capability to allow you to do data access programming using Structured Query Language (SQL). This is a big product designed to meet the needs of a broad range of developers.

To make the diversity of Internet development work that you can do with Visual InterDev easier, the product offers a variety of User Interface configurations. The following sections give you an overview of these configurations.


HTML Toolbar
Standard Toolbar
Bolded event procedures have implementation codes
Menu bar
Searches
Project Explorer
Use Interface Selector
Show Toolbox
Event Procedure Selector
Source view for creating pages and script using code
Properties window
Show Script Outline
Design View for making Web pages visually
Quick View for testing pages and scripts in a browser

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Tr
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" ht
<META NAME="GENERATOR" Content="MSHTML 5.00.2
<SCRIPT LANGUAGE=javascript>
<!--
   //This is a script block that was entered
   //using Visual InterDev's Script Block|Cli
   //menu.
//-->
</SCRIPT>


<SCRIPT ID=clientEventHandlersJS LANGUAGE=java
<!--

function button1_onclick() {
    alert("This is implementation code");
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=
size=4>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
```

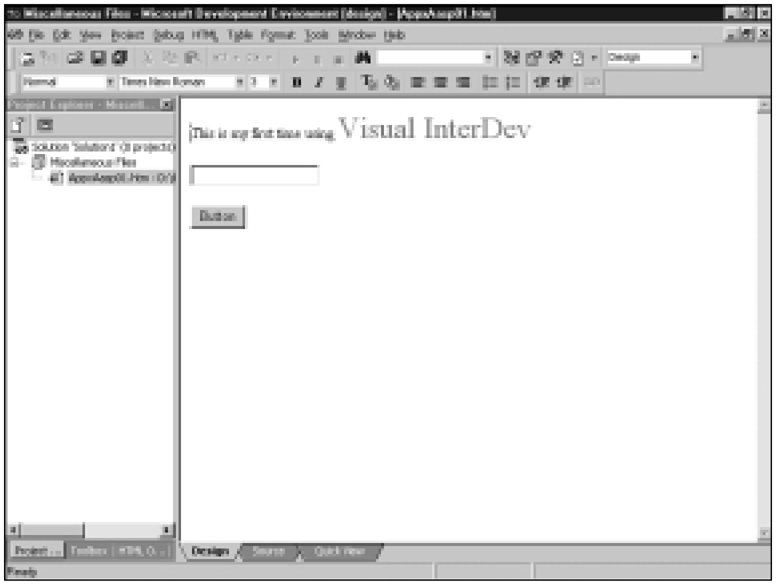
__Figure 20:__ Visual InterDev Integrated Development Environment provides a full set of tools and a variety of working environments.

Visual Basic

Figure 20 shows you the Visual Basic user interface for Visual Interdev. This interface uses a Visual Basic type of layout. You have a Toolbox/Script Outline pane on the left and a Project Explorer and Properties window on the right. The center pane shows your Web page's layout or code window, depending on the tab you choose at the bottom of the pane. Use the Project Explorer to move between the pages that make up a Visual InterDev project. Use the Properties window to edit the different properties in a Web page. These properties can be from a DOM object or an HTML element.

The Design User Interface

The Design User Interface (UI) shown in Figure 21 is made for doing Web page layout visually. The Project Explorer and Properties Window are hidden, giving you a bigger central pane in which you can do layout or write code. The DevStudio UI is similar to the Design UI.

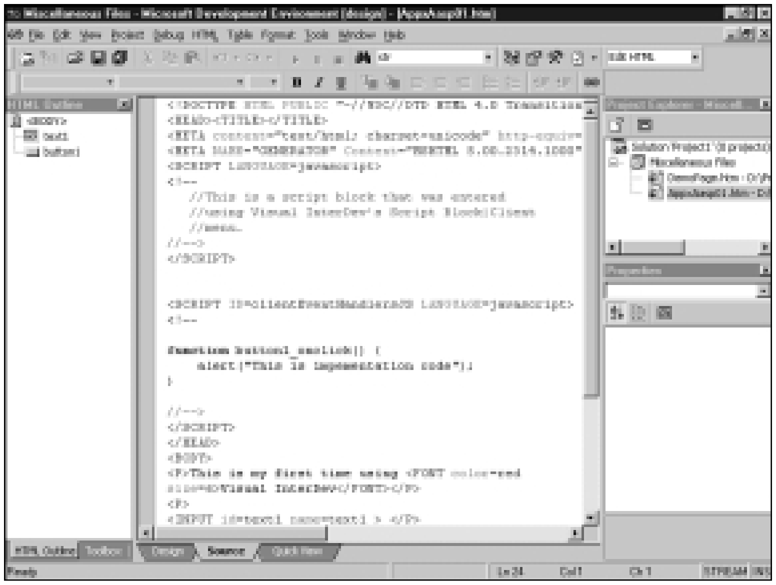

Miscellaneous Files - Microsoft Development Environment [design] - [AppsAasp01.htm]

This is my first time using Visual InterDev

Figure 21: *The Design UI is made for doing Web page layout.*

The EditHTML Layout

The EditHTML layout, as shown in Figure 22, shows the Document outline window on the right as well as the Project Explorer and Properties Window. In addition, the HTML Outline tool is presented in the Toolbox pane. The HTML Outline tool shows you the various HTML elements on the Web page in the central pane. When you click an element in the HTML outline, that element becomes selected in the central pane when the pane's Design tab is selected. If you select the Source view tab to expose your HTML code, clicking an element in the HTML Outline brings you to the place in the HTML code where the element is positioned.

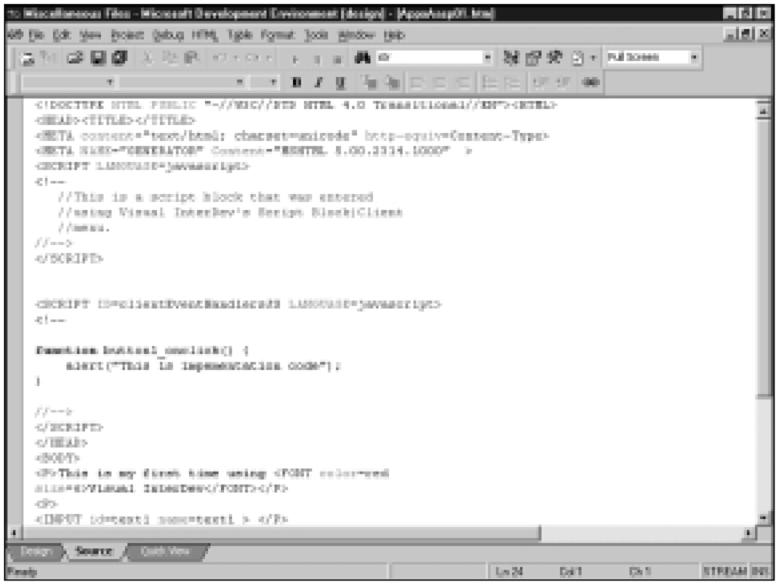

Miscellaneous Files - Microsoft Development Environment [design] - [AppsAsp01.htm]
File Edit View Project Debug HTML Table Format Tools Window Help
Edit HTML
HTML Outline
<BODY>
text1
button1
HTML Outline
Toolbox
Design
Source
Quick View
Project Explorer
Solution 'Project1' (0 projects)
Miscellaneous Files
DemoPage.htm
AppsAsp01.htm
Properties
Ready
Ln 24
Col 1
Ch 1
STREAM

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-equiv=
<META NAME="GENERATOR" Content="MSHTML 5.00.2314.1000"
<SCRIPT LANGUAGE=javascript>
<!--
   //This is a script block that was entered
   //using Visual InterDev's Script Block|Client
   //menu.
//-->
</SCRIPT>


<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!--

function button1_onclick() {
    alert("This is implementation code");
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=4>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
```

Figure 22: *The HTML Outline UI allows you to work with elements in your Web page easily.*

The FullScreen User Interface

Figure 23 shows the FullScreen UI. This user interface hides all panes and windows except the central pane. This UI layout is good for reviewing code and viewing large areas of a Web page.


Miscellaneous Files - Microsoft Development Environment [design] - [AppsAsp01.htm]
File Edit View Project Debug HTML Table Format Tools Window Help
Full Screen
Design
Source
Quick View
Ready
Ln 24
Col 1
Ch 1
STREAM INS

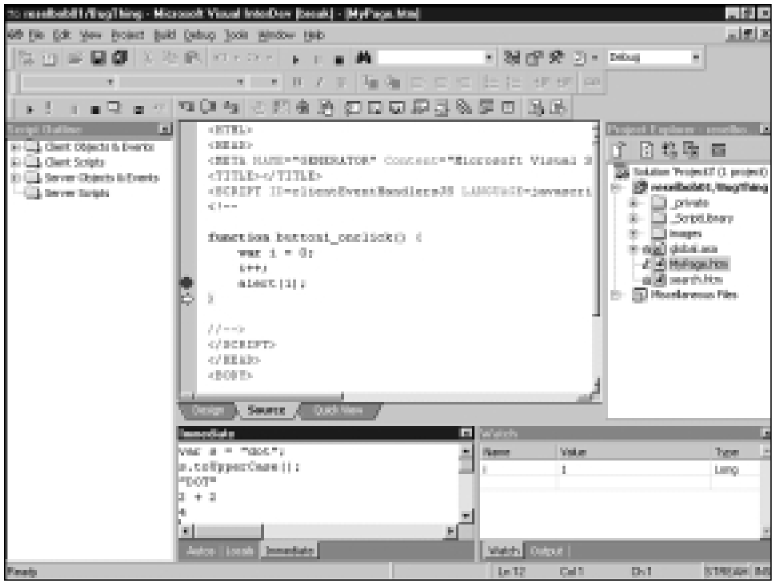
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN"><HTML>
<HEAD><TITLE></TITLE>
<META content="text/html; charset=unicode" http-equiv=Content-Type>
<META NAME="GENERATOR" Content="MSHTML 5.00.2314.1000" >
<SCRIPT LANGUAGE=javascript>
<!--
   //This is a script block that was entered
   //using Visual InterDev's Script Block|Client
   //menu.
//-->
</SCRIPT>


<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!--

function button1_onclick() {
    alert("This is implementation code");
}

//-->
</SCRIPT>
</HEAD>
<BODY>
<P>This is my first time using <FONT color=red
size=6>Visual InterDev</FONT></P>
<P>
<INPUT id=text1 name=text1 > </P>
```

Figure 23: *The FullScreen UI hides all windows and panes except the central pane.*

The Debug User Interface

The Debug UI, as shown in Figure 24, is the most complex of the User Interfaces. You use Debug to work with a project's code in a very detailed manner. The panes at the center and lower-right of the Visual InterDev IDE allow developers writing scripts and Java programs to observe and test variables and expressions as they are running in code.

For example, when you look in the Immediate pane in the lower center of the Visual InterDev IDE in Figure 24, you'll see that you can test expressions before you put them into code. In this example, the developer made a lowercase string appear in uppercase and added the value of `2 + 2`. Both expressions were written in JavaScript. In addition, on the right side pane, which is called the Watch pane, the developer was able to observe the value of `i` change as the code moved from line to line in the main code pane.



```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual S
<TITLE></TITLE>
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascri
<!--

function button1_onclick() {
    var i = 0;
    i++;
    alert(i);
}

//-->
</SCRIPT>
</HEAD>
<BODY>
```

```
var s = "dot";
s.toUpperCase();
"DOT"
2 + 2
4
```

Name	Value	Type
i	1	Long

Figure 24: *The Debug UI allows a developer to take advantage of the many debugging tools that ship with Visual InterDev.*

Creating Web Projects Using Visual InterDev

A Visual InterDev project contains those HTML Pages and Active Server Pages that make up a distinct Web site. A Visual InterDev project can be expanded to include a standalone Java Program and a Java Applet as well as a data access code. When you start up Visual InterDev you are asked to select a type of Web project (refer to Figure 2). If you are using Visual InterDev to make a Web project, you select New Web Project from the New Project menu. Selecting New Web Project activates a Wizard that walks you through the steps necessary for creating a framework for your Web application. The Wizard asks you to identify an Internet server upon which you will be doing your Web development. The Wizard then creates a number of "helper" files on the Internet server. Also, the Wizard will prompt you to select a style for the pages of your Web site, should you so desire. The "helper" files allow you to treat scripts within pages in your Visual InterDev project as distinct objects, which you can identify, reference, and use throughout your project. This is the essence of the Scripting Object Model, which you read about earlier. The Scripting Object Model (SOM) is a powerful enhancement for Web development, but not a necessary one. It is quite possible to create useful web projects and never use the SOM.

TIP

For more information about the Scripting Object Model, go to the Microsoft Developer's Network online at `http://msdn.microsoft.com/library/devprods/vs6/vinterdev/vidref/viconthescriptingobjectmodel.htm`.

Figure 25 shows the way the IDE appears when you create a New Project. In addition, the figure shows the context menu that appears when you right-click an item in the project. Please notice the folders `_private`, `_ScriptLibrary`, and `Images`. These folders were created by Visual InterDev. These are the "helpers" to which we referred earlier. Also notice the context menu allows you to declare a page in your Web project as a starter page. After you declare a starter page, you can run the project in Debug mode. Running in debug mode is a little tricky. Certain conditions need to be in place. (We'll cover these conditions in the section "Setting Up the Debugger" that follows.)

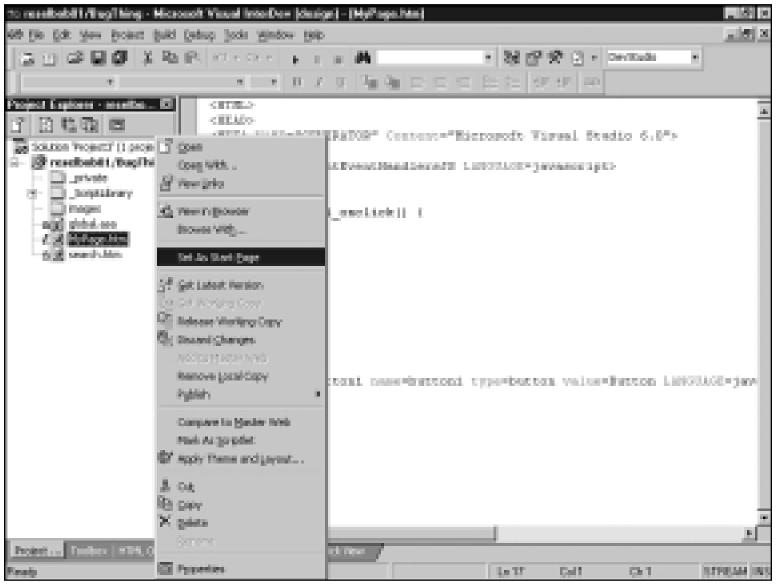


Figure 25: *Visual InterDev creates the framework for a Web project for you. You can add pages to your project, and then select one to be your start page.*

Creating a Site Design

Visual InterDev comes with a Site Diagram tool that allows you to create a Visual Site map for your Web project. After you create the site map, Visual InterDev creates HTML and ASP pages that haven't any content and adds them to your project. You access the Site Diagram tool from the Project, Add Item menu. Figure 26 shows you the Add Item dialog box from which you select the Site Diagram tool.

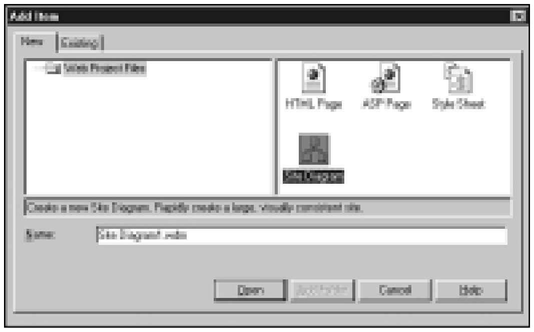


Figure 26: *You select the Site Diagram for the New Item dialog box.*

You can add pages to the Site Diagram by right-clicking the Site Diagram work area. A context menu appears from which you add an HTML Page, an Active Server Page, or an existing file (see Figure 27).

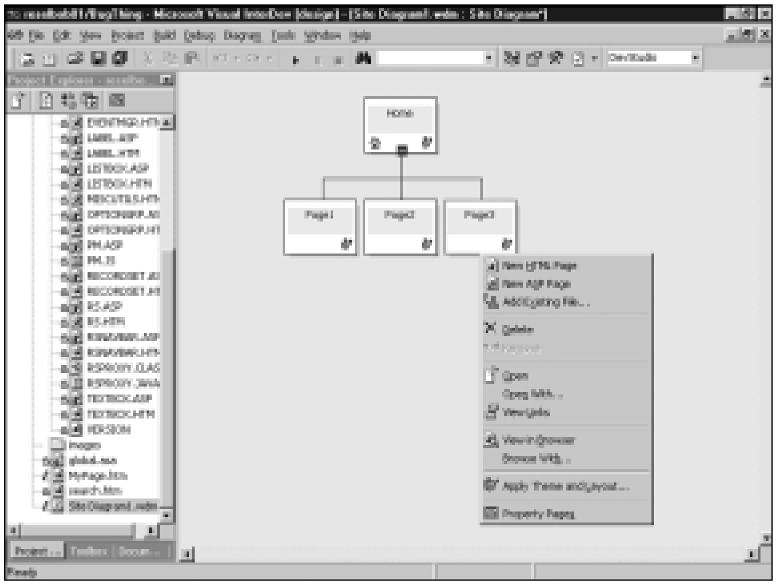


Figure 27: *The Visual InterDev Site Diagram tool allows you to create your Web project site visually.*

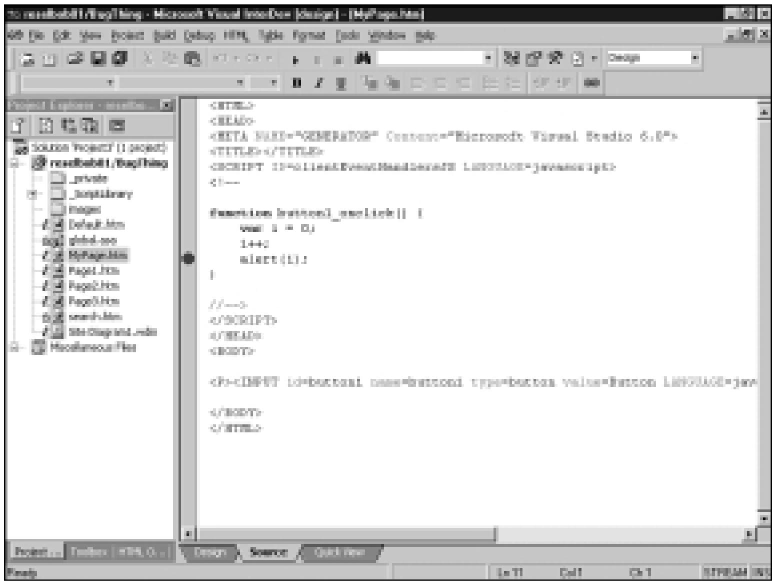
When you close the Site Diagram work area, Visual InterDev saves the files you created in the diagram to your hard drive and adds them to your Web project.

Setting Up the Debugger

Visual InterDev allows you to debug client-side or server-side script. Debugging is a little tricky. The first thing you need to remember when it comes to debugging is that you can only debug pages that are part of a Visual InterDev project. The IDE will not allow you to debug pages outside of a project.

After you have a Web Project set up, debugging client-side script is straightforward. Be sure a Start Page has been identified in the Project Explorer. Then, strike the F5 key or click the Start arrow at the Debug toolbar. Visual InterDev activates Internet Explorer and you are on your way. However, to actually do any sort of debugging, you usually set a breakpoint.

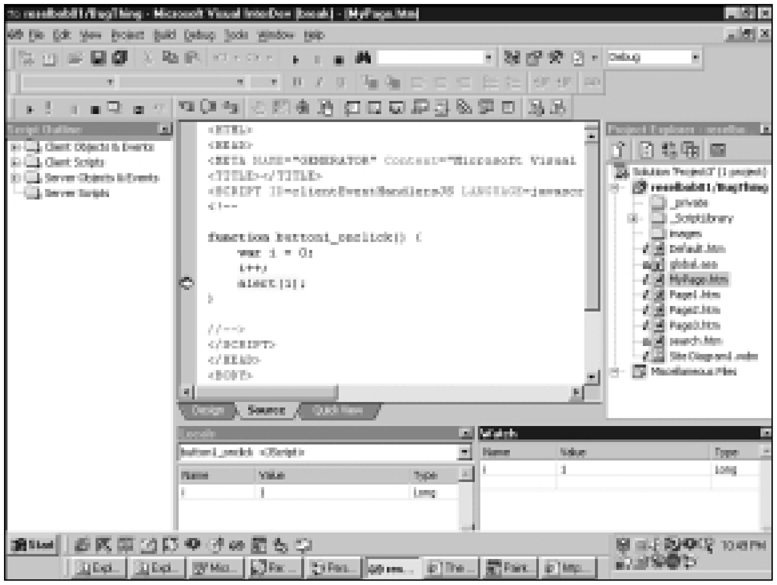
Set a breakpoint in your Web project by clicking in the left margin of the code pane in which your script appears. When you click in the margin, a red dot appears (see Figure 28). Your code will run until it comes to this line.



```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual Studio 6.0">
<TITLE></TITLE>
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascript>
<!--

function button1_onclick() {
    var i = 0;
    i++;
    alert(i);
}

//-->
</SCRIPT>
</HEAD>
<BODY>

<P><INPUT id=button1 name=button1 type=button value=Button LANGUAGE=jav

</BODY>
</HTML>
```

Figure 28: *You set a break point by clicking in the left margin of the code pane.*

When your code pauses, as shown in Figure 29, an arrow will appear over the break point dot, and the line at which the code pauses is highlighted in yellow. When your code is in Break Mode you can use tools such as the Watch and Local tabs in the Debug panes to examine the values of the variables in your code.



```
<HTML>
<HEAD>
<META NAME="GENERATOR" Content="Microsoft Visual
<TITLE></TITLE>
<SCRIPT ID=clientEventHandlersJS LANGUAGE=javascr
<!--

function button1_onclick() {
    var i = 0;
    i++;
    alert(i);
}

//-->
</SCRIPT>
</HEAD>
<BODY>
```

button1_onclick <Script>

Name	Value	Type
i	1	Long

Name	Value	Type
i	1	Long

Figure 29: *When your code goes into Break Mode, the Debug panes become active.*

Debugging an Active Server Page

If you want to configure either Personal Web Server or Internet Information Server to accommodate server-side debugging, you should do so in a mode that is called local server debugging. This means you are going to run your development machine as a server and that your server-side scripts will run upon your desktop server. You have to do a little work to get a local server debugging up and running. The process for doing this is well documented on the Microsoft Developers Web site and can be found at

```
http://msdn.microsoft.com/library/techart/
➥msdn_videbugging.htm#videbugging_localserver
```

Server-side debugging is a tricky ordeal. The best way to do it is on a local server. It's best for you and best for your development team. Local server debugging doesn't tie up any workstation except for yours. After you get your scripts debugged on your local server, deploying them to the production box is a simple enough affair. If you are working on a large-scale project, local server debugging is the only way to go.

Visual InterDev does offer the capability to do what is called remote debugging. However, you need to do a lot of setup work to get the server to respond to the wishes of the debugging client, and only one client at a time can access a server for debugging. It's not that hard for an experienced developer, but for someone starting out, it can be a real chore. To that end, it's something you can aspire to do. You'll need to make sure your copy of Visual InterDev is the Enterprise Edition.

For a detailed look at how to do remote debugging, press the F1 key to display the MSDN Visual Studio Library that ships with Visual InterDev and is installed on your computer. Then, look for "Remote Debugging" in the Index of the Library. If you follow the instructions carefully, you should be able to have the Remote Debugger up and running in a short amount of time.

Moving On

The intent of this tutorial is to give you an introduction to Visual InterDev. As you can probably guess by now, Visual InterDev is a big tool.
It can do a lot for a wide variety of developers. The tool is useful to anyone from the hard core Java-head to a seasoned page designer who uses JavaScript and VBScript. The tool will save you time, and in the long run, save you money. But as with any tool that has such a broad reach, it takes time to learn. If you give yourself the time, you'll find the tool will provide a noticeable increase in your productivity while not sacrificing any of the intrinsic fun of developing dynamic pages for the Web.