The scriptlet zone

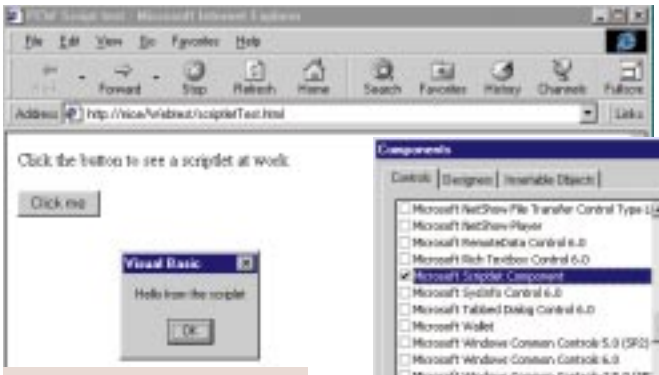
Why bother with batch files when you can run VB Script right off the desktop? Tim Anderson explores the world of scripts and scriptlets.

If you have an idle moment, try this: right-click the Windows 95 or NT 4.0 desktop, choose New Text Document and then double-click to open the new file in Notepad. Type some exotic code like:

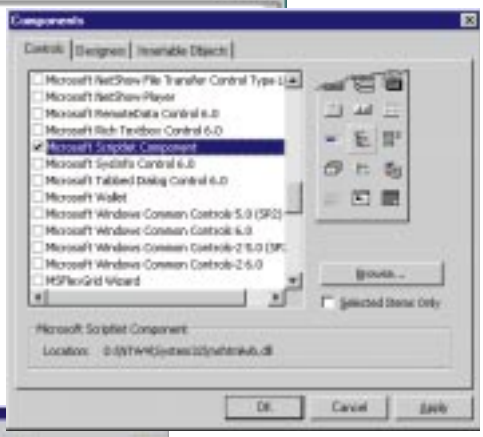
```
Msgbox "No more batch files"
```

Save the document and then rename it with a .VBS extension. Double-click and a message should appear. If not, no problem. You need the Windows Scripting Host, which is free and is an option in Windows 98 setup, or in the NT 4.0 Option Pack, or as a download from Microsoft's web site. Once installed, you can run either VBScript or JScript straight from the desktop [Fig 1] — a neat trick which is evidence that Microsoft is serious about scripting.

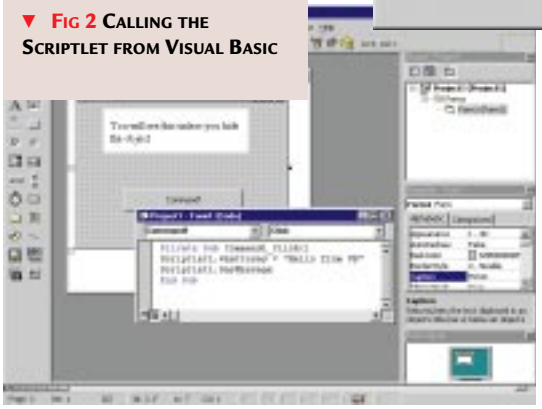
VBScript started life as a rival to Netscape's JavaScript in Internet Explorer 3.0, but has developed to the point where a full COM automation scriptlet, or script object, can be created with Notepad and a few lines of code. How useful this is remains to be seen, but it does mean that VB skills can now be used anywhere in Windows.



▲ **Fig 1 The end of the batch file? Now you can run VBScript straight from the desktop**



▼ **Fig 2 Calling the Scriptlet from Visual Basic**



▲ **Fig 3 Adding the Scriptlet container to a Visual Basic application**

Windows scripting

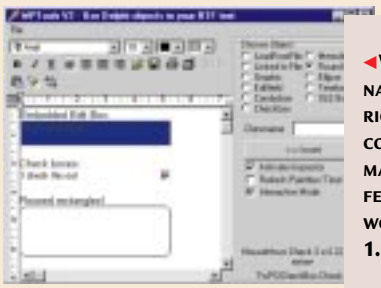
Windows scripting is based entirely on COM, ActiveX, or whatever you like to call Microsoft's object technology. At its heart is an ActiveX scripting interface which sits between client applications such as Internet Explorer or the Windows Scripting Host, and script engines such as VBScript and JScript. If you feel so inclined, you can create your own script engine (a Perl script engine is available from a third party).

You may be wondering which language to use. VB is easy, performs well, and is most likely to be familiar to Windows users. JScript is based on JavaScript and conforms to a standard laid down by the ECMA (European Computer Manufacturers Association). JScript is the best choice for web pages because used with care it will run in NetScape as well as Internet Explorer, although in this case you will need to

SNIPPETS

The Sax Basic Engine lets you add macro programming to your VB application or any that supports COM automation. The new version is VBA-compatible and makes it easy to add custom extensions. By using the Active Template Library, Sax has halved runtime size and removed dependency on VB or MFC runtime files.

● Delphi users should take a look at Word Processing Tools v2, a native Delphi component for working with rich text. It's useful for 16-bit Delphi users or for situations where Delphi's rich text control is not enough. Version 2.0 supports true WYSIWYG printing, embedded objects and full undo. There are some rough edges, particularly in the documentation, but on the plus side the author is responsive to problems, full source is available, and it is inexpensive. Delphi 4.0 support was not available at the time of writing.



◄**WPTools is a native Delphi rich text control with many additional features. It even works in Delphi 1.0**

[FIG 4]

A simple scriptlet

```
<html>
<head>
<title>Message➧
 Scriptlet</title>
</head>
<body>
<p>You will see this unless
you hide the object </p>
<SCRIPT➧
 LANGUAGE="VBScript">
Dim public_whattosay
Sub public_sayMessage()
MsgBox(public_whattosay)
End Sub
</SCRIPT>
</body>
</html>
```

(Key : ➧ Code continued on next line)

avoid its ActiveX extensions. Internet Explorer 3.0 can run scripts but although this opens up great possibilities for dynamic web pages or web applications, it is a messy way to program.

It is hard to avoid bugs if you have little sections of script strewn all over an HTML page, and if you want to re-use code on another page the only option is copy and paste.

Scriptlets, introduced in Internet Explorer 4.0, are an attempt to bring scripts under control. Instead of including all your script in the current page, you can create dedicated scriptlet pages that become a code library. You can then reference those pages using the Object tag, set properties and call methods. Figs 4 & 5 show the simplest example. Although in this example the scriptlet has been hidden using the visibility style tag, there is no problem with having it display a user interface. Looking at it one way, you are simply embedding one HTML page inside another. Looking at it another way, a scriptlet is the easiest way to create an ActiveX control.

Container control

To prove the point, Visual Basic 6.0 comes with a scriptlet container control that lets you include a scriptlet in a VB application [Fig 2]. Yes, an ordinary application that runs from the desktop.

[FIG 5]

A simple client page using a scriptlet

(Requires Internet Explorer 4.0)

```
Note: You must change the Data tag in the OBJECT element to
the URL of where you placed the page in Listing 1. In this
example they are in a local directory, D:\WebTest.
<html>
<head>
<title>PCW Script test</title>
</head>
<body>
<p>Click the button to see a scriptlet at work</p>
<BUTTON TYPE="BUTTON" LANGUAGE="VBScript"➧
 onclick="TestScriptlet">
<p>Click me </BUTTON> </p>
<OBJECT ID="ScriptObj" TYPE="text/x-scriptlet"➧
 Data="file://d:\WebTest\pcwScriptlet.html"
style="visibility:hidden">
</OBJECT>
<SCRIPT LANGUAGE="VBScript">
Sub TestScriptlet()
ScriptObj.whattosay = "Hello from the scriplet"
ScriptObj.sayMessage
End Sub
</SCRIPT>
</body></html>
```

(Key : ➧ Code continued on next line)

[FIG 6]

Talk to the scriptlet

```
Private Sub Command1_Click()
Scriptlet1.whattosay = "Hello from VB"
Scriptlet1.SayMessage
End Sub
```

[FIG 7]

Word the WSH way

```
setwd=createobject("Word.application")
wd.Visible = Truewd.Documents.Add
Set wd = Nothing
```

Here is how you can include your scriptlet:

1. Start a new, standard VB application, right-click the toolbox, choose Components, and select Microsoft Scriptlet Component [Fig 3].
2. Add a Scriptlet control to the form.
3. Set its URN property to point to the scriptlet page. This can be either a file location or an ht address.
4. Now you can write code that talks to the scriptlet [Fig 6].

Mein Host

Like Internet Explorer, the Windows Scripting Host (WSH) is a client application for the ActiveX Scripting interface. The difference is that WSH is small and lightweight — so much so, that you will not notice a significant loading time. Scripts such as the "no more batch files" example (*mentioned at the beginning*) have a file association with WSH, so when you open them WSH runs the script.

You also get the benefit of the Wscript object which is a COM interface to the WSH. Crucially, this has both CreateObject and GetObject methods through which you can get at any automation objects available on your system. Fig 7 shows how you could start a new Word document the WSH way.

Save this as a text file with a .vbs extension, double-click and, all going well, Word will open. I know there are easier ways to open Word but the point is that once you have a reference to the Word object, you have full control over Word with interesting automation possibilities.

This also means that when you want to know how to do something with WSH

Questions & answers

Q I write a lot in VB, and over the years I've built a large repository of code which I use regularly. I believe it's possible to create an ActiveX DLL which can hold all these routines globally. But the $64,000 question is, how? I have tried, but each time I try to call a DLL'd function from within a project, I can't.

GRANT MATTHEWS

***A** Here's how you can do this with a VB DLL. Note that you need the Professional or Enterprise version of VB.*

1. Create a new project, ensuring that the Project Type is ActiveX DLL. DLL functions need to be in a class module so, if you do not have one in your project, add one from the Project menu. The class module itself has properties.

2. Set the Instancing to Multiuse.

3. In Project properties, set the name as required (say, MyFuncs). Now add your functions to the class module. Fig 8 shows an example called GetSquareRoot. Choose Make MyFuncs.dll from the File menu to compile the project.

4. Start a new Standard project. From the project menu choose References, and in the dialog check MyFuncs.

5. Write code like that in Fig 9 to use the functions in the DLL. There are a few points to note. First, you have to create an instance of the ActiveX object before calling its functions. This is achieved with New or, alternatively, CreateObject. You could do this with a global variable of type MyFuncs and instantiate it when your project starts up, if you want to use your code library from any point. Second, use Add Project to load both the DLL project as well as your client project, so that you can debug the code. Third, when you deploy your dll to other machines, you will need to register it either manually with RegSvr32, or by using the VB setup wizard which does this for you. Finally, once you get to the point where applications are using the DLL in earnest, you need to keep future versions compatible with your first version, and to use binary compatibility when you compile. See also, Creating an ActiveX Dll in the VB Components Tools Guide.

[FIG 8] Class module

```
Code for the class module in MyFuncs.dll:
Function GetSquareRoot(num As Double) As
 Double
GetSquareRoot = 0
If num < 0 Then
Err.Raise vbObjectError + 600,
 "MyFuncs.Class1", "Invalid number"
Else
GetSquareRoot = Sqr(num)
End If
End Function
```

(Key : → Code continued on next line)

[FIG 9] DLL client app

```
Code for an ActiveX DLL
 client application:
Private Sub Command1_Click()
Dim mf As New MyFuncs.Class1
Dim num As Double
num = Val(Text1.Text)
MsgBox "The square root is: "
 + Str(mf.GetSquareRoot(num))
End Sub
```

(Key : → Code continued on next line)

(how to do the equivalent of Net Use in a batch file, for instance) you need to find a COM object model that exposes the particular feature. The Wscript object itself provides many of these administrative tools and instead of Net Use you could try WshNetworkObject.MapNetworkDrive. It is documented in the Platform SDK — the set of online documents that describe the Windows API.

Sometimes you may want to run a script without user intervention. CSCRIPT.EXE is a command-line version of WSH which can suppress all display of user prompts and script errors if you use the appropriate switches.

You can also customise how the graphical version runs scripts, by right-clicking a script file and choosing Properties. This creates a file with a .whs extension, and if you run this instead of opening the script directly, then the chosen properties will be applied.

A neat feature of using WSH instead of batch files is that you can step through the code in a debugger. You can download a free script debugger from Microsoft, or use Visual InterDev for full-featured debugging with watch windows and the ability to edit on-the-fly.

Other scriptlets

Finally, there is yet another scriptlet technology, now available in beta. These are called Server scriptlets or XML scriptlets (the name may change).

Server scriptlets are full COM controls with a ProgID and optionally a ClassID. The ProgID is the name of the scriptlet object in dot notation, while the ClassID is a GUID (Globally Unique Identifier).

The easiest way to experiment with XML scriptlets is by visiting the Microsoft scripting web site and downloading the Scripting wizard. You will also find a control that lets you include a scripting capability in your own applications. Note that you should run scriptlets on the server if your web site is open to the whole internet, as this is Microsoft technology that will not work on browsers other than Internet Explorer 4.0 or higher. Another cautionary note is that scripts are neither as fast nor robust as compiled code. All the same, scriptlets and scripting makes impressive use of COM and are great for quick-and-easy tasks as well as for structuring and re-using script on web sites.

PCW CONTACTS

Tim Anderson *can be contacted via the PCW editorial office (address, p10) or email* visual@pcw.co.uk

Microsoft Scripting www.microsoft.com/scripting/

Sax Basic Engine *£381.88 (£325 ex VAT) from Contemporary Software 01344 873434* www.contemporary.co.uk/

Word Processing Tools *For information, see* http//members.aol.com/jziersch/