CHAPTER 37

Using Scripts

In This Chapter

Installing and accessing scripts

Exploring JavaScript

Exploring AppleScript

Exploring VBA

Writing scripts

Learning more about scripting

Scripting automates many features in InDesign — it's essentially a way to program InDesign to do specific actions. Because InDesign uses standard script languages, you can also run scripts that work with multiple programs in concert, including InDesign. (All the applications must support the same scripting language, of course.) For example, you might use scripts to automate database publishing, such as to run a database search, export data to a text file, import that file into InDesign, and then apply the appropriate formatting.

InDesign supports three scripting languages:

- On both Mac and Windows, it supports JavaScript.
- On the Mac only, it supports AppleScript.
- On Windows only, it supports Visual Basic for Applications (VBA).

Because of this, I recommend you use JavaScript wherever possible, so your scripts can work in cross-platform environments. InDesign doesn't force you to choose just one scripting language, so you could keep using old AppleScript or VBA scripts created for previous versions of InDesign, as well as new scripts written in JavaScript.

JavaScript support is new to InDesign CS.

As you become comfortable with scriptwriting, you're also likely to discover that virtually everything you do with InDesign is a repetitive task. The more you can free yourself of this kind of work by using scripts, the more time you have to be creative. The possibilities are endless. But before you get too excited, remember that scripting *is* programming, so most layout artists stay clear, using scripts only if they have a programmer available to write them.

Using Scripts

Accessing scripts is easy — they show up in the Scripts pane (Window➪Scripting➪Scripts) if you've placed scripts in the Scripts folder inside the folder that contains the InDesign CS application, as shown in Figure 37-1. Scripts don't have to be in the Scripts folder — they can be anywhere on your computer — but to use a script outside this folder means you have to double-click the script from your desktop rather than access it in InDesign.

If you create many scripts, you can control how their names sort in the scripts list. Also, you can now assign keyboard shortcuts to scripts.

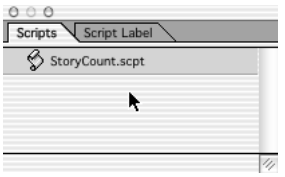

Figure 37-1: The Scripts pane in InDesign.

No matter what scripting language you use, there are several basic principles to observe. These fall into four basic categories:

- **Grammar.** All languages — including programming languages such as Pascal and C++, as well as scripting languages — include grammatical components that are used in standardized sequences. In English, we combine nouns, verbs, adjectives, adverbs, and so on to create sentences. Everybody knows the meaning of "The weather is especially nice today," because it uses common words in a sequence that makes sense. The sentence "Nice is the especially today weather," has the right components, but it's arranged in the wrong sequence, so the meaning is lost.
- **Statements and syntax rules.** In JavaScript, AppleScript, and VBA, verbs, nouns, adjectives, and prepositions are combined to create statements; statements are combined to form scripts. Verbs are also called *commands* and *methods;* nouns are called *objects;* and adjectives are called *properties.* Syntax rules specify how statements and scripts must be constructed so that they can be understood by a computer.
- **The object hierarchy.** All three scripting languages use a structural element called an *object hierarchy.* It's a fancy term for a simple concept. An object hierarchy works like a set of boxes within boxes. A large box contains a smaller box, which contains a smaller box, which contains a smaller box, and so on, until you reach the smallest box, which contains nothing and is the final level in the hierarchy of boxes.

✦ **The InDesign hierarchy.** InDesign contains its own hierarchy, which lends itself nicely to scripting. A project contains layouts, a layout contains pages, pages contain boxes, and boxes contain text and pictures. You can create scripts that perform actions at any of these levels. In other words, with scripts you can create documents, add pages, add items to pages, and modify the contents of boxes — right down to a particular character in a text box. You can think of this hierarchy in InDesign as a chain of command. You can't talk directly to an item that's at the bottom of the chain. Rather, you must first address the top level, then the next, and so on, until you've reached the item at the bottom of the chain. This is analogous to the way you use InDesign: You create new layouts, add pages, place text and graphics on the pages, and, finally, modify the contents of the frames containing those items.

Scripts can now address individual graphics in a layout as objects, create dialog boxes, access object labels (which you can set in InDesign, as well as have InDesign automatically extract from converted QuarkXPress files), set object properties, and identify the file path on which the script resides.

If you're thinking about dabbling with any of the scripting languages supported by InDesign, the following words of both caution and encouragement are in order. First the encouragement: You don't necessarily need programming experience, scripting experience, or a pocket protector to begin creating scripts. A bit of curiosity and a touch of patience will suffice. Now the caution: Scripting is essentially a euphemism for programming (that is, figuring out the right commands and then typing them in for the application to execute). Writing scripts isn't a matter of choosing commands from menus, clicking and dragging, or entering values into fields; nor is it like writing a limerick. If you're starting from scratch, know in advance that you'll have to learn some new skills.

Learning to create scripts is like learning to swim: You can read books, documentation, and articles until your head spins, but eventually you have to get a little wet. The best way to learn about scripting is to write a script. So put on your swimsuit and dive in.

Be forewarned: There's something almost narcotic about creating scripts, and it's not uncommon for novice scriptwriters to get hooked. Don't be surprised if what starts out to be a 15-minute look-see turns into a multi-hour, late-night programming episode. There's a reason why Adobe's PDF guide on scripting is more than 1,800 pages long!

Because scripting languages differ, you can't always duplicate the functionality of a specific script in one language into a script written in a different language.

Exploring JavaScript

JavaScript is a scripting language developed by Sun Microsystems and initially meant to let Web browsers manage resources on far-flung servers by running scripts to control the servers from a desktop. It soon became a popular scripting language because it runs on so many types of computers, including Windows, Macintosh, and Unix. Now, JavaScript is used for both server and desktop programs. Because it is based largely on the object-oriented approach taken by professional computer languages such as C and C++, it can be difficult for nonprogrammers to use.

The JavaScript functionality in InDesign CS was not available before this book went to press. For updated information on how to use JavaScript with InDesign, go to this book's companion Web site, `www.INDDcentral.com`.

There are lots of JavaScript editor programs available. Most of these are developed by individuals and small firms, so the list is always changing. I recommend you use the Google search engine (`www.google.com`) and search for *JavaScript editor* to find the most current programs.

Learning the language

JavaScript is a very complex language based on the concept of object orientation, which abstracts items and attributes as objects that are then grouped, changed, or otherwise manipulated. This means that JavaScript is less "Englishlike" than other scripting languages, since it requires a fair amount of setup of the objects before they can be manipulated. However, JavaScript is a very powerful language, so those who can figure it out can create very powerful programs, not merely scripts.

```
myObject.strokeTint = newValue;
```

This example shows that there is a current object named `strokeTint` that is being set to a new value; the actual value for `newValue` is set earlier in the script.

What you need to write and run scripts

You'll need a program that can display, edit, and test your JavaScript — there is no bundled JavaScript editor in Windows or Mac OS X. In addition to stand-alone utilities, you can usually use an HTML editor such as Macromedia Dreamweaver or Adobe GoLive to edit JavaScripts in, though they typically don't provide any debugging tools to help you track and fix coding (syntax) errors. In this case, you will need to open the error window in your browser as you test the code and see if it identifies the error location to help you find it in your HTML editor. Such editors

typically format the JavaScript code for you, indenting it automatically, graying out comments, and highlighting certain keywords.

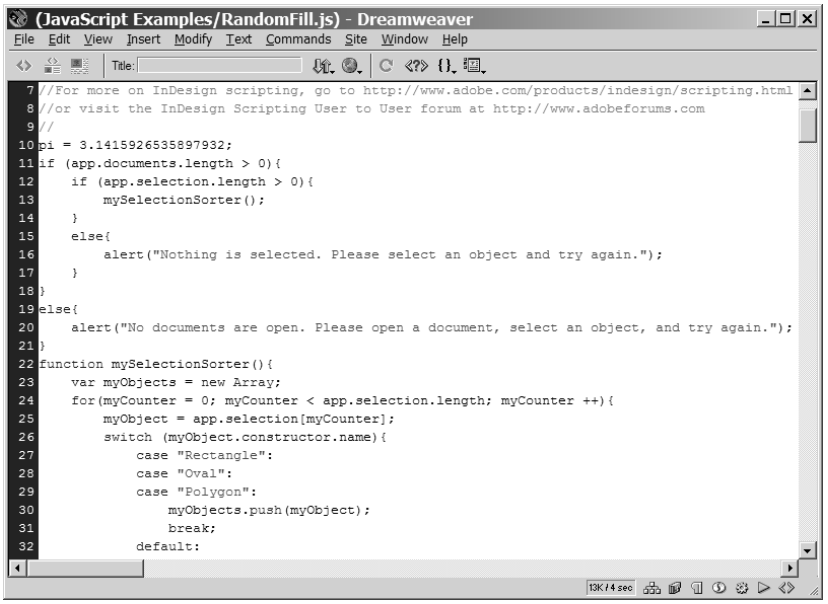

Figure 37-2: A JavaScript program viewed in Macromedia's Dreamweaver.

Getting more information on JavaScript

Before you venture too far into scripting, you should review the JavaScript-related information provided with InDesign:

- **JavaScript documentation and tools.** Sun places the very technical JavaScript documentation on its Web site at `http://devedge.netscape.com/central/javascript/`. A good independent source is the O'Reilly & Associates Web site's scripting section (`http://scripting.oreilly.com`).
- **InDesign scripting documentation.** The InDesign CD contains a 600-plus-page PDF file that explains JavaScript programming for InDesign. This document, although a bit on the technical side, is a valuable resource. It includes an overview of JavaScript scripting and the object model, as well as a list of InDesign-specific scripting terms and scripting examples.

If you want still more information about JavaScript, several books are available, including *JavaScript: The Definitive Guide,* by David Flanagan; *Beginning JavaScript,* by Paul Wilton; and *JavaScript Bible,* 4th edition, by Danny Goodman.

Exploring AppleScript

AppleScript is a scripting language developed by Apple and initially released with System 7.5 that can be used to control Macs, networks, and scriptable applications, including InDesign. The AppleScript language was designed to be as close to normal English as possible so that average Mac users — specifically, those who aren't familiar with programming languages — can understand and use it.

InDesign can now run text-only AppleScripts in addition to compiled (binary) ones.

Learning the language

Many of the actions specified in AppleScripts read like sentences you might use in everyday conversation, such as:

```
set color of myFrame to "Black"
```

or

```
set applied font of myCharacterStyle to "Times"
```

Getting more information on AppleScript

Before you venture too far into scripting, you should review the AppleScript-related information provided with the Mac OS and with InDesign:

- **Mac scripting documentation and tools.** Apple places the AppleScript documentation on its Web site at `www.apple.com/applescript`. In your hard drive's Applications folder, you should have a folder called AppleScript that contains the Script Editor program, along with a folder of example scripts and the AppleScript Script Menu that adds the Script menu to the Finder. Apple also offers a professional AppleScript editor called AppleScript Studio for download at its developer Web site, `http://developers.apple.com/tools.macosxtools.html`.
- **InDesign scripting documentation.** The InDesign CD contains a 600-plus-page PDF file that explains scripting, including AppleScript programming, for InDesign. This document, although a bit on the technical side, is a valuable resource. It includes an overview of Apple events scripting and the object model, as well as a list of InDesign-specific scripting terms and scripting examples.

If you want still more information about AppleScript, several books are available, including *AppleScript in a Nutshell: A Desktop Quick Reference,* by Bruce W. Perry; *Danny Goodman's AppleScript Handbook,* 2nd Edition; and *AppleScript 1-2-3,* by Sal Soghoian.

What you need to write and run scripts

The Script Editor, provided with the Mac OS, lets you write scripts. You'll find the Script Editor inside the AppleScript folder inside your Applications folder (at the root level of your hard drive). An uncompiled script is essentially a text file, so you can actually write scripts with any word processor. The Script Editor, however, was created for writing AppleScripts and includes several handy features for scriptwriters.

Checking for syntax errors

The next step is to determine if the statements are correctly constructed. Click the Check Syntax button. If the Script Editor encounters a syntax error, it alerts you and highlights the cause of the error. If the script's syntax is correct, all statements except the first and last are indented, and a number of words are displayed in bold, as illustrated in Figure 37-3. Your script has been compiled and is ready to test.

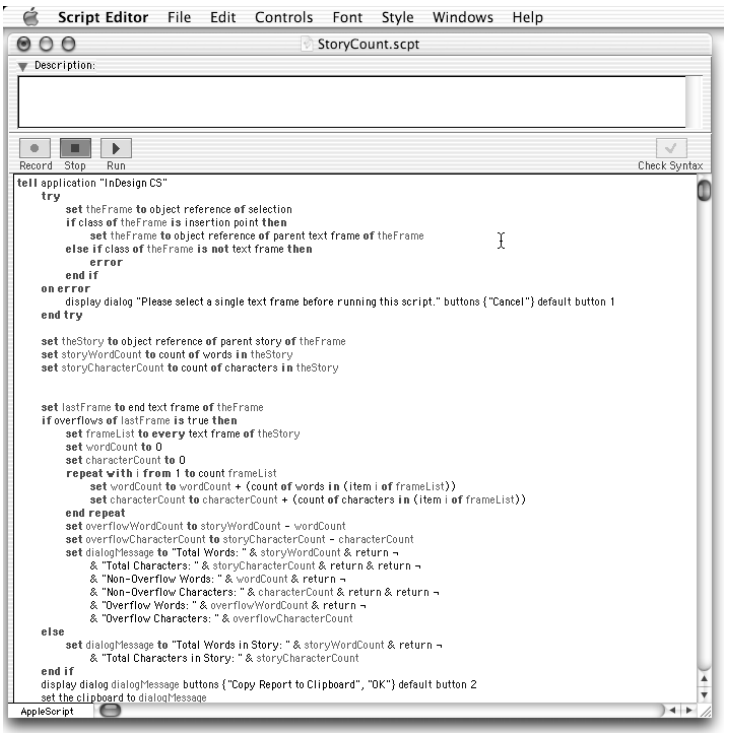

Figure 37-3: The Script Editor window containing sample AppleScript text. When you check the syntax of a script, the Script Editor applies formatting and indents.

Running your script

Click the Run button and then sit back and watch. If you've done everything correctly, you'll see InDesign become the active program, and then the actions you put in your script will take place. Voilà — and congratulations! You can now call yourself a scripter without blushing. That's all there is to creating and running a script.

If you have trouble getting a script to run, double-check the name that InDesign uses for itself. It might use InDesign® CS or simply InDesign® (yes, the name may include the registered trademark symbol). If you run a script from AppleScript (rather than just double-clicking it) and AppleScript can't find InDesign, it will give you a dialog box with which you find the InDesign program. When you've found and selected the InDesign application, AppleScript will find out what InDesign's filename is and use that in your script.

Saving your script

When you're finished writing and testing a script, choose Save from the Script Editor's File menu. Name your script, choose its storage location, and choose Compiled Script from the Format pop-up menu. It's best to save the script in the Scripts folder inside the InDesign folder, so it will show up in the Scripts menu (after you restart InDesign).

If you save the script in Application format and want to edit your script later, you must open it by dragging and dropping it on the Script Editor application. This is because Application-format scripts are designed to immediately run when double-clicked. You would choose the Application format when creating scripts for use by others, since chances are you don't want them to open the script in Script Editor but instead simply want them to use the script by double-clicking it like any other application.

Locating more AppleScript tools

A few software utilities are also available for AppleScripters. The most widely used is Script Debugger, from Late Night Software ($189, `www.latenightsw.com`); it's an interface development tool to quickly create AppleScript-based applications that have the standard Mac look and feel. Apple also offers its AppleScript Studio as a free download to developers who register at the Apple site; this is more capable than the basic Script Editor that comes with Mac OS X.

Exploring VBA

Visual Basic for Applications (VBA), and its subset version VBScript, is Microsoft's technology for writing your own programs, both those that run in other programs (scripts) and those that run by themselves (custom applications). InDesign works with both VBA and VBScript. The Visual Basic language that underlies both VBA and VBScript is not meant for everyday computer users — a knowledge of programming is very useful in taking advantage of this technology. Although based on the Basic language developed in the 1970s to help new users write their own programs, it's evolved a lot since then and is no longer so simple.

Learning the language

Many of the actions specified in VBA have some degree of "Englishness," such as:

```
set myTextFrame =
InDesign.Documents.Item(1).Spreads.Item(1).TextFrames.Add
```

or

```
mySelection.RotationAngle = 30
```

But as you can see, VBA has moved far from English. The first code segment, for example, means to add a text frame to the first spread in the first document. The second means to rotate the selected object by 30 degrees.

Getting more information on VBA

Before you venture too far into scripting, you should review the VBA-related information provided by Microsoft and with InDesign:

- ✦ **Microsoft scripting documentation and tools.** Microsoft has a lot of information on VBA, Visual Basic, and VBScript on its Web site. Unfortunately, it's not well organized and is hard to find and understand. There's no tutorial that simply explains how a new scripter needs to get started. However, you can search on the Microsoft site for VBA, Visual Basic, and VBScript to get links to documents that may prove useful.
- ✦ **InDesign scripting documentation.** The InDesign CD contains a 600-plus-page PDF file that explains VBA programming for InDesign. This document, although a bit on the technical side, is a valuable resource. It includes an overview of VBA scripting and the object model, as well as a list of InDesign-specific scripting terms and scripting examples.

If you want still more information about VBA and its two "sister" technologies, several books are available, including *Programming Microsoft Visual Basic 6.0,* by Francesco Balena; *Visual Basic 6 For Dummies,* by Wallace Wang; and *VBScript in a Nutshell by* Paul Lomax and Ron Petrusha.

What you need to write and run scripts

To use InDesign scripting in Windows, you'll need Microsoft Visual Basic or an application that contains Visual Basic for Applications (VBA); these include Microsoft Office, Microsoft Visio, and AutoCAD. In Microsoft Office, you can run the Microsoft Script Editor by choosing Tools➪Macros➪Microsoft Script Editor, which lets you create scripts, edit them, test your code, and fix errors. Figure 37-4 shows the editor with a sample script.

You can also write scripts in VBScript, a VBA subset, in a text editor such as WordPad. You'll need Microsoft's free Windows Scripting Host (`WSCRIPT.EXE`), which is usually installed with Windows and can be downloaded from Microsoft's Web site.

There's a third choice for your scriptwriting: You can also use the full Microsoft Visual Basic product from Microsoft.

To use InDesign scripting in Windows, your user profile must have Administrator privileges.

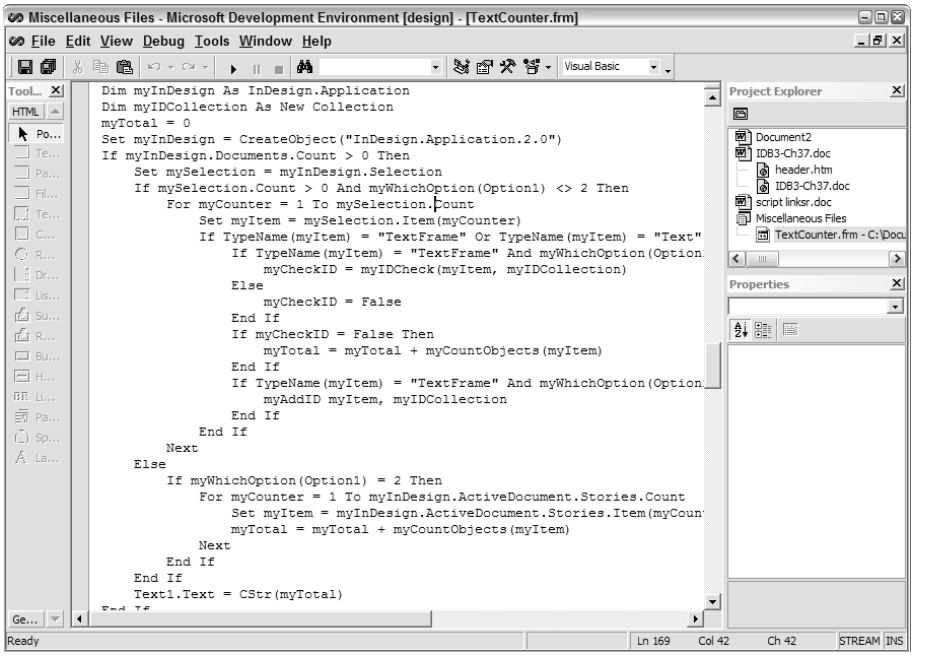

Figure 37-4: The Microsoft Script Editor window containing sample VBA text. When you work on a script, the Microsoft Script Editor applies indents automatically.

Running your script

To run a VBA, Visual Basic, or VBScript program, simply double-click the script. You can also run the script directly from the application that you create a VBA or Visual Basic script in, such as the Microsoft Script Editor. (For VBScripts, you can run them from the Scripting Host application.) If you've done everything correctly, you'll see InDesign become the active program, and then the actions you put in your script will take place. Voilà — and congratulations! You can now call yourself a scripter without blushing. That's all there is to creating and running a script.

Saving your script

When you're finished writing and testing a script, choose Save from the script editor's File menu. Name your script and choose its storage location. It's best to save the script in the Scripts folder inside the InDesign folder (usually `C:\Program Files\Adobe\InDesign CS`), so it will show up in the Scripts menu (after you restart InDesign).

Creating and Running Scripts

At this point, I'm assuming that the appropriate scripting software is installed on your computer. If this is the case, you're ready to begin. For your first trick, you're going to make InDesign roll over — sort of. Actually, you're going to rotate an EPS graphic. First, you'll prepare InDesign for its role. Launch the program, and then create a new document (do not check Automatic Text Box). In the middle of the first page, place an EPS graphic. Make sure that it remains active after you place it.

Writing simple scripts

The following three scripts, taken from Adobe's InDesign script examples, do the same thing in AppleScript and VBA: Rotate an EPS graphic and its frame. (For JavaScript versions, go to the companion Web site at `www.INDDcentral.com`.)

You can get the example scripts shown in this chapter, as well as other samples, from the PDF scripting manual that comes on the InDesign CS CD. Just cut and paste them from the PDF file into the appropriate scripting editor. Adobe's user forums are also a good place to go for scripting help.

Enter the lines that follow this paragraph exactly as they're written for the scripting language you've chosen. Enter a return character at the end of each line. Note also the use of straight quotation marks instead of curly typesetter's quotes (the script editor does this for you). Be very careful when you enter the text: Typos are script killers.

JavaScript

```
for(myCounter = 0; myCounter <
    app.activeDocument.pages.item(0).pageItems.length;
    myCounter ++){
  var myPageItem =
    app.activeDocument.pages.item(0).pageItems.item(myCounter);
  var myPageItemType =
    myPageItem.getElements()[0].constructor.name;
  alert(myPageItemType);
  if (myPageItemType == "EPS"{
    myPageItem.rotation = 30;}
  }
```

This script first searches through all the items on the page and checks if any are EPS; if so, it sets the rotation of the item to 30. The code above would be part of a function defined in JavaScript that sets up the various objects.

AppleScript

```
tell application "InDesign CS"
  set myPageItems to {EPS, oval, rectangle, polygon}
  set mySelection to selection
  if class of item 1 of mySelection is in myPageItems and ¬
        (count mySelection) > 0 then
    if class of item 1 of mySelection is EPS then
      set myFrame to parent of mySelection
    else
      set myFrame to item 1 of mySelection
    end if
    set rotation angle of myFrame to 30
  end if
end tell
```

Note the use of the ¬ character; it indicates that the code continues onto the next line. You would not actually type in this character; the AppleScript Editor would insert it automatically to akert you that it wrapped the code on screen to keep it all visible to you.

Make sure to enter the name of your InDesign program exactly as it appears on the desktop. Because you're free to rename your program, the name may not match the name in the first line of the script.

Finally, perhaps you noticed the chain of command used in the preceding script. First, the script addresses InDesign, then the active document (layout), and finally the active frame. If you understand this concept, you'll be scripting like a pro in no time.

If you're in an adventurous mood, try substituting the statement `set rotation angle of myFrame to 30` in the preceding script with each of the following statements:

```
set text wrap of myFrame to off
set shear angle of myFrame to 30
set vertical scale of myFrame to 200
```

If you want to get really fancy, combine all the `set` statements into a single script, so you can use the script to make all the changes at once.

VBA

```
Dim myInDesign As InDesign.Application
Set myInDesign = CreateObject("InDesign.Application.CS")
Set mySelection = myInDesign.Selection
If TypeName(mySelection.Item(1)) = "EPS" Then
  mySelection.Parent.RotationAngle = 30
Else
  mySelection.RotationAngle = 30
End If
```

Perhaps you noticed the chain of command used in the preceding script. First, the script addresses InDesign, then the active document (layout), and finally the active frame. If you understand this concept, you'll be scripting like a pro in no time.

Labeling items

As you can see from the examples in the previous section, scripts often refer to items by their type and location in the document. But there's another way to refer to objects that makes sure you can select an item precisely: You can label, or name, an item. You do so in the Script Label pane (Window➪Scripting➪Script Label). The process is easy: Select the object, then enter a name in the pane. That's it!

When writing scripts, you refer to the labeled object as follows. In these examples, the label is *TargetFrame,* and don't worry that the samples seem to do different things — they in fact are unrelated examples, not variations of the same command.

JavaScript

```
with(app.documents.item(0).pages.item(0)){
  myTargetFrame = textFrames.item("myTargetFrame");
}
```

AppleScript

```
select (page item 1 of page 1 of myTargetDocument whose label
is "TargetFrame")
```

VBA

```
Set myAsset = myLibrary.Assets.Item("TargetFrame")
```

Writing conditional scripts

Some scripts simply automate a set of tasks in documents whose content is predictable. But more often than not, documents differ, and so you need conditional statements to evaluate if certain things are true before applying a script's actions. Otherwise, you'll get an error message when something turns out not to be true. As a simple example, a script that does a search and replace needs to have a document open and a frame selected. If no frame is selected, the script won't know what to search, and the user will get an error message.

The same issue arises for repeated series of actions, where you want the script to do something for all occurrences. The script will need to know what to do when it can't find any more such occurrences. As an example, look at the following script, which counts all open documents. For it to work, at least one document has to be open, so the script checks first to see if in fact any documents are open, and delivers an error message that the user can understand if none are open. The rotate-EPS-graphic script earlier also used a conditional to make sure there was an EPS graphic in the document. Notice that in all three scripting languages, it is the command `If` that you use to set up such conditionals.

JavaScript

```
if(app.documents.length==0){
  alert("No InDesign documents are open!");
}
```

JavaScript uses == for comparing values (as in the example above) and = for assigning values. Visual Basic and AppleScript use = for both purposes.

AppleScript

```
tell application "InDesign CS"
  set myNumberOfDocuments to (count documents)
  if myNumberOfDocuments = 0 then
        display dialog "No InDesign publications are open!"
  end if
end tell
```

VBA

```
Dim myInDesign as InDesign.Application
  Set myInDesign = CreateObject ("InDesign.Application.CS")
  If myInDesign.Documents.Count
    MsgBox "No InDesign publications are open!"
```

```
  End If
End Sub
```

Another form of conditional is what's called a control loop, in which an action occurs either for a specified number of iterations or until a condition is met. The following scripts show an example of each for each language. Note the use of comments in the scripts — a handy way to document what you're doing for later reference. In JavaScript, a comment begins with `/*` and ends with `*/` (or you can use `//` at the beginning of each line instead). In AppleScript, a comment begins with `--` and continues until you press Enter or Return. In VBA, it begins with `Rem` or `'` followed by a space, and it continues until you press Enter or Return.

JavaScript

```
for(var myCounter = 0; myCounter < 20; myCounter++){
  //do something
}

while (myStop == false){
  /* do something, at some point setting myStop to true
  to leave the loop. */
}
```

AppleScript

```
repeat with counter from 1 to 20
  --do something
end repeat

set myStop to false
repeat while myStop = false
  --do something, at some point setting myStop
  --to true to leave the loop.
end repeat
```

VBA

```
For counter = 1 to 20
  Rem do something
Next counter

Do While myStop = false
  Rem do something, at some point setting myStop
  Rem to true to leave the loop.
loop
```

Summary

If your workflow goes beyond original designs for each client and reaches into repetitive production, scripting is for you. Scripts are ideal for automating repetitive tasks — from importing pictures to creating and formatting entire documents. You can even link InDesign to other scriptable applications.

Because InDesign supports JavaScript on the Mac and Windows, as well as AppleScript on the Mac only and VBA on Windows only, you can choose the script language you're most familiar with and/or that is compatible with your other applications. Even better, you can use more than one scripting language with InDesign (though any individual script can use only one language).

✦ ✦ ✦

Exploring AppleScript

AppleScript is a scripting language developed by Apple and initially released with System 7.5 that can be used to control Macs, networks, and scriptable applications, including InDesign. The AppleScript language was designed to be as close to normal English as possible so that average Mac users — specifically, those who aren't familiar with programming languages — can understand and use it.

New Feature
InDesign can now run text-only AppleScripts in addition to compiled (binary) ones.

Learning the language

Many of the actions specified in AppleScripts read like sentences you might use in everyday conversation, such as:

```
set color of myFrame to "Black"
```

or

```
set applied font of myCharacterStyle to "Times"
```

Getting more information on AppleScript

Before you venture too far into scripting, you should review the AppleScript-related information provided with the Mac OS and with InDesign:

- **Mac scripting documentation and tools.** Apple places the AppleScript documentation on its Web site at `www.apple.com/applescript`. In your hard drive's Applications folder, you should have a folder called AppleScript that contains the Script Editor program, along with a folder of example scripts and the AppleScript Script Menu that adds the Script menu to the Finder. Apple also offers a professional AppleScript editor called AppleScript Studio for download at its developer Web site, `http://developers.apple.com/tools.macosxtools.html`.
- **InDesign scripting documentation.** The InDesign CD contains a 600-plus-page PDF file that explains scripting, including AppleScript programming, for InDesign. This document, although a bit on the technical side, is a valuable resource. It includes an overview of Apple events scripting and the object model, as well as a list of InDesign-specific scripting terms and scripting examples.

If you want still more information about AppleScript, several books are available, including *AppleScript in a Nutshell: A Desktop Quick Reference,* by Bruce W. Perry; *Danny Goodman's AppleScript Handbook,* 2nd Edition; and *AppleScript 1-2-3,* by Sal Soghoian.

What you need to write and run scripts

The Script Editor, provided with the Mac OS, lets you write scripts. You'll find the Script Editor inside the AppleScript folder inside your Applications folder (at the root level of your hard drive). An uncompiled script is essentially a text file, so you can actually write scripts with any word processor. The Script Editor, however, was created for writing AppleScripts and includes several handy features for scriptwriters.

Checking for syntax errors

The next step is to determine if the statements are correctly constructed. Click the Check Syntax button. If the Script Editor encounters a syntax error, it alerts you and highlights the cause of the error. If the script's syntax is correct, all statements except the first and last are indented, and a number of words are displayed in bold, as illustrated in Figure 37-3. Your script has been compiled and is ready to test.

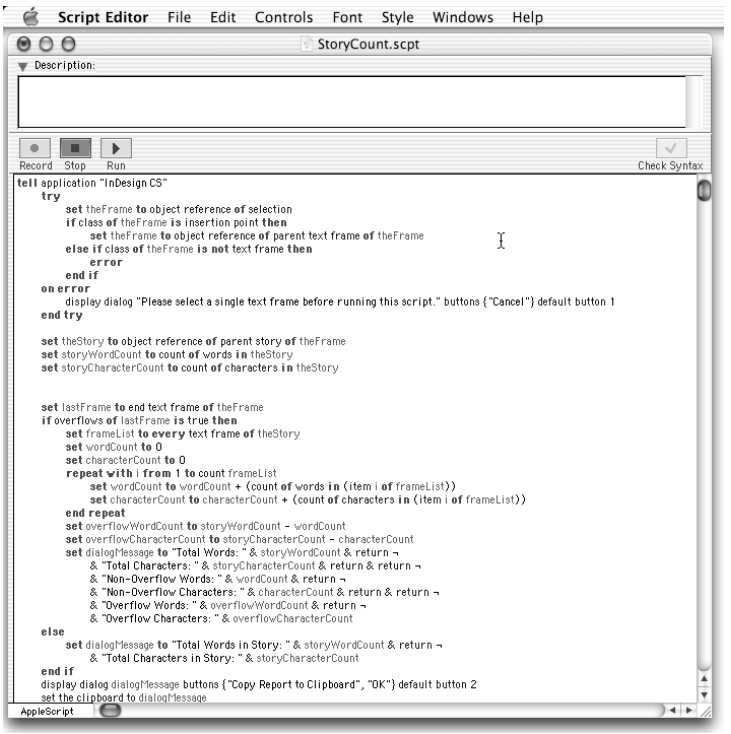

Figure 37-3: The Script Editor window containing sample AppleScript text. When you check the syntax of a script, the Script Editor applies formatting and indents.