Macro location

Macros stand alone from applications in AppleScript

The most obvious difference between Visual Basic for Applications (VBA) and AppleScript is that VBA macros run within an application's documents or templates, while AppleScript macros are stand-alone applications or scripts that run from the operating system's Script menu. In fact, AppleScript macros are actually run by the System Events background application.

Identify applications in tell blocks

In AppleScript, in order to call a command or access an object in Microsoft Word, Excel, PowerPoint, or any other application, you must direct your code to the application in a `tell` block, as shown below:

```
tell application "Microsoft Word"
        save as active document file name myDocName
        file format format text
        -- all your other Word-specific code
end tell
```

If you don't frame the code within a `tell` block, it won't compile, let alone run, as shown in the following example:

```
save as active document file name myDocName
file format format text
```

When you compile, you get the following error message:

```
Syntax Error : A class name can't go after this identifier.
```

In this error, `active document` is selected behind the error message. That's because `save` and `as` are both reserved AppleScript terms, whereas neither `active` nor `active document` are defined terms outside of a Word`tell` block. So, the AppleScript compiler thinks that `active` is a variable. The reserved term `document` cannot follow a variable without an appropriate preposition between them, such as the following example:

```
save as active in document file name
```

This code compiles before hitting another error at `myDocName`, but it doesn't mean anything outside of Word and can't run without generating the following error:

```
AppleScript Error: Can't make file name into type reference
```

Therefore, always remember to include the `tell` block when calling Word commands or classes.

Invoking one application from another

Invoking one Microsoft Office application from another is easy in AppleScript. It makes no difference whether your script is primarily for one application or another. For example, when you call either Word or Excel, you type `tell application "Microsoft Word"` or `tell application "Microsoft Excel"`, and neither application is running the script.

For example, a VBA Excel macro can use a lot of code to get an existing instance of Word or to create a new instance, as in the following example:

```
On Error Resume Next

Set oWord = GetObject(,"Word.Application")
If Err Then
        Set oWord = New Word.Application
        WordWasNotRunning = True
End If
```

In AppleScript, you can delete all that code. Instead, you `end tell` your Excel block, and then start a new `tell application "Microsoft Word"` block. If Word is not open, this instruction opens it.

Your only additional step is to decide whether you now want to bring Word to the front. If you do, type the first line of the Word block as follows:

```
activate
```

If Word is already open, **activate** brings it to the front. If it isn't open, **activate** does not add anything since Word automatically comes to the front anyway, but it's okay to include it.

If you want Word to stay in the background, type the first line of the Word block as follows:

```
launch
```

If Word is already open in the background, `launch` does nothing.