Syntax

Show containment and properties using of

AppleScript is not a "dot" language, like Visual Basic for Applications (VBA), other BASIC languages, or Python. To show containment and attributes, AppleScript's "English-like" syntax combines the preposition `of` with word order that is the reverse of VBA. Compare the following examples:

VBA

```
ActiveDocument.Paragraphs(1).Range.Text
```

AppleScript

```
tell application "Microsoft Word"
        content of text object of paragraph 1
        of active document
end tell
```

In AppleScript, you can specify the **paragraph** element by an index, `1`, but you use the singular term, **paragraph**, and not the plural form, **Paragraphs**, as you do in VBA. And, you don't type parentheses in AppleScript; you simply use `paragraph 1`.

Note There are no collection objects in AppleScript.

You may be used to "dot" languages in which you "drill down" from a top level object to the property you want to access. In AppleScript, you "burrow up" from the property to the top-level object, and there are AppleScript strings with multiple uses of the preposition `of`.
There is an alternative. You can also use the `'s` syntax, in the preferred word order, as shown in the following example:

```
active document's paragraph 1's text object's content
```

Because that sounds awkward in English, most people use this syntax only one level deep, such as `active document's paragraph 1` or `text object's content`.
A third and often very useful option is to `tell` an object to get some property or element, as shown in the following code:

```
tell active document to get its text object
```

You can also use nested `tell` blocks, as shown in the following example:

```
tell application "Microsoft Word"
        tell active document
                tell paragraph 1
                        tell text object
                                get content
                        end tell
                end tell
        end tell
end tell
```

Note This is construction is similar to `With/End/With` blocks in VBA.

Implicit vs. explicit get

Occasionally, AppleScript code with multiple uses of the preposition `of` doesn't work. In these cases, there is no implicit **get**, and you have to explicitly state **get** to access a deeper `of` property. Compare the following examples:

Implicit `get`

```
tell application "Microsoft Word"
        tell active document
                tell text object
                        tell find object
                                set content to "blue"
                                set forward to true
                                execute find
                        end tell
                end tell
        end tell
end tell
--> false
```

Explicit `get`

```
tell application "Microsoft Word"
        tell active document
                tell text object
                        tell (get find object) -- crucial!
                                set content to "blue"
                                set forward to true
                                execute find
```

```
                        end tell
                end tell
        end tell
endtell
--> true
```

It can be difficult to understand what is wrong because this situation doesn't yield an error. Instead, you get an incorrect result. This rarely happens in Microsoft Word, but is more likely in Microsoft Entourage with Microsoft Exchange Server properties where you can receive either an error or a missing value as a result, when you don't use an explicit **get**. Therefore, when in doubt, use a **get**.

Use set, not =, to assign values to variables

An error that scripters new to AppleScript encounter could make is using = as an assignment operator (the implicit **Let** method in VBA), as in:

```
someVariable = "text"
```

In AppleScript, = is used only as an equality comparison operator, as in:

```
if someVariable = "text" then...
```

To assign a value to a variable, you must use `set...to`, as in:

```
set someVariable to "text"
```

Also note that unlike VBA, the syntax is the same regardless of the data type of the item being assigned to the variable. Whether the item is a string, integer, or object, you always use **set**.