Dictionaries

You can use application-specific AppleScript dictionaries to learn AppleScript terminology.

AppleScript dictionary basics

You won't find definitions for AppleScript terms on the Macintosh Script Editor Help menu, as you might in Microsoft Visual Basic Editor. Script Editor Help doesn't cover Microsoft Word or Excel or any of the hundreds of other scriptable applications. Instead, look in the dictionary for the application you want. For example, to find the Word dictionary:

1. In Script Editor, on the **File** menu, click **Open Dictionary**.
2. In the **Open Dictionary** dialog box, click **Microsoft Word**, and then click **Open**.

Now you see the Word AppleScript dictionary, and the hundreds of classes and commands available. They are divided into suites of associated terms.
If you don't find what you're looking for, check more than one suite, usually starting with the Microsoft Word Suite. In particular, always remember that text range, corresponding to the **Range** object in Visual Basic for Applications (VBA), is a class in the Text Suite. (But in Excel, the **range** class is in the Table Suite.) Or, if you think you know all or part of a term, or what it might be called in "AppleScriptese," enter it in the **Search** box.
You'll soon figure out that the color-coded icons containing letters represent "command," "class," "property," and other terms. They are spelled out in the **Kind** column.

Class descriptions

The dictionary entries provide all of the properties and elements of a class, as in the following **footnote** class entry:

```
footnote n [inh. base object]: Every footnote

ELEMENTS
contained by documents, selection objects, ranges.

PROPERTIES
entry index (integer, r/o): Returns the index for the
        position of the object in its container element
        list.

note reference (text range, r/o): Returns a text range
        object that represents a footnote mark.

text object (text range, r/o): Returns a text range object
        that represents the portion of a document that's
        contained in the footnote object.
```

The current version of Script Editor shows you which classes the entry is an element of, that is, which classes it is contained by. You can also see any elements the entry may contain. In this example, there are none.
Note that the description of each property begins with the data type of the property, such as integer (a basic AppleScript type) or **text range** (another Word class). Classes, such as **text range**, are links to their own entries in the dictionary, which is useful.
If the properties are read only, they are marked " r/o." When making a new object, there are default values for every property, so you do not have to mention them. The dictionary usually tells you what the defaults are, especially for boolean properties (*true* or *false* values).
Note Some r/o properties can be set at the time of creation only, when using the `make new [object] with properties {...}` command. In other words, it's always worth a try to see if it's possible.

Command descriptions

The dictionary also includes entries for commands. The entry provides any parameters and whether a result is returned or not, as in the following **undo** command entry:

```
undo v: Undoes the last action or a sequence of actions,
        which are displayed in the undo list. Returns true
        if the actions were successfully undone.

        undo document
                [times integer]: The number of actions to be
                undone.
                -> boolean
```

The first indented line after the definition shows the command followed by the data type of its direct object, which is **document** in the example above.
The lower lines, indented another level, specify any parameters (arguments) that the command may have. Parameters that are inside [square brackets] are optional parameters, such as [**times**] in the example. The parameter is followed by the data type that it takes, or a full enumeration, with enumerated values separated by slashes, if the enumeration is not a class itself.
For example, the entry for the **file path type** parameter of the **get default file path** command is as follows:

```
get default file path v: Returns the default folders for
        items such as documents, templates, and graphics.

        get default file path

                file path type documents path/pictures path/
                user templates path/workgroup templates path/
```

```
user options path/auto recover path/
tools path/tutorial path/startup path/
program path/graphics filters path/
text converters path/proofing tools path/
temp file path/current folder path/
style gallery path/trash path/office path/
type libraries path/border art path : Which
path should be returned.
->Unicode text: The specified default path.
```

In the final line of the definition, if the **—>** is present, it shows the data type and description (if needed) of the result returned.

Dictionary suites

If you opened an application dictionary in Script Editor, you may have noticed that the various terms are listed in suites. This is something of a legacy, inherited from the beginnings of AppleScript.
In Word, for example, everything to do with **Range** (called **text range** in Word AppleScript) is found in the Text Suite, everything to do with tables is in the Table Suite, everything to do with WordArt is in the Drawing Suite, and most of the rest is in the Microsoft Word Suite.
This seems logical when you know that Text, Table, and Drawing Suites are already defined for AppleScript and exist in many applications, though each application can implement particular terms in its own way.
But it also means that very similar commands can be found in two different suites, rather than close together. For example, the **selection object** class is in the Microsoft Word Suite, while **text range** is in the Text Suite. Yet they both have very similar commands, such as **extend** for **selection** properties, and **move end of range** for **text object** properties and other ranges.
It is quite easy to forget this and to look for a command or class in the Microsoft Word Suite to no avail, forgetting to look in the Text Suite or Table Suite. Fortunately, in the more recent versions of Script Editor, you can enter a term such as "move end" in the **Search** box, which you might recall from the **MoveEnd** method in VBA, and the dictionary will return all occurrences of your term from wherever they may occur in the dictionary.

Script Editor entries

Script Editor allows you to view classes and commands by containment and inheritance. Also, within each suite, commands and classes are listed separately, and alphabetically, with the commands listed before the classes. In Script Editor 2, the lists do not have **Commands** and **Classes** headings, but icons are used: a light blue circle that contains a "C" represents commands, and a purple square that contains a "C" represents classes.
Note Unlike many programming languages, especially rigorous object-oriented (OO) languages, AppleScript commands do not "belong" to classes in the way that methods belong to classes in OO languages, although they act only upon such classes as they are restricted to.
A major difference between AppleScript application dictionaries and VBA Help is that properties are not separated into their own subsection, not even for browsing. There are classes (like VBA objects) and commands (like VBA methods), and that's it.

Property and element descriptions

Properties and elements are listed separately within each class. It often happens, especially in Microsoft Office applications, that properties with the same name occur in several different classes. For example, in Excel, both **range** and **row** have a **row height** property. Many, many classes have a **value** property. But you won't see generic definitions in the Excel dictionary for **row height** or **value**, as you might in VBA Help. Instead you'll find **row height** listed under **Properties** in both **range** and **row** classes, and **value** listed under **Properties** in **listbox**, **range**, **cell**, and many other classes.
In AppleScript, unlike VBA, there is a distinction between the two different kinds of attributes that classes can have: properties and elements.
Properties are one-to-one. Every instance (object) of a particular class must have one, and only one, instance of each property available within the class. The advantage to this is that when you create a new object, there is a predefined default value for every property, and you don't have to specify them.
Note Properties can be application objects, enumerated values, or basic AppleScript types, such as Unicode text, an integer, or a boolean.
Elements are many-to-one. Objects can have zero or any number of elements. They are generally made when you create a new object. Elements are themselves application objects whose type is specified in the dictionary.
One thing to note about the current Script Editor dictionary is that it lists, under **Elements**, both the classes that contain the entry (its parent classes) and the classes of elements the entry contains (its child classes). For example, in the **footnote** entry above, many classes are "contained by" **footnote** (its parent classes); however, **footnote** doesn't "contain" any classes (its child classes). Just be sure that you note the difference between the real elements that the defined class "contains" and these "contained by" classes.