Volume 3 • Number 2

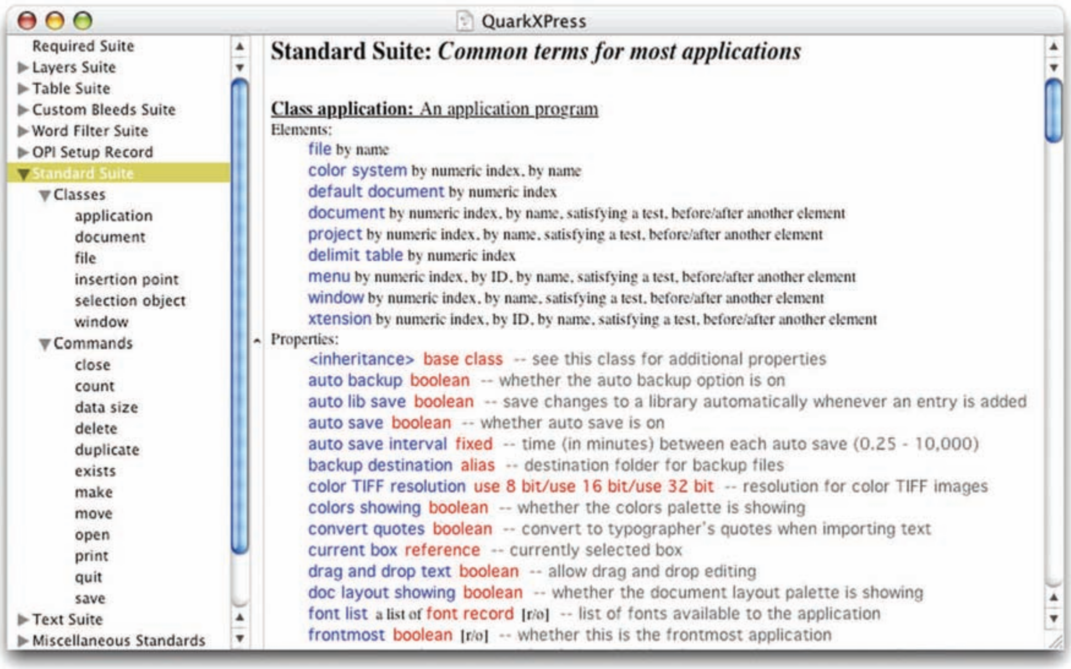
X-RAY MAGAZINE

PUBLISHING, WORKGROUP & ENTERPRISE TECHNOLOGY FOR QUARK USERS

Getting to Know
QuarkXPress' AppleScript Terminology

BY **BENJAMIN S. WALDIE**

For both beginning and experienced scripters, implementing AppleScript automation into a workflow frequently brings with it a learning curve. This is because every scriptable application extends the base AppleScript language with its own set of unique terminology. If you are attempting to script an application that you have not automated before, then you will first need to familiarize yourself with that application's AppleScript terminology. As you begin to script the application, you will also become more comfortable in doing so. Before long, scripting the application will become second nature. Automating QuarkXPress with AppleScript is no different.



Standard Suite: ***Common terms for most applications***

Class application: An application program
Elements:
- file by name
- color system by numeric index, by name
- default document by numeric index
- document by numeric index, by name, satisfying a test, before/after another element
- project by numeric index, by name, satisfying a test, before/after another element
- delimit table by numeric index
- menu by numeric index, by ID, by name, satisfying a test, before/after another element
- window by numeric index, by name, satisfying a test, before/after another element
- xtension by numeric index, by ID, by name, satisfying a test, before/after another element

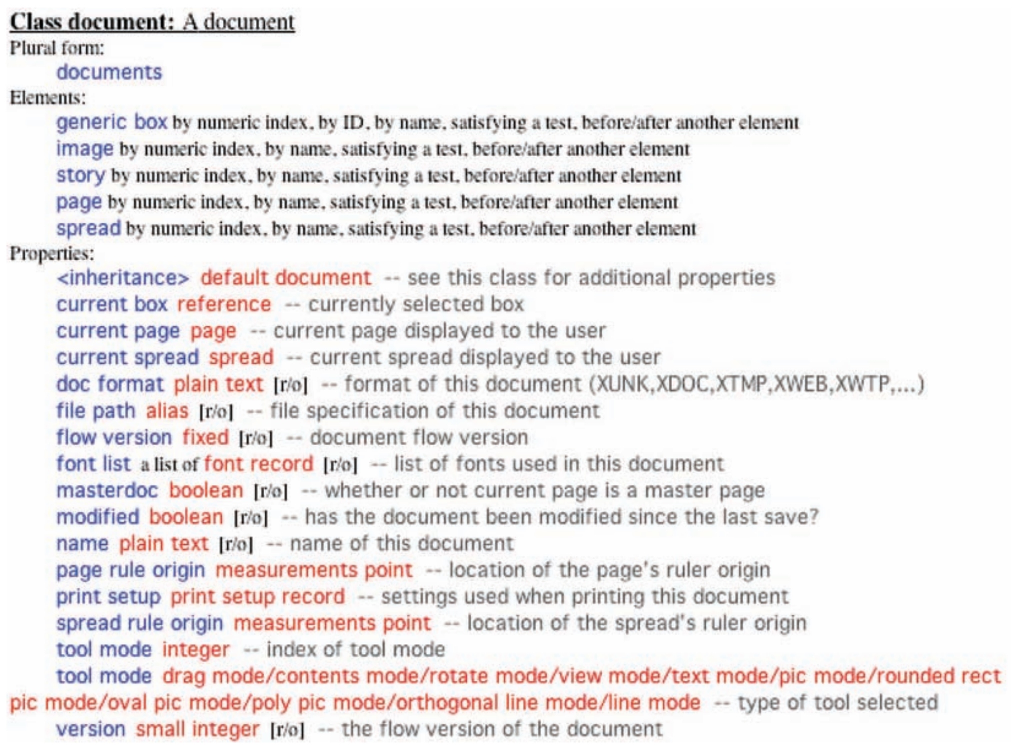
Properties:
- <inheritance> base class -- see this class for additional properties
- auto backup boolean -- whether the auto backup option is on
- auto lib save boolean -- save changes to a library automatically whenever an entry is added
- auto save boolean -- whether auto save is on
- auto save interval fixed -- time (in minutes) between each auto save (0.25 - 10,000)
- backup destination alias -- destination folder for backup files
- color TIFF resolution use 8 bit/use 16 bit/use 32 bit -- resolution for color TIFF images
- colors showing boolean -- whether the colors palette is showing
- convert quotes boolean -- convert to typographer's quotes when importing text
- current box reference -- currently selected box
- drag and drop text boolean -- allow drag and drop editing
- doc layout showing boolean -- whether the document layout palette is showing
- font list a list of font record [r/o] -- list of fonts available to the application
- frontmost boolean [r/o] -- whether this is the frontmost application

fig. 1▶ QuarkXPress' AppleScript Dictionary

Viewing QuarkXPress' AppleScript Terminology

The first step in scripting QuarkXPress is to determine the AppleScript terminology that QuarkXPress understands. To do this, you will need to consult QuarkXPress' AppleScript dictionary. Every scriptable application possesses an AppleScript dictionary, which contains a complete listing of the terminology that is specific to that application. Be aware that the rules for displaying and navigating an application's AppleScript dictionary are the same regardless of whether you are working with QuarkXPress or another scriptable application.

To view QuarkXPress' AppleScript dictionary, first launch the Script Editor application, located within the APPLICATIONS ▸ APPLESCRIPT folder in Mac OS X. Once launched, select FILE ▸ OPEN DICTIONARY and, when prompted, select QuarkXPress. The AppleScript dictionary for QuarkXPress will then be displayed before you in a new dictionary browser window (see figure 1).



```
Class document: A document
Plural form:
    documents
Elements:
    generic box by numeric index, by ID, by name, satisfying a test, before/after another element
    image by numeric index, by name, satisfying a test, before/after another element
    story by numeric index, by name, satisfying a test, before/after another element
    page by numeric index, by name, satisfying a test, before/after another element
    spread by numeric index, by name, satisfying a test, before/after another element
Properties:
    <inheritance> default document -- see this class for additional properties
    current box reference -- currently selected box
    current page page -- current page displayed to the user
    current spread spread -- current spread displayed to the user
    doc format plain text [r/o] -- format of this document (XUNK,XDOC,XTMP,XWEB,XWTP,...)
    file path alias [r/o] -- file specification of this document
    flow version fixed [r/o] -- document flow version
    font list a list of font record [r/o] -- list of fonts used in this document
    masterdoc boolean [r/o] -- whether or not current page is a master page
    modified boolean [r/o] -- has the document been modified since the last save?
    name plain text [r/o] -- name of this document
    page rule origin measurements point -- location of the page's ruler origin
    print setup print setup record -- settings used when printing this document
    spread rule origin measurements point -- location of the spread's ruler origin
    tool mode integer -- index of tool mode
    tool mode drag mode/contents mode/rotate mode/view mode/text mode/pic mode/rounded rect
pic mode/oval pic mode/poly pic mode/orthogonal line mode/line mode -- type of tool selected
    version small integer [r/o] -- the flow version of the document
```

◂ fig. 2 The Document Class

Navigating QuarkXPress' AppleScript Dictionary

Since QuarkXPress' AppleScript support is extensive, its dictionary may appear daunting at first glance. However, there is a method to the madness. If you look closely, you will see that QuarkXPress' AppleScript terminology is actually organized into groups. These groups are called suites, and each one represents a set of terms that are related in some manner. For example, the Layers Suite contains terms that relate to manipulating layers in a document, the Tables Suite contains terms that pertain to tables, and so forth.

Within each suite, the scripting terminology is organized even further into two types of terms, *classes* and *commands*. Some suites will contain both types of terms, and some may contain only one. For example, the *Layers Suite* contains both classes and commands, while the *Tables Suite* contains only classes.

In AppleScript, a class represents an object, or element, with which you may interact through scripting. In a scriptable application such as QuarkXPress, a document, a page, a layer, and even the application itself, are all considered to be classes. A command represents an action that can be taken through scripting, and a command targets a class. For example, the term QUIT is a command, and may be directed toward the QuarkXPress application in order to instruct it to quit.

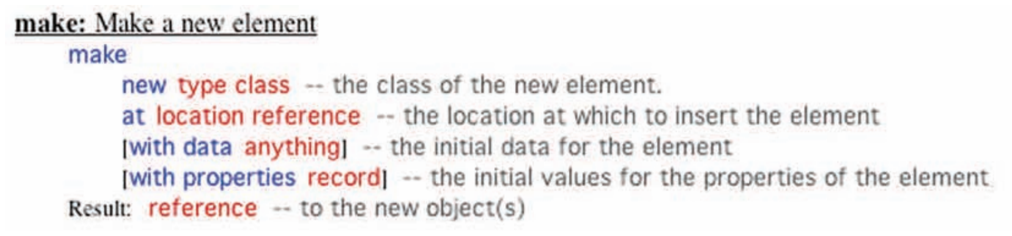
More About Classes

When working with classes, there are a few important concepts to grasp, some of which may initially seem complex.

First, most classes possess properties. A property is an attribute of a class that may be accessed via AppleScript. For example, a document is a class, and in reality a document possesses a number of attributes, including a name, a file path, a page height, a page width, and so on. In AppleScript, these are all considered to be properties of a document, and may be accessed via scripting. In QuarkXPress' dictionary, selecting a class will display a listing of the properties that are accessible for that class (see figure 2). When viewing properties, you may notice that certain properties are followed by the text `[r/o]`. This indicates that the property is a read-only property, and may not be changed through scripting. The property may, however, be *retrieved* through scripting.

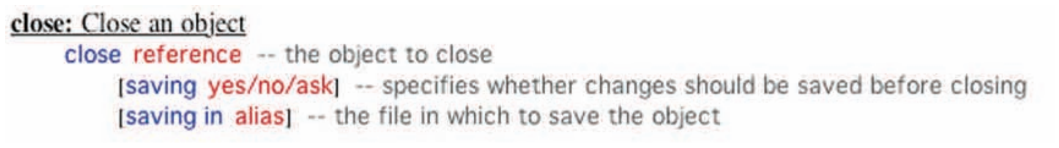
Another concept regarding properties is that a class has the ability to possess the properties of one or more other classes. This concept, known as inheritance, typically occurs when multiple classes possess many of the same properties. Rather than displaying those properties multiple times within the dictionary, they are listed under a more generic type of class, and then referenced by other, more specific

Understanding the layout and organization of QuarkXPress' AppleScript dictionary is really the first step in the process.



```
make: Make a new element
    make
        new type class -- the class of the new element.
        at location reference -- the location at which to insert the element
        [with data anything] -- the initial data for the element
        [with properties record] -- the initial values for the properties of the element
    Result: reference -- to the new object(s)
```

▶ fig. 3 The Make Command



```
close: Close an object
    close reference -- the object to close
        [saving yes/no/ask] -- specifies whether changes should be saved before closing
        [saving in alias] -- the file in which to save the object
```

▶ fig. 4 The Close Command

classes. In QuarkXPress' dictionary, the document class indicates that it inherits the properties of a default document class. Therefore, if you look up the default document class, you can assume that the properties listed there pertain to a document as well.

Classes may also contain other classes as elements. For example, a document contains pages, which may contain text boxes, which may contain paragraphs, and so forth. This structure of nested classes is known as an application's object hierarchy. When referring to a class within an application, you must do so very specifically within its hierarchy, so that the application knows exactly what you are targeting. For example, the AppleScript code to refer to a specific character may appear as follows:

```
tell application "QuarkXPress"
   character 1 of paragraph 3 of text
box 1 of page 3 of document 1
end tell
```

In QuarkXPress' dictionary, when selecting a class, the elements of that class, if any, are displayed (see figure 2).

More About Commands

Commands may seem slightly more straightforward than classes. As mentioned before, a command targets a class, and causes some type of action to occur. Within QuarkXPress' dictionary, clicking on a command displays a definition for that command. A command's definition consists of a brief description for the command, as well as a listing of any parameters that may be used with the command (see figure 3). A parameter is a value that will affect the behavior of the command.

When viewing commands, there are two types of parameters that may be present. A direct parameter is a parameter that immediately follows a command, and a labeled parameter is a parameter that follows a label (see figure 4).

When initiating a command, some parameters may be required, and some may be optional. Brackets surrounding a parameter indicate that the parameter is optional.

Another thing to note when working with commands is that some commands may return a value, once they have completed their task. The type of result, if any, that is returned by a command, will be displayed in the command's definition within the dictionary (see figure 3).

Pulling it Together

Let's take a look at some examples of how classes and commands are utilized when scripting. The following example code will create a new, empty QuarkXPress document, using QuarkXPress' default document setup.

```
tell application "QuarkXPress"
   make new document at beginning
end tell
```

In the code above, the `make` command is targeted toward the QuarkXPress application, instructing it to make a new document. In this instance, this is done through the use of a `tell` statement. Rather than including a reference to the QuarkXPress application in the line of code containing the command, the command is placed within a `tell` statement. This will ensure that any code within the statement uses the QuarkXPress application as its default target.

The following example code will behave in a similar manner as the previous code. However, in addition to creating the document, the `with parameters` optional parameter is also specified, indicating the page width and page height of the document to be created.

```
tell application "QuarkXPress"
   make new document at beginning with
properties {page width:9, page height:12}
end tell
```

Now, let's take a look at another command. The following example code will close the front document.

```
tell application "QuarkXPress"
   close document 1
end tell
```

Like the `make` command, the `close` command may also be enhanced with the use of optional labeled parameters. For example:

```
tell application "QuarkXPress"
   close document 1 saving no
end tell
```

Extending QuarkXPress' AppleScript Support

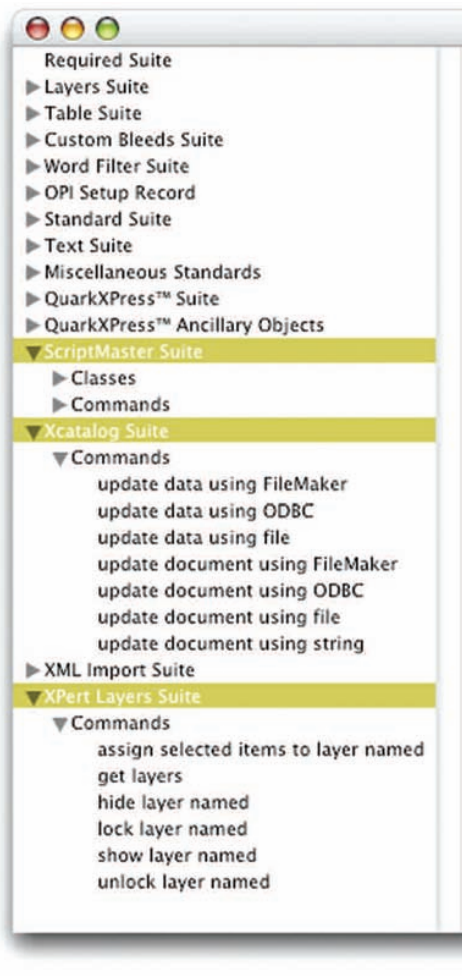
Depending on the XTensions you have installed, you may notice additional terminology suites within QuarkXPress' AppleScript dictionary (see figure 5). This is because software developers may choose to implement AppleScript support within their XTensions, thus expanding QuarkXPress' AppleScript terminology even further.

The following are some examples of scriptable XTensions, which can provide you with some very powerful options for automation. All of the XTensions mentioned below are available from ThePowerXChange at http://www.thepowerxchange.com

ScriptMaster XT adds a number of classes and commands to QuarkXPress' scripting implementation, which can ease tasks that would otherwise require more complex scripting to be done. Tasks that can be easily automated with ScriptMaster XT include find and replace, step and repeat, inserting pages, exporting text, and more. ScriptMaster XT also provides the ability to record AppleScript code as you perform manual operations. This functionality can prove to be an invaluable resource for anyone getting started with QuarkXPress scripting, as it can help you to easily determine the proper scripting syntax for a specific task, when navigating the dictionary may not be yielding the desired result.

Xcatalog enables a user to create links between elements in a QuarkXPress document and fields in a database or delimited file. When triggered, Xcatalog may be used to automatically synchronize these links, either importing data from a data source directly into a QuarkXPress document or updating modified data in the document back to the data source. Since Xcatalog is scriptable, it also will allow a user to initiate its automated process through scripting. This functionality can enable users to create very robust catalog-automation systems, eliminating countless hours of manual work.

XPert Layers enables users to quickly and easily work with and manipulate items by layer within a QuarkXPress document. It also offers a number of AppleScript commands for automating common layer-related activities.



```
Required Suite
▶ Layers Suite
▶ Table Suite
▶ Custom Bleeds Suite
▶ Word Filter Suite
▶ OPI Setup Record
▶ Standard Suite
▶ Text Suite
▶ Miscellaneous Standards
▶ QuarkXPress™ Suite
▶ QuarkXPress™ Ancillary Objects
▼ ScriptMaster Suite
   ▶ Classes
   ▶ Commands
▼ Xcatalog Suite
   ▼ Commands
      update data using FileMaker
      update data using ODBC
      update data using file
      update document using FileMaker
      update document using ODBC
      update document using file
      update document using string
▶ XML Import Suite
▼ XPert Layers Suite
   ▼ Commands
      assign selected items to layer named
      get layers
      hide layer named
      lock layer named
      show layer named
      unlock layer named
```

◀ fig. 5 Scriptable Plug-In Terminology Suites

In Conclusion

Throughout this article, we have reviewed a number of topics that should help to put you one step closer to creating your own AppleScript-based QuarkXPress workflows. Understanding the layout and organization of QuarkXPress' AppleScript dictionary is really the first step in the process, and is essential in order to begin scripting.

In articles to come, we will begin actually pulling together the things that we have discussed in this article and in the article in Volume 3, Number 1, in order to begin creating some functional scripts that will help to make your everyday workflows more efficient. Until then, I encourage you to begin exploring QuarkXPress' AppleScript terminology on your own, in order to gain a better understanding of the classes with which you may interact through scripting, and the commands that may be used to inflict action on those classes.

See you in the trenches.

)