



# AppleScripting QuarkXPress 7



BY BEN WALDIE

**QuarkXPress 7 is now available, and is full of great new features and options that will make your documents better than ever and your desktop-publishing workflow more efficient, but what has changed with regard to QuarkXPress' AppleScript support, if anything?**

I'm sure this is a question that is on the minds of many, as QuarkXPress-based AppleScript workflows are prevalent throughout the Macintosh community. Will your existing AppleScripts continue to work, or will adjustments be necessary? Has existing AppleScript terminology changed? Have new scripting features been added? These questions, and more, usually come to mind whenever any scriptable application is updated, and in this column we will discuss them as they pertain to QuarkXPress 7.

## Faster Scripting

The first topic that I want to address is the speed by which AppleScripts can interact with QuarkXPress 7. Some users have noted that AppleScripts interacting with QuarkXPress 6.X in Mac OS X actually seemed to run a bit slower than AppleScripts interacting with older versions of QuarkXPress. To determine if QuarkXPress 7 showed any improvement in this regard, I decided to perform a test.

I wrote an AppleScript that, when run, would create twenty-five text boxes, each containing a single line of text. I wrote the script to repeat this process twenty-five times, creating a new page every time the current page contained twenty-five text boxes. The final result was a twenty-five-page QuarkXPress project with each page containing twenty-five text boxes, each with one line of text.

First, I ran the script with QuarkXPress 6.5 on my iMac G5, and it took 109 seconds to complete its tasks. Next, I ran the script with QuarkXPress 7. This time, the script completed its tasks in 78 seconds, a roughly 28% increase in processing speed over QuarkXPress 6.5.

Now, this simple test was hardly comparable to a complex AppleScript-based workflow, such as catalog automation. However, it was sufficient enough to show that AppleScripts that interact with QuarkXPress 7 do appear to run faster than when interacting with its predecessor.

## Printing

When Apple released Mac OS X 10.4, there were some significant changes made to the printing architecture in the operating system. Unfortunately, these changes resulted in some AppleScript problems with QuarkXPress 6.5. Specifically, when attempting to print a project from AppleScript, whether sending it to a printer or to a PostScript file, QuarkXPress would unexpectedly crash. Well, I am pleased to say that, in my testing, this problem seems to be resolved with QuarkXPress 7. I was successfully able to print QuarkXPress 7 documents to both a printer and a PostScript file with AppleScript.



### What's Different?

Next, I would like to address some of the differences between QuarkXPress 6.5 and QuarkXPress 7's AppleScript terminology.

When I'm not writing articles for *X-Ray Magazine*, I develop custom AppleScript solutions for clients and many of these solutions often interact with QuarkXPress. So, I decided to conduct another test. I gathered a large amount of my existing AppleScript code, which I had written for various QuarkXPress 6.5 projects, and gave it a try with QuarkXPress 7. By and large, I was surprised to find that almost all of my code ran successfully, without flaw.

Being accustomed to the AppleScript terminology changes that often occur whenever a new version of an application is released, I had anticipated that a large number of changes would be necessary in order for my existing AppleScript code to function with QuarkXPress 7. As I said before, this was not the case. QuarkXPress did an excellent job of providing backward compatibility with almost every one of its scripting features, so much so that many existing QuarkXPress 6.5 scripts may not need to be updated at all. They may, in fact, just work.

There were, however, a few minor differences that I did encounter. Below is a list of some of these differences, of which I made note. Please note that this list is not meant to be a comprehensive list.

### SAVING EPS FILES

When saving QuarkXPress pages as eps files with AppleScript, the EPS format parameter for the save command has been changed slightly. With QuarkXPress 6.5, this parameter would accept one of the following values for the eps format of the saved file:

```
Mac black and white/Mac color/Mac
dc /Mac dc 2/pc black and white/pc
color/pc dc /pc dc 2
```

In QuarkXPress 7, this parameter will now accept the following values:

```
tandard ep /Multiple ile
dc /ingle ile dc
```

Now, here is an example of AppleScript code that will save a QuarkXPress page as an EPS file to the desktop, using this updated AppleScript terminology in QuarkXPress 7:

```
et theOutput ilePath to path to
de ktop folder a tring
" e t.ep

tell application "QuarkXPre

tell layout pace 1 of project 1

ave page 1 in
theOutput ilePath EP format
tandard EP OP include
image EP data binary EP

end tell

end tell
```

### PRINT SETTINGS

In QuarkXPress 6.5, the print setup record class contained a property called print colors. This property could be used to specify the method for printing colors in QuarkXPress' print settings, and its options included: black and white/gray cale/compo ite CM /compo ite

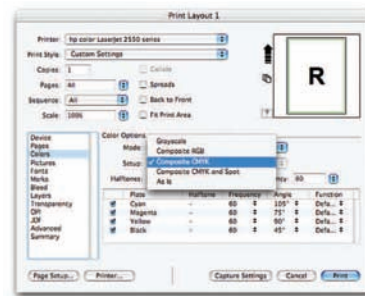


Figure 1  
Color Options  
Setup

In QuarkXPress 7, this property is now referenced as output setup. Furthermore, the property no longer accepts the same values that it did in QuarkXPress 6.5. It now requires a string, rather than a value class, that indicates the method to be used for printing colors. Acceptable strings include the values that are displayed in the color options setup pop-up menu in QuarkXPress' print dialog. See figure 1. Note that the values available in this pop-up menu depend on the value selected in the mode pop-up. In figure 1, the selected value in the mode pop up was composite.

Here is an example of the proper AppleScript syntax for modifying the output setup property with AppleScript in QuarkXPress 7. This code will set the print color options setup to composite CMYK, first ensuring that separation is turned off, which is equivalent to setting the color options mode pop-up to composite.

```
tell application "QuarkXPre
    tell layout page 1 of project 1
        tell print setup
            set separation to false
            set output setup to
                "Composite CMYK
        end tell
    end tell
end tell
```

### What's New?

Now, let's take a look at some of the new AppleScript functionality in QuarkXPress 7. Again, not a whole lot has changed. With the exception of a few minor differences, most of the AppleScript functionality in QuarkXPress 7 is virtually identical to that of QuarkXPress 6.5. There were, as with many software updates, a handful of new additions.

#### OPACITY PROPERTIES

Probably one of the most welcomed features in QuarkXPress 7 is the ability to utilize transparency with regard to text boxes, picture boxes, and the like. Well, I'm pleased to say that this feature has been made accessible to scripting. With AppleScript, you can now set the opacity level of text and picture boxes, characters, and more.

The following example code demonstrates how to set the opacity level of an image to 50%.

```
tell application "QuarkXPre
    tell layout page 1 of project 1
        set opacity of image 1 of
            picture box 1 to 50
    end tell
end tell
```

Here is another example. This code will set the opacity level of a single word in a text box to 50%.

```
tell application "QuarkXPre
    tell layout page 1 of project 1
        set opacity of word 1 of
            text box 1 to 50
    end tell
end tell
```



#### OPEN TYPE

Character styles for OpenType fonts are now accessible via scripting. When utilizing an OpenType font you can write AppleScript code that will enable or disable character styles such as fractions, swashes, ordinals, and more. The following example code demonstrates how to retrieve all of the OpenType character styles of a single word in a specified text box.

```
tell application "QuarkXPre
    tell layout page 1 of project 1
        tell text box 1
            open type style of word 1
        end tell
    end tell
end tell

    standard ligature :false
    discretionary ligature :false
    ordinal :true
    titling :false
    alternate :false
    all small caps :true
    fraction :false
    wahe :true
    small caps :false
    contextual alternate :false
    tabular figure :false
    proportional figure :false
    lining figure :false
    old style figure :false
    position:none
```



Figure 2 New Table Gridline Classes

## TABLE GRIDLINES

QuarkXPress now provides access to horizontal and vertical gridlines in tables, allowing AppleScript code to be written to retrieve or modify the width, shade, opacity, color, and more of table gridlines. Figure 2 shows these new classes, as they appear in the Table Suite in QuarkXPress' AppleScript dictionary.

Here is an example of AppleScript code that will set the width of all horizontal and vertical gridlines in a specified table to 2-point.

```
tell application "QuarkXPre
    tell layout page 1 of project 1
        tell table box 1
            set width of every
            horizontal gridline to 2
            set width of every
            vertical gridline to 2
        end tell
    end tell
end tell
```

## In Conclusion

All in all, as I have mentioned several times already, QuarkXPress 7's AppleScript terminology hasn't changed too much from QuarkXPress 6.5. While some scripters may be seeking out new scriptable functionality, others are probably more interested in backward compatibility with their existing AppleScripts. While QuarkXPress 7 seems to offer few new scriptable features, its backward compatibility with existing AppleScripts is extremely impressive.

If you are currently running AppleScripts with QuarkXPress 6.5, then you should find that most of your scripts will require little or no adjusting in order to continue functioning with QuarkXPress 7. For those that will require adjustments, those adjustments will probably be fairly minor.

That said, as with the introduction of any software update into a scriptable workflow, I highly recommend running your scripts in a test environment prior to implementing them into your live workflow. If you do encounter a problem, and there are issues to be overcome, this will allow you to catch them and address them before they affect your bottom line.

See you in the trenches. ☒