

Tips for Updating Your QuarkXPress AppleScripts

BY BEN WALDIE

Many long-time QuarkXPress users will probably agree, if it works, then leave it alone. This ideology is evident in many QuarkXPress-based workflows, especially those that have been working reliably for years and years. If you attend a QuarkXPress users' group meeting, or a QuarkXPress-related seminar, you will no doubt find people using all different versions of QuarkXPress, from version 3.32 and even earlier, to the latest and greatest.

When it comes to upgrading QuarkXPress, there are usually many factors involved in making the decision of whether or not to make the upgrade. These factors include everything from licensing costs, to service-provider compatibility issues, to stability questions, to system requirements. In this column, I'd like to focus on one specific factor that is often carefully considered when upgrading QuarkXPress — making sure that your existing AppleScripts will continue to function after the upgrade is complete.

The Myth About QuarkXPress Script Updates

Unfortunately, there is a major misconception that plagues many QuarkXPress users whenever the thought of upgrading arises — I'm about to completely dispel it. Many people have the notion that updating QuarkXPress-based AppleScripts, or any AppleScripts for that matter, is a time-consuming and arduous process. The truth of the matter is that the process of updating is not always as time consuming and involved as one might think. In fact, updating an existing AppleScript to function in the latest version of the Mac OS, with the latest version of QuarkXPress, is typically relatively quick and easy. I'll explain why this is the case throughout this article.

Your scripts have probably been customized to work in your specific workflow; therefore the actual update

process will vary to some degree for each person and script. If you are expecting me to outline actual AppleScript changes that have occurred in the Mac OS and QuarkXPress over the years, then I apologize in advance. This particular column is not intended to provide a comprehensive list of AppleScript code changes that must be made in order to update your scripts. Rather, I intend to shed light on the update process as a whole, hopefully alleviate some of the fear of updating, and provide some best-practice guidelines to follow in order to actually make the update a reality.

Updating a Mac OS 9* Script For Mac OS X

**Whenever I refer to Mac OS 9 throughout this column, it should be taken to mean Mac OS 9 or earlier.*

The first topic that I would like to address is updating your Mac OS 9 AppleScripts for use in Mac OS X. The reason this is important is because I know for a fact that there are a large number of users out there still running Mac OS 9 simply because the version of QuarkXPress they are using is stable and their existing AppleScripts are functioning well. This may well be the case, however, the choice to upgrade operating systems, or QuarkXPress versions for that matter, should not be put off forever. There are many good reasons to upgrade, including the ability to use the newer and faster hardware and software that will not run in Mac OS 9.

The first thing to consider when updating a Mac OS 9 script for use in Mac OS X is any AppleScript terminology changes that may have occurred in the operating system and in applications such as the finder. The main thing to understand is that when any AppleScript-able software is updated, there is a chance that its AppleScript terminology may have changed. Typically these changes are relatively minor and usually occur due to things such as new features being added or existing features being removed from the software.

In the case of updating a Mac OS 9 script for use in Mac OS X, you should primarily watch out for general changes in the operating system that may affect your script's execution. For example, when Mac OS X was initially released there was no way to apply a label color to a file or folder in the finder. Due to this limitation, any AppleScripts that were written to apply a label color to a file or folder in Mac OS 9 would produce an error when attempting to perform the same function in Mac OS X. Since Mac OS X's initial release, label colors have been added back into the operating system so this is no longer an issue. However, it did give scripting headaches to some and it serves to prove my point.

In general, you just need to be aware that things behave differently in Mac OS X than they did in Mac OS 9. The operating system behaves differently; it contains many new features, and some old features have been removed or have changed completely. Keep this in mind and be aware of what your scripts are actually doing. If your script attempts to interact with a feature that does not exist anymore, your script will require an adjustment.

Don't let this put you off though. AppleScript's core terminology has not changed from Mac OS 9 to Mac OS X. Some minor behavioral changes have occurred in the finder so if your script relies heavily on performing tasks in the finder, you may run into instances where your finder code requires some adjusting. If this is the case, you will probably be looking at relatively minor adjustments. The best way to determine if adjustments are necessary is to set up a test machine and run your code in Mac OS X. If you're lucky the code may work just fine, and this is sometimes the case. If your code does fail, try to isolate the code that failed by running smaller chunks of code separately. More than likely, any problems you do encounter will be process specific in nature, such as code that deals with the finder or a control panel that no longer exists, for example. In the case of the finder, you should consult the finder's AppleScript dictionary in Mac OS X's Script Editor, locate the terminology that is failing, and determine what may have changed. In the case of a non-existent

control panel, you will need to do some additional research. Posting a question to Apple's AppleScript users' mailing list at <http://lists.apple.com> is usually a good place to start.

Another consideration when updating a script for use in Mac OS X is whether the script interacts with any Scripting additions. Scripting additions are files that can be installed into the operating system in order to extend the capabilities of the AppleScript language. Scripting additions add new commands to the AppleScript language, thus allowing scripts to perform tasks quickly and easily which might otherwise require lengthy coding or inefficient processing speeds. If a script utilizes a scripting addition, then that scripting addition must be installed on the machine that will actually be running the script.

When updating a Mac OS 9 script that utilizes a scripting addition, the first thing to do is determine whether that scripting addition is available for Mac OS X. This can be done by contacting the company that created the scripting addition, or by consulting MacScripter.net's scripting additions repository at <http://osaxen.com/>. Many commonly used Mac OS 9 scripting additions have been updated for Mac OS X. However, if you encounter a scripting addition that has not been updated then you will need to find another solution.

One solution could be to find another scripting addition that contains similar commands. Another solution could be to eliminate the need for the scripting addition altogether by writing AppleScript code to perform the desired task. This may be difficult to do but it is often possible, especially now that triggering UNIX commands in Mac OS X can extend AppleScript.

Updating Scripts for a New Version of QuarkXPress

The process of updating QuarkXPress code within your scripts is virtually the same as that for updating Mac OS 9 scripts for use in Mac OS X. First, you will want to check for changes in QuarkXPress' AppleScript terminology. What many QuarkXPress users often don't realize is that these changes are typically very minor. Quark, Inc., has done an excellent job over the years of retaining forward compatibility and consistency with regard to their AppleScript terminology. Overall, the primary changes that have occurred within QuarkXPress' AppleScript terminology have been due to bug fixes or new scriptable functionality being added.

For example, if you take a look at QuarkXPress 4's AppleScript dictionary (see figure 1), you will notice that it is very similar to QuarkXPress 5's AppleScript dictionary (see figure 2). Most of the terminology from QuarkXPress 4 has simply been carried over into

QuarkXPress 5, and will still continue to function as it did previously. There are some minor differences here and there, and some new terminology has been added for newer scriptable features, such as tables and OPI set up. QuarkXPress 6's AppleScript dictionary (see figure 3) is also very familiar. Again, much of the terminology from previous versions of QuarkXPress has not been changed though new terminology has been added, such as a suite of XML import terminology.

Another factor to consider when updating a QuarkXPress script is whether or not that script interacts with any QuarkXPress XTensions. Some XTensions can actually extend QuarkXPress' AppleScript terminology. Therefore, if your script interacts with an XTensions module, you should consult the XTensions developer in order to determine whether an updated version is available. If it is not, then you will need to find another solution, such as writing AppleScript code to perform the desired function. Depending on the functionality of the XTensions software, this could quickly become a complex and difficult task. However, it is usually possible in one way or another.

As you may already know, unfortunately, not every aspect of QuarkXPress is directly accessible by AppleScript. In Mac OS 9, users were frequently able to get around this limitation by utilizing a third-party tool, such as PreFab Player (<http://www.prefab.com>). This tool provided a way for AppleScript to simulate user interaction in QuarkXPress by interacting directly with interface elements such as clicking buttons, selecting menus, and entering text into fields. By using this technique, scripts could be written to perform virtually any task imaginable in QuarkXPress. Unfortunately, PreFab Player is not compatible with Mac OS X. However, Apple has added this ability into the operating system as a feature. Beginning with Mac OS X 10.3, AppleScripts can utilize a feature called UI scripting in order to interact with many of the interfaces in non-scriptable applications. What does this mean if you have existing PreFab Player code in a script? Well, since Apple's UI scripting's terminology is different than that of PreFab Player, it means that you will need to rewrite certain portions of your script. However, more than likely those portions will be relatively small. In Mac OS 9, printing QuarkXPress documents to PostScript format was a common task that could be performed directly by AppleScript. However, doing so lacked the ability to embed fonts in the resulting PostScript file. To get around this limitation, scripts often utilized PreFab Player to navigate both QuarkXPress and the operating system's print dialog boxes and the system option to embed fonts.

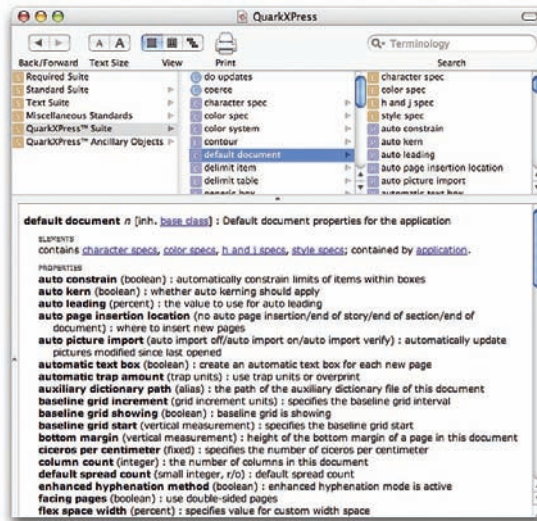


fig 1 QuarkXPress 4's AppleScript dictionary.

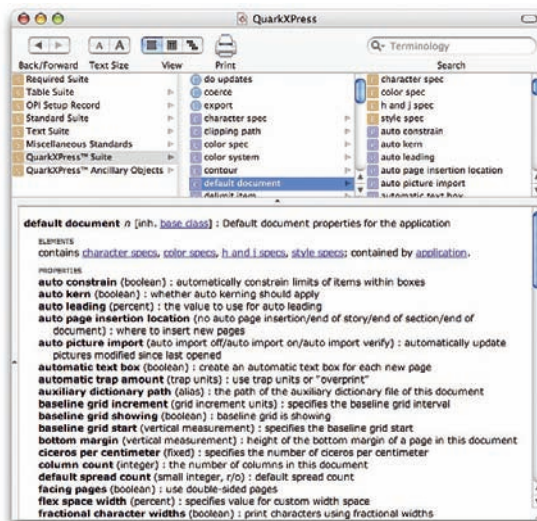


fig 2 QuarkXPress 5's AppleScript dictionary.

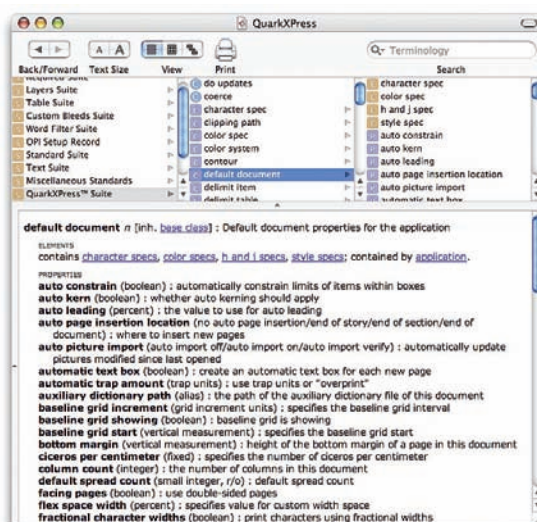


fig 3 QuarkXPress 6's AppleScript dictionary.

The bad news is that if your script performs this task, it will need some updating. The good news is that in Mac OS X, the entire process can be simplified. First, virtually all of QuarkXPress' print settings are directly accessible through scripting. Second, you can print a document to a PostScript directly from AppleScript. Third, Mac OS X will automatically embed the fonts in the resulting PDF.

Another method of interacting with non-scriptable features in QuarkXPress is to make use of QuicKeys (<http://www.quickeys.com>), a system-wide macro utility, which can be triggered from AppleScript to interact with most application interface elements.

Testing, Testing, Testing

I have touched on a number of things to consider when updating a script, however, the best way to update a script is to create a test environment. If at all possible, set up an isolated machine for testing purposes; install the desired versions of the Mac OS and QuarkXPress on that machine, along with your script. Install updated versions of any scripting additions and XTensions activated by the script. Next, begin testing the script relentlessly. If you know of specific problem areas in the script, then attempt to adjust the code in those areas; if not, then try to run the script without making any changes. If and when errors occur, begin testing smaller portions of the code within your script. Make adjustments, and integrate those adjustments back into your main script. If you run into difficulties during the update process, don't panic. Be assured that there is almost always a workaround of some kind.

If you don't feel comfortable making script updates yourself, then you may want to search your staff for that hidden AppleScript developer. Since AppleScript is often self-taught, many designers and non-programmers actually have a knack for picking up at least a basic understanding of the language. This is coming from a guy who started as a designer, taught himself AppleScript, and is now doing AppleScript consulting for a living. If you cannot find an in-house AppleScript developer then you may want to secure an outside developer to help you with the update. If you decide to stick with it and perform the update yourself, there are many resources online where you can post questions and get answers quickly from experienced scripters. The Quark forums (<http://www.quark.com/service/forums/>) contain a forum dedicated solely to QuarkXPress and AppleScript. Another great resource, which I mentioned earlier, is the AppleScript users' mailing list (<http://lists.apple.com/>), where you can find the answer to virtually any AppleScript question imaginable — no matter how large or small.

In any case, after some testing and adjusting, before long your script will be up and running again. I can't stress enough that you should be sure to conduct very thorough parallel testing prior to implementing the updated script into your live workflow. Nothing hurts more than implementing a script that's not adequately tested into a live workflow and having all hell break loose.

Other Considerations

Before making a major script update, you may also want to consider evaluating the functionality in your existing script in order to determine whether it truly meets your needs. In some cases, a major update may be an opportune time to consider re-writing your script in a more efficient manner or adding new features or functionality into your script. While it may take longer to rewrite a script, it may save you time and make your workflow more efficient in the long run.

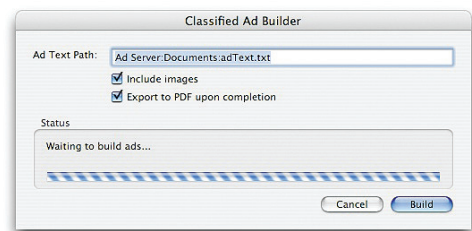


fig 4 An example of an AppleScript Studio script interface.

Using an application such as AppleScript Studio (<http://www.apple.com/macosx/features/applescript/studio.html>) to add an interface to your script can also help to take a script to a new level of user friendliness and usefulness (see figure 4).

In Conclusion

If you are one of the many users that have been putting off updating your existing AppleScripts, then I hope this column will entice you to reconsider that decision. Updating an AppleScript is a much less arduous process than many people think, and the benefits of updating often greatly outweigh the benefits of leaving things the way that they are.

I encourage you to seriously consider updating your scripts, especially if that sole task is holding you back from taking advantage of a new operating system and a new version of QuarkXPress — both with lots of great new features. Once your workflow is running more smoothly on the most current hardware and software, you will be glad that you did.

See you in the trenches. ☒