Make AppleScript Do Your Dirty Work

by Ed Haynes and Kris Fong

IMAGE COURTESY OF APPLE COMPUTER

Rather than repeatedly opening your daily apps, folders, Web sites, and favorite songs, you can easily create scripts that launch everything at once.

EASY

WHAT YOU NEED

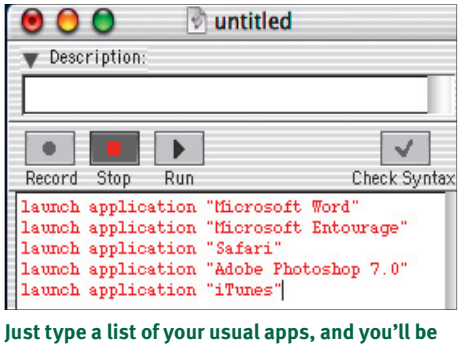
- AppleScript, part of OS X ($129, www.apple.com)

In a world of repeat offenders, repeat performances, and Eddie the Echo, it's nice to know you don't have to get slogged down doing repetitive tasks on your Mac. With AppleScript, anyone can create scripts that take care of some of your most mundane deeds—don't worry, you don't need to know one iota about programming to make an application your slave.

We get you started by showing you some basic AppleScript commands you can use to automate everyday tasks such as launching all the apps you use daily or frequently used folders, fixing disk permissions, opening your usual Web sites, and playing your favorite tunes. We then show you how to turn your scripts into a Cocoa application that will carry out your tasks with a mere click.

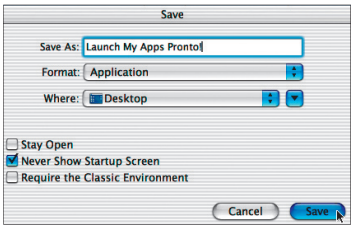
Launch Apps Automatically

1 Compose the Code If you launch a group of apps on a day-to-day basis, create a script that opens all of them at once. Launch Script Editor (Applications › AppleScript); in the editor window, the bottom box is where you write the script (filling in the top Description box is optional). Type *launch application,* type a space, and then type the name of one of your apps, enclosed in double quotes (""). Press Return and repeat the process for all other apps you wish to launch. Be sure to type application names exactly as they appear in the Finder.



Just type a list of your usual apps, and you'll be able to launch them all with a single double-click.

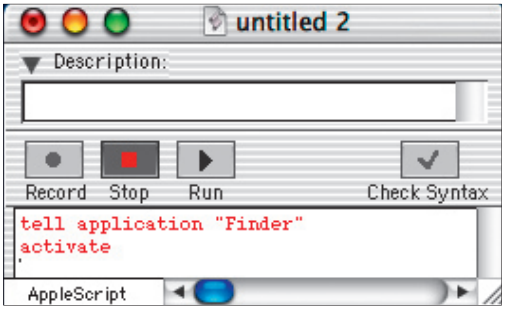
2 Check It and Run When you're finished, click the Check Syntax button to have AppleScript format your script—if it doesn't, correct your mistakes and try again. Once it's formatted, click Run to watch your script in action. Then save it by selecting Save from the File menu. In the resulting dialog, name your script (for example, App Launcher Script) in the Save As field, select Compiled Script from the Format pop-up menu, and click Save to preserve your handiwork. Then save the script as an application for use. Select Save As from the File menu, give it a name (for example, App Launcher), select Application from the Format pop-up menu, check the Never Show Startup Screen box, and click Save. Now, anytime you want to launch your programs, just double-click your new app.



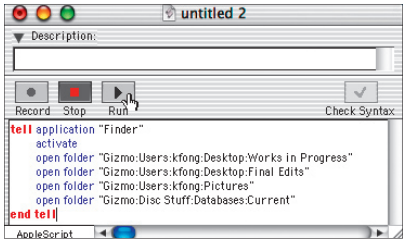
After test-running our script, we saved it as a compiled script (to preserve our scripting), and then as an application (for actual use).

Open Folder Sets

1 Activate the Finder We open the same set of folders every day for our work. Luckily, we can have AppleScript do this for us instead. Open a new Script window (Command-N), and type *tell application "Finder"* in the scripting box; this selects the Finder. Press Return, type *activate*, and press Return again.



Issuing commands like a drill sergeant, we can make the Finder do our grunt work.

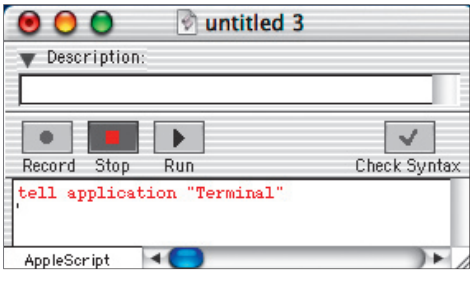


After we formatted our script by clicking Check Syntax, we clicked Run to test it.

2 Target Folders To add folders, type *open folder*, a space, and then the path to the folder, enclosed by double quotes; separate each folder in the path with a colon (:). For example, to open your Applications folder, type *open folder "Macintosh HD:Applications"*, where *Macintosh HD* is the name of your hard drive. Press Return and repeat these steps for all other folders you wish to include in your set. Then type *end tell* on a new line, click Check Syntax to format your script, and click Run to test it. Resize and position your folder windows the way you want them to appear on your desktop. Finally, close them and save your script as a compiled script (for future editing) and as an application (for use).

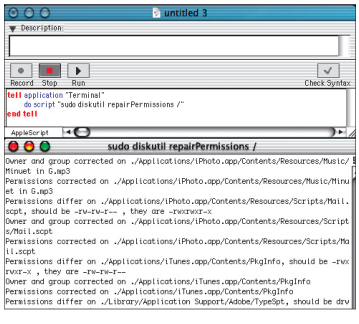
Fix Disk Permissions

1 Call Upon the Terminal You can script many apps to make your life easier, and the Terminal is no exception. Rather than launch the Terminal and type out commands every time you want to do some system maintenance, you can create an AppleScript that launches the Terminal and types the commands to, say, run the repair Permissions disk utility for you. To do this, open a new script window and type *tell application "Terminal"* on the first line. Then press Return.



To script an application, select it first by typing the *tell application* command.

2 Execute the Terminal Command To make the Terminal execute a command, type *do script* and type a space. Then type *"sudo diskutil repairPermissions /"*. Press Return, type *end tell* (close all *tell* commands with an *end tell* command), and then click Check Syntax to format the script and Run to test it; the Terminal launches and asks for your administrator password. Type this into the Terminal and press Return, and the utility goes to work, repairing any disk permission oddities on your hard drive (this may take some time—you'll know it's finished when you see your user name and prompt again). Save your script as a compiled script and an application.



We had no idea how screwed up our disk permissions were—luckily all we have to do is double-click our script the next time we need to do some maintenance.

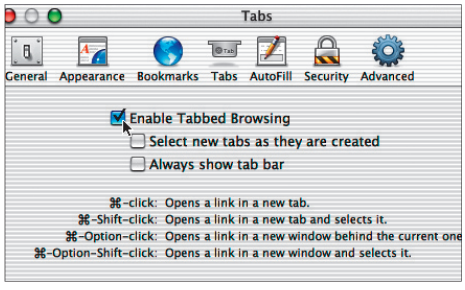
DO MORE WITH LESS EFFORT

We showed you a few ways that you can use AppleScript to automate tasks, but you don't necessarily have to create a separate script for everything. You can take everything you just learned and combine all the syntax into one superscript. For example, instead of double-clicking your app launcher, folder-sets opener, Web-site launcher, and music-player scripts every morning, you can combine all four scripts into one "launch everything" script that'll launch your apps, folders, URLs, and playlist in one double-click fell swoop.

If you want to be lazy (or maybe just efficient), drop your new superscript in the Dock, and you can launch everything with just one click instead of double-clicking. Better yet, you can execute your script without clicking at all if you set it as a login item. Open System Preferences, click Login Items, and click Add. In the resulting dialog, navigate to and select your superscript, and then click Add. Now anytime you boot your Mac, your script will have everything waiting for you—without you lifting a finger.

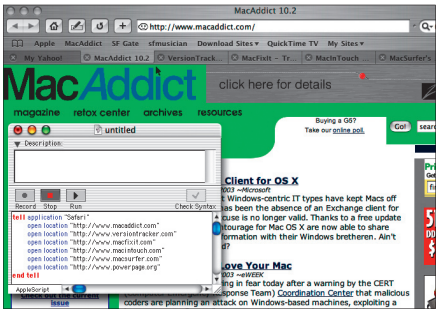
Launch Your Favorite Sites

1 Activate Your Browser Besides getting that first cup of coffee each morning, we like to check out what's going on at all of our favorite Mac Web sites. Selecting every bookmark in Safari is a repetitive bore, but a chore we can ignore if we script once more (OK, we'll stop with the rhyming). First make sure tabbed browsing is enabled in Safari. Launch Safari, select Preferences from the Safari menu, click Tabs, check the Enable Tabbed Browsing box, and then quit the app. Then open a new script in Script Editor, type *tell application "Safari"*, and press Return.



To make Safari open all of your URLs in one window, enable tabbed browsing.

2 Add Favorites For any URL you want to open, type *open location "http://www.macaddict.com"*, where *http://www.macaddict.com* is your desired URL. Press Return and repeat the process for all other URLs you'd like to launch simultaneously. When finished, type *end tell*, click Check Syntax to format the script, and then click Run; Safari will launch and open each site under a separate tab in one browser window. Then save your script as a compiled script and an application.

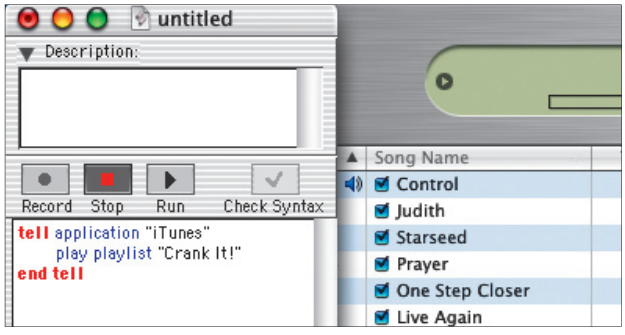


Our new script launches our favorite sites under separate tabs—just click a tab to view the site.

Play Your Favorite Songs

1 Create a Playlist If you're lazy like us, you can create a script that automatically launches iTunes to play your favorite songs. First, create a playlist of your favorite songs in iTunes if you don't have one already; in iTunes, select New Playlist from the File menu and type a name for your playlist. Click the Library (in the left column under Source), drag and drop your favorite songs onto the new playlist icon, and then quit the app.

To make a script that automatically plays your favorite songs, first create an iTunes playlist if you haven't got one already.



After clicking Run, our script launched iTunes and started playing the first song in our chosen playlist.

2 Script the Script Open a new script in Script Editor, type *tell application "iTunes"*, and press Return. On the next line, type *play playlist "my playlist"*, where *my playlist* is the name of your own playlist. Press Return and then type *end tell* to close the script. Click Check Syntax and then click Run to test it. If all's well, save your work as a compiled script and as an application. Anytime you want to hear your favorite songs, just double-click the application.

COMMANDEER THE COMMANDS

While the command for launching an application is self-explanatory, many other AppleScript commands aren't as obvious. Fortunately, you can cheat by calling up the AppleScript Dictionary, which lists every command available for every scriptable app you have installed on your machine and explains how to use them.

To open the Dictionary, select Open Dictionary from the Script Editor's File menu; the resulting dialog lists all applications that can be controlled via AppleScript. Select any app and click Open. The selected app launches along with its Dictionary window, which lists every command you can use to control that particular app. Click any command in the Dictionary window to get a description of what it does and how to use it in Script Editor.

Turn Scripts into Applications

by Ed Haynes

TRICKY

WHAT YOU NEED

- AppleScript, part of OS X ($129, www.apple.com)
- Apple Developer Tools, December 2002 or later release (free, either on Apple's Developer Tools CD or as a download from http://developer.apple.com/tools/macosxtools.html)
- Scripts—your own or someone else's

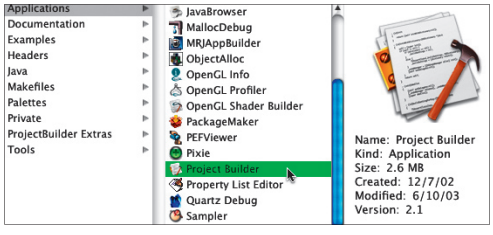
Running a multitude of individual application scripts isn't always practical (we're assuming you didn't already create one big superscript). Luckily, with AppleScript Studio, a part of Apple's Developer Tools package, you can build your own Cocoa application to launch the scripts you've already made or any third-party scripts you want to group together. Note: If your Developer Tools version predates the December 2002 release, you need to download the latest package from Apple's site.

Repetitive launching, opening, and selecting are things of the past—with AppleScript Studio, you can create a Cocoa app that does your deeds at the click of a button.

IMAGE COURTESY OF APPLE COMPUTER

1 Install and Set Yourself Up

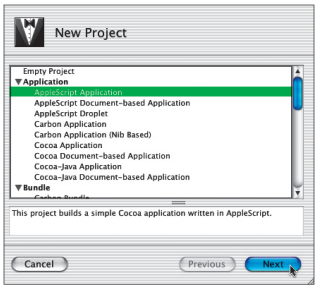
If you haven't done so already, install Developer Tools. Then go to Developer › Applications, locate Project Builder, and launch the application. If you've never used this app before, you'll be greeted by an Assistant window that'll set you up as a new user. Use the Assistant to choose where to store product builds, what type of window environment you'd like to work in, and how you'd like to have projects saved and closed (we chose all the defaults). Click Finish when you're done.



Once you install Developer Tools, launch Project Builder to begin creating your own application.

2 Create a New Project

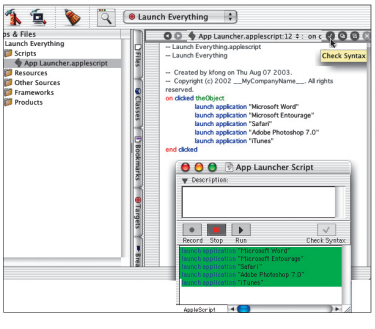
In Project Builder, select New Project from the File menu; in the resulting window, select AppleScript Application from the Application category, and then click Next. In the resulting screen, name your project in the Project Name field, and then click Finish; Project Builder launches an interface for your project.



Select New Project from the File menu to build the framework for your pending application.

3 Paste In a Script

In the Groups & Files column, click the Scripts disclosure triangle to reveal a blank script file that bears the same name as your project. Rename this as one of the scripts you created (for example, App Launcher.applescript); Control-click the script name, select Rename from the contextual menu, and type the new name. Then double-click the corresponding compiled script to open it in Script Editor, select all text, copy it, click in the Project Builder interface's right-side scripting window below the text info, and paste in the script. Modify it by typing *on clicked theObject* above the script's first line, and then typing *end clicked* as the last line. Click the checkmark to format the script.



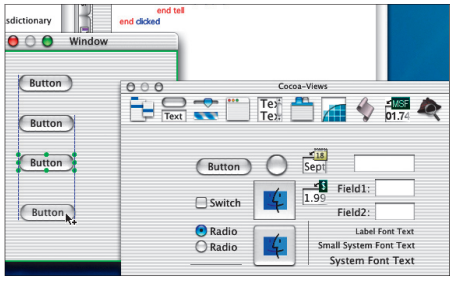
After pasting in and modifying the existing script, format it by clicking the checkmark.

4 Add Other Scripts

To add more scripts, click Scripts in the Groups & Files column, and then select New File from the File menu. In the resulting Assistant window, select AppleScript File, click Next, type a script name in the File Name field (for example, Open Folders.applescript), and click Finish to add the file to the Scripts folder. Double-click your corresponding compiled script, copy and paste the text into the Project Builder window, modify the script by adding the same two lines mentioned in step 3, and click the checkmark to format it. Repeat for all other scripts.

5 Build a Push-Button Interface

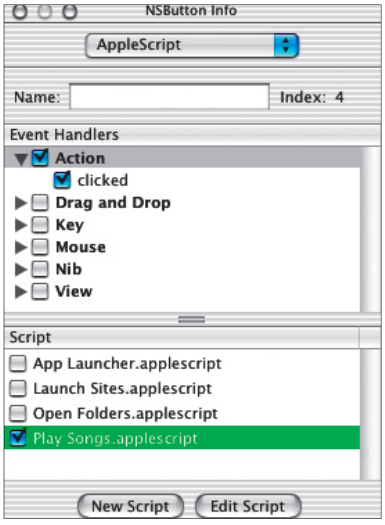
Click the disclosure triangle next to Resources, and double-click MainMenu.nib to open it in Interface Builder; an empty window named Window and a series of tool palette windows appear. In the window with Cocoa in the title, click the cocoa-views button (the translucent button icon with the word Text beneath it) to display its graphics. Drag and drop the Button graphic from the Cocoa-Views window onto the Window window—drag as many buttons as you have scripts (in our case, we dragged four buttons).

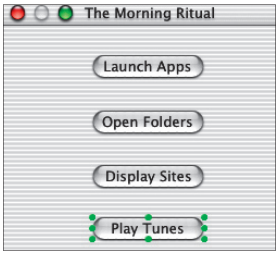


The buttons you see here will launch each of our scripts, and the Window window will ultimately become our app's interface.

6 Set the Interactivity and Build It

Select your first button. From the Tools menu, select Show Info. Select Attributes from the top pop-up menu, and then type a name that describes one of your script's actions in the Name field (for example, Launch Apps). Then select AppleScript from the top pop-up menu. From the Event Handlers list, check the Action box (this also checks the Clicked box), and then check the corresponding AppleScript from the Script list. Repeat for all other buttons. Select the Window window, select Attributes from the Info window, give your app a window title, save your work, go back to Project Builder, and save your project. From the Build menu, select Build to save your new app in your project folder's Build folder.





To program each button's interactivity (right), just link its graphic to the corresponding script using the Info window (left).

Ed and Kris wish TV script writers were as diligent as AppleScripters.