ntroduction

In the July 1997 issue of Apple Wizards, I described in A Spider Speaks how to create a simple AppleScript that would empty a browser's cache folder and then launch the browser. It was a rather pithy example of AppleScript's capabilities, but it generated quite a bit of feedback, all saying "Cool" or "I want more." Well, here's more AppleScript for you! — Erik J. Barzeski

This article assumes that you have the AppleScript package, which includes the Script Editor application, installed on your Mac. If you don't have it, please download it from the Apple website (at http://applescript.apple.com/default.html) and install it before proceeding.

mproving the Daily Routine

If you're like me, there's a certain routine (perhaps even a ritual) to starting up your Mac the first time each day. There are a number of things I alway do first such as view my daily calendar, log onto the net, and check my email.

Additionally, for the last six months or so, I have also been reading 6 or 7 different news sites on the web (listed in the Resources section at the end). Loading these pages is a fairly long and tedious process using my 33.6 modem. Since these sites are fairly graphically intense, load times for each can take as long as a minute or two. After doing the math, I figure that I am spending 10-15 minutes per day waiting on the web.

I could streamline the process a bit by starting all of the pages downloading, each in its own window, and then go do something else while they finish up. However, there is no easy way to do this using the browser's

interface. I still have to manually create a new window and then select the URL from my bookmarks for each site. There has got to be a better way.

Enter AppleScript. AppleScript was designed for exactly this kind of repetitive task automation. With a little knowledge of what sort of commands Navigator will accept, I can write a script that will access all my news site URLs, downloading each one in its own window (we will go over the details of this script later). The script can run in the background, leaving me free to read my email while it works. I have the added benefit of being able to run this script at any time, so I can now use it to check for late-breaking news in the afternoon. The best part about using AppleScript is that it is free — the scripting environment (which contains everything you need to write your own scripts including the Script Editor application) is built into the Mac OS.

ebunking the Misconceptions

"That's great!" you say, "But I could never write a script like that. I'm not a programmer." Well, you don't have to be a programmer to use AppleScript. Granted, to do some of the more powerful things with AppleScript, it helps to understand the basics of programming, but let me emphasize again that to do basic automation using AppleScript you do not need to be a programmer.

In fact, you may have already been using AppleScript and you didn't know it. In your Apple Menu, there should be a folder called "Automated Tasks," which includes many useful items such as "Add Alias to Apple Menu." All of these items are AppleScript applications (sometimes called "applets") that are provided with Mac OS. If you haven't already done so, go ahead and try them out. See for yourself how easy it is to use AppleScript.

"OK, I can run existing AppleScripts. but what if I want to create my own custom script?" AppleScript has a great thing called "recordability" which works very much like the record button on a tape deck. All you have to do is press the record button in the Script Editor, and then you simply perform the actions you want to script. The editor will capture these actions for you automatically. Thus, it's not necessary to start from scratch in order to write your own script.

Just to prove how easy it is, go ahead and open up the Script Editor. You should see something like the following:

ress the Record button and then click on the Desktop, then complete the following tasks in the Finder: create a folder, change its name to "my stuff," move it onto your hard drive, open it, and then change the view and sort properties. Once you are finished, switch back to Script Editor. Your new script should look something like the below. Note: The below text might look a bit different because I've changed the colors of my AppleScript Formatting. This option is available in the Edit menu of the Script Editor.

```
	tell application "Finder"
		activate
		make new folder at desktop
		select folder "untitled folder"
		set name of selection to "my stuff"
		move selection to startup disk
		open selection
		select folder "my stuff" of startup disk
		open selection
		set view of container window of folder "my stuff" of startup disk to name
		set view of container window of folder "my stuff" of startup disk to modification
date
		close container window of folder "my stuff" of startup disk
	end tell
```

Now press the Run button on your script. The script should execute up until the "move selection to startup disk" line. It should then give you an error:

This is a good thing, and
llustrates one major
difference between Apple-
Script and traditional
programming — with
AppleScript, you never
have to worry about
breaking your computer.
There are a lot of
safeguards built into
AppleScript that make it practically impossible for an error to crash your computer, delete important data, etc.

After dismissing the Error dialog, go back to the Finder and delete the two folders that were recently created. Now when you run the script, you can watch it automatically perform all of the tasks that you did manually.

OK, so there are a few catches to the recordability feature. First, not all applications support it, so if you want to script an application that is not recordable, you'll have to resort to other ways of generating a script. Also, recordability only let's you generate basic linear scripts without any of the more powerful capabilities of AppleScript such as decision-making.

odifying an Existing Script

Let's say we already have a simple script that opens a URL in Netscape Navigator (this same script should work in Internet Explorer as well; simply replace the text "Netscape Navigator" with "Internet Explorer"):

```
set theDialog to display dialog "Enter the URL to open:" default answer ""
set theURL to text returned of theDialog
tell application "Netscape Navigator"
      OpenURL theURL
end tell
```

This script is included as "OpenURL" with the DOCMaker version of Apple Wizards. Look in the "Special Files" folder.

Copy this script into a new window in the Script Editor and try it out. Note: Script Editor will prompt you to locate your browser on your hard disk first. Type in any URL, click OK, and your browser will load that page. Not bad, but there is one problem with this script: if we run it a second time, the second page loads in the same window causing the first page to be lost. This is the same behavior as the Open Location command, but it's not what we

want. Let's modify the script so that each new URL opens in its own window.

To see what AppleScript commands are possible for any application, we need to look at the application's Dictionary. To do this, select Open Dictionary... from the File menu of the Script Editor. Select Netscape Navigator (or Internet Explorer) in the open dialog. Tip: As a shortcut for opening an application's Dictionary, you can also drag an application onto the Script Editor icon in the Finder. You should see a window like this:

he window is split into two panes, with keywords on the left, and a description of the syntax on the right. Select the OpenURL keyword, and you should see a description of this command. The text in bold is the name of the command, and the plain text which comes after it tells you what kind of data this command needs in order to work. In this case, the OpenURL command needs a string (specifically a URL) in order to work. There are optional parameters you can add to the command. These are shown in brackets. There is one of particular interest to us: toWindow. It allows you to specify in which window (using the window ID as a reference) to open the URL.

So, all we need now is to figure out how to get a reference to a new window and we'll be ready to modify the script. One of the basic commands that AppleScript allows is the "make new" command. This allows you to create a new object for nearly any object that is listed in the Dictionary. If you look back at the first script you recorded, you can see this command in the line

"make new folder at desktop." This command will return a reference to the new object, so we can save that reference for later use in the script: "set folderReference to make new folder at desktop."

We can use this same concept in our modified script. We can get a referece to a new window by telling the browser to "set newWindowID to make new window." Then we can open the URL in the new window using the reference to it. Here is a modified script:

```
set theDialog to display dialog "Enter the URL to open:" default answer ""
set theURL to text returned of theDialog
tell application "Netscape Navigator"
set newWindowID to make new window
      OpenURL theURL toWindow newWindowID
end tell
```

Unfortunately, this script doesn't work, and it brings up an important point about writing AppleScripts: there are many different ways to do the same thing in AppleScript, and due to quirks in the application's dictionary, some of these ways may not work. Don't give up, simply try it another way. In this case, since we only use the reference to the new window once, we don't need to store it in the newWindowID variable. By combining the two lines into one, the new working script looks like this:

```
set theDialog to display dialog "Enter the URL to open:" default answer ""
set theURL to text returned of theDialog
tell application "Netscape Navigator"
      OpenURL theURL toWindow (make new window)
end tell
```

This script works for Navigator, but if you try it in Explorer, you'll find out that Explorer doesn't support the "make new window" command; yet another quirk in an application's dictionary. So now I adopt a trial-and-error approach. Since window IDs are simply numbers, I try to reference a window directly:

```
set theDialog to display dialog "Enter the URL to open:" default answer ""
set theURL to text returned of theDialog
tell application "Internet Explorer"
      OpenURL theURL toWindow 1
end tell
```

It turns out that for some strange reason, this does exactly what we want. This is highly inconsistent with other applications and illustrates the point that AppleScript can be quirky. Try things a few different ways in order to get it working correctly.

pple Data Detectors

Also known as Internet Address Detectors, this is a new technology add-on for Mac OS 8 that Apple released about 9 months ago, and it is a very cool way to easily manipulate things like URLs and email addresses. If you don’t have it, I highly recommend getting it (see the Resources section at the end of this article for the URL). If you do have it, great! You can now customize it because, as it turns out, the Actions in ADD/IAD are AppleScripts.

If you have this package installed, there should be a folder called AppleData Detectors in your System Folder. Inside that folder is a folder called Actions. Open the Actions folder, and you will see a list of files which make up the actions within ADD. Drag the action named "Open HTTP in Netscape" onto your Script Editor icon. This will open the script in the editor. Don't be intimidated by stuff like "«event std1hddt» inStructure" — we don't care about that. What we want to concentrate on is the stuff we already know about, which is customizing the Navigator commands.

It so happens that OpenURL and GetURL are very similar commands. You could insert the OpenURL command that we create above in this script to get the same behavior (new URLs are opened in their own window). Or you could look in the dictionary at the GetURL command and notice that it has an optional parameter called "inside" (i.e. Get URL myURL inside (make new window)"). You may also want to get rid of the "launch," "run" and "activate" commands (all of which are redundant) so that the browser stays in the background when the action is executed. You can do whatever you want to customize these scripts.

After you save the script in the Script Editor, the icon will change because the creator code has changed. You may wish to change this code back to its original state, though it's not necessary for the script to work properly.

My New Productivity Script

Getting back to the problem I described at the beginning of the article, it should now seem rather simple to write such a script. Using what we know about browsers and the OpenURL command, here is the final version of my script:

```
property URL1 : "http://my.excite.com/"
property URL2 : "http://www.news.com/"
property URL3 : "http://www.macintouch.com/"
```

```
property URL4 : "http://www.macosrumors.com/"
property URL5 : "http://www.maccentral.com/"

tell application "Netscape Navigator"
	OpenURL URL1 toWindow (make new window)
	OpenURL URL2 toWindow (make new window)
	OpenURL URL3 toWindow (make new window)
	OpenURL URL4 toWindow (make new window)
	OpenURL URL5 toWindow (make new window)
end tell
```

cripting Additions

Sometimes applications and AppleScript do not provide the functionality we need in order to write a script. AppleScript supports Scripting Additions for just these situations. Scripting Additions are a bit like Extensions on your computer; they exend the functionality of AppleScript by adding new commands for you to use in your scripts. One of my favorite OSAXEN (as they are sometimes called) is "Jon's Commands." This addition supports commands such as "set cursor" (so you can use a busy cursor to indicate your script is running) and "keys pressed" (which allows users to interact with your script via the keyboard).

Where do We Go from Here?

Of course, this is a fairly basic introduction to AppleScript. I should point out that AppleScript can do a lot more than what I described in these simple examples. If you are interested in more advanced scripting, you should check out the web sites and/or books listed below. I also highly encourage you to download and use other people's scripts. Learning by example is very effective, and there are some great scripting communities out there (such as the Claris Emailer one, to which I belong).

When you write or modify a script, have faith in yourself. As I pointed out before, there are a lot of quirks in AppleScript, and scripting turns out to be more of an art than a science. Don't be afraid to try different things and experiment. Remember, it is very difficult to break anything using AppleScript.

pilogue

Now I don't mean to take sides in a platform war, but allow me to quote from The 75 Macintosh Advantages — Why Macintosh computers are better than PCs running Windows:

53. The Mac OS has AppleScript automation. A big part of the next generation of personal computing is end-user automation — giving users the ability to automate their computers and tasks using plain English and point-and-click commands. AppleScript — the built-in, systemwide scripting capability of the Mac OS — lets you automate routine and highly complex tasks, giving you extremely powerful ways to extend and customize the features of the Macintosh.

Windows does not include any systemwide scripting or automation capability.

esources

Apple Computer's AppleScript Sites
http://applescript.apple.com/default.html
http://devworld.apple.com/dev/techsupport/insidemac/AppleScriptLang/AppleScriptLang-2.html
http://devworld.apple.com/dev/techsupport/insidemac/AppleScriptFind/AppleScriptFind-2.html

Other AppleScript Sites
http://www.scriptweb.com/
http://www.documentation.com/applescript/applescript.html
http://www.scripting.com/
http://applescript.infovista.com/
http://applescript.infomatters.com/

Apple Data Detectors
http://applescript.apple.com/data_detectors/

Scripting Additions (OSAXEN)
ftp://ftp.cadence.com/pfterry/applescript/osaxen/
http://www.natural-innovations.com/as/osaxref.html
http://www.scriptweb.com/scriptweb/osaxen/default.html

Claris Emailer Scripting
http://www.fogcity.com/em_utilities2.0.html
http://www.access.ch/private-users/swelter/emailerscript.html
http://www.claris.com/support/products/emailer/scripting/ScriptingIntroduction.html

FileTyper Home Page
http://www.ugcs.caltech.edu/~dazuma/filetyper/

The 75 Macintosh Advantages
http://www.apple.com/whymac/advantagecond/advcond.html

Why Do People Prefer Mac?
http://www.apple.com/whymac/brochure/why.html

David's Favorite News Sites
http://my.excite.com/
http://www.news.com/
http://www.macintouch.com/
http://www.macosrumors.com/
http://www.maccentral.com/

Great Books
AppleScript for Dummies by Tom Trinko (IDG Books)
BMUG's The Tao of AppleScript by Derrick Schneider (Hayden Books)

A bit About the Author

David Cortright is a human-computer interface designer who has worked for companies such as Claris, Oracle, Macromedia, Infoseek, and Lotus. Although, he learned the basics of programming at Stanford, he is definitely not a programmer. In addition to AppleScript, he has worked with other scripting languages such as HyperTalk, Lingo, VBScript, and JavaScript. He is a Claris Emailer enthusiast and has written numerous freeware scripts for the Emailer user community. These scripts are available at http://www.fogcity.com/em_utilities2.0.html .

David Cortright
davidc@cs.stanford.edu