Introduction

Technology Overview

Mac OS X supports scripting languages that include AppleScript, JavaScript, Perl, PHP, Ruby, Python, and various shell script dialects. To move beyond the standard features of these languages, Mac OS X provides programming interfaces so that developers can make their applications user scriptable (or OSA compliant). This allows you, for example, to use AppleScript, a native Mac OS X scripting language, to write scripts that combine features from Mac OS X Finder, Apple applications such as Mail, iTunes, and QuickTime Player, and scriptable third-party applications such as web, database, and spreadsheet products.

By integrating scripting support with many Apple technologies, Mac OS X can help you automate tasks ranging from system administration, to web content creation, to implementation of complex multiapplication workflows.

Scripting environments that are bridged to the Mac OS X Objective-C runtime (including Ruby, through RubyCocoa, Python, through PyObjC, and AppleScript, through AppleScript Studio) can access Objective-C frameworks, including Apple's Cocoa framework. Through the technology of framework metadata, these bridges can also provide access to functions and data types in most C-language frameworks.

Other Apple and open source software bridges make it easier to take advantage of features offered by scriptable applications. For example, Scripting Bridge—available starting in Mac OS X version 10.5 (Leopard)—allows Cocoa applications or other Objective-C code to efficiently access scriptable applications using native Objective-C syntax. Scripting languages such as Ruby and Python can also use Scripting Bridge in Leopard; in addition, they have their own open source bridges (RubyOSA and py-appscript) to scriptable applications running in Leopard or Tiger.

Mac OS X provides the Web Kit, which is used by Safari and Dashboard to render web content and widgets. To develop an interactive website or widget, you use JavaScript, Web Kit DOM APIs, and in the case of widgets, Dashboard-specific JavaScript APIs. In addition, several technologies, including Installer, Quartz Composer, and iSync Plug-in Maker, provide scripting environments that use JavaScript.

Beyond standard scripting languages, Apple's Automator application offers a drag-and-drop process for selecting and connecting individual operations, known as actions. Automator includes many actions that target Mac OS X and popular Apple applications, so users can automate a wide variety of tasks. Using Xcode templates, developers can create actions that make features from their applications available to Automator users.

Start Here

Before you begin to use scripting on Mac OS X, you should read:

- Introduction to Open Source Scripting on Mac OS X for an overview
- The ADC topic Open Source: Scripting for a list of supported open source scripting languages, with links to websites for those languages
- Getting Started with AppleScript for a quick introduction to Automator, AppleScript, and AppleScript Studio
- Getting Started with Internet and Web for information on developing web content and applications for the web in Mac OS X, including the use of JavaScript, Web Kit DOM, and Apple's WebObjects

Choose a Learning Path

The uses for scripting and automation in Mac OS X are as varied as the goals and inventiveness of the developers who use them. The following are guidelines for performing some common tasks.

Automation with Automator

Developers, system administrators, and other Mac users can use Automator to automate repetitive tasks for themselves and their clients. For introductory tips, sample actions, and links to additional Automator resources, see Automator Resources.

Development with Automator

Developers can use Xcode to build actions that allow Automator users to access key features of their applications. Starting in Mac OS X version 10.5, you can also embed and execute Automator workflows in an application.

- **To learn how to create Automator actions**, read Automator Programming Guide, which describes how to create AppleScript-based, Objective-C-based, and shell-script-based actions. For sample Xcode projects for building actions, see `<Xcode>/Examples/Automator`. For information about the classes, constants, and other symbols you use with Automator, read Automator Framework Reference.
- **To embed Automator workflows in your application**, see the class descriptions for `AMWorkflow`, `AMWorkflowView`, and `AMWorkflowController` in Automator Framework Reference, as well as Automator Release Notes.

Automation and Development with AppleScript

See Getting Started with AppleScript for detailed descriptions of the following learning paths:

- Writing AppleScript scripts to automate scriptable applications for script writers.
- Making your application scriptable for application developers.

Using Ruby and Python with Mac OS X Frameworks

You have several options for accessing Mac OS X frameworks from Ruby and Python:

- **To create Mac OS X applications using the RubyCocoa and PyObjC bridges to the Objective-C runtime**, see Ruby and Python Programming Topics for Mac. (For additional information on using Ruby, see the featured article Using Ruby on Rails for Web Development on Mac OS X.)
- **To access functions or constants defined by most Mac OS X frameworks from Ruby and Python scripts**, see Generating Framework Metadata in Ruby and Python Programming Topics for Mac.

Accessing Scriptable Applications with Scripting Bridge

You can access scriptable languages from Cocoa applications (or other Objective-C code) and from scripting languages such as Ruby and Python.

- **To use Scripting Bridge to access scriptable applications from Cocoa applications**, see Scripting Bridge Programming Guide for Cocoa and Scripting Bridge Framework Reference.
- **To use Scripting Bridge to access scriptable applications from Ruby and Python**, see Ruby

and Python Programming Topics for Mac.

Scripting for System Administrators

Shell scripts are a fundamental part of the Mac OS X programming environment, and they can be particularly useful for system administrators. See Shell Scripting Primer to jump start your shell script development.

In addition, you can execute shell scripts both with AppleScript scripts and with Automator actions. For information on the former, see “Scripting With AppleScript” in AppleScript Overview. For the latter, see “Creating Shell Script Actions” in Automator Programming Guide.

Web Content and Dashboard Widget Development with JavaScript and Web Kit DOM

- **To develop an interactive website with JavaScript and Web Kit DOM**, see Apple JavaScript Coding Guidelines and Web Kit DOM Programming Topics.
- **To make direct calls to the underlying JavaScript engine used by WebKit**, see JavaScriptCore Framework Reference.
- **To develop Dashboard widgets**, see Dashboard Programming Topics.

Scripting with QuickTime

For information on how to automate and control browser operations with QuickTime and how to script operations with QuickTime Player, see QuickTime Scripting.

Application Scripting with JavaScript

You can use JavaScript in various Mac OS X technologies.

- **To write an iSync plug-in with JavaScript**, see iSync Plug-in Maker User Guide.
- **To use JavaScript to write Installer scripts**, see Installer JavaScript Reference.
- **To work with JavaScript in Quartz Composer**, see the section “Using Programming Patches” in the chapter “Basic and Advanced tasks, Tips, and Tricks” in Quartz Composer User Guide.

Creating an AppleScript Studio Application

You can use AppleScript Studio to quickly create applications with complex user interfaces. For details, see the learning paths in Getting Started with AppleScript.

Next Steps

Scripting & Automation Reference Library includes the following high-level resource pages, which you can bookmark for easy access:

- Guides

  Conceptual and how-to information for scripting and automation.
- Featured Articles

  Articles on working with scripting and automation.

- Release Notes

  Late-breaking news and highlights of new or changed features in the latest releases.

- Sample Code

  Examples for writing code that works with scripting and automation.

- Technical Notes

  Late-breaking documents on timely technology issues.

- Technical Q&As

  Programming tips, code snippets, & FAQs by Apple's support engineers.

For mailing lists where users and developers can share experiences, questions, and comments with other interested parties on technologies described in this document:

- AppleScript mail lists include `applescript-implementors` for issues related to scriptable applications, `applescript-users` for issues related to writing AppleScript scripts, and applescript-studio for issues related to AppleScript Studio.
- Automator mail lists include `automator-dev` for developers creating new Automator actions and `automator-users` for users developing workflows and solutions using Automator.
- For Dashboard development, see `Dashboard-dev`.
- For Java development, see `Java-dev`.
- For administrators of Mac OS X Server and related technologies, see `macos-x-server`.
- For QuickTime for Java development, see `QuickTime-Java`.
- For client-side and server-side Web and Internet development on the Macintosh, see `Web-dev`.
- For Quartz Composer development, see `Quartzcomposer-dev`.
- For iSync development, see `syncservices-dev`.
- For Apple software installation technologies, see `installer-dev`.