



AppleScript[®]



AppleScript[®]

AppleScript®

Mark Conway Munro



WILEY

Wiley Publishing, Inc.

AppleScript®

Published by
Wiley Publishing, Inc.
10475 Crosspoint Boulevard
Indianapolis, IN 46256
www.wiley.com

Copyright © 2010 by Wiley Publishing, Inc., Indianapolis, Indiana

Published by Wiley Publishing, Inc., Indianapolis, Indiana

Published simultaneously in Canada

ISBN: 978-0-470-56229-1

Manufactured in the United States of America

10 9 8 7 6 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except as permitted under Sections 107 or 108 of the 1976 United States Copyright Act, without either the prior written permission of the Publisher, or authorization through payment of the appropriate per-copy fee to the Copyright Clearance Center, 222 Rosewood Drive, Danvers, MA 01923, (978) 750-8400, fax (978) 646-8600. Requests to the Publisher for permission should be addressed to the Permissions Department, John Wiley & Sons, Inc., 111 River Street, Hoboken, NJ 07030, 201-748-6011, fax 201-748-6008, or online at <http://www.wiley.com/go/permissions>.

LIMIT OF LIABILITY/DISCLAIMER OF WARRANTY: THE PUBLISHER AND THE AUTHOR MAKE NO REPRESENTATIONS OR WARRANTIES WITH RESPECT TO THE ACCURACY OR COMPLETENESS OF THE CONTENTS OF THIS WORK AND SPECIFICALLY DISCLAIM ALL WARRANTIES, INCLUDING WITHOUT LIMITATION WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE. NO WARRANTY MAY BE CREATED OR EXTENDED BY SALES OR PROMOTIONAL MATERIALS. THE ADVICE AND STRATEGIES CONTAINED HEREIN MAY NOT BE SUITABLE FOR EVERY SITUATION. THIS WORK IS SOLD WITH THE UNDERSTANDING THAT THE PUBLISHER IS NOT ENGAGED IN RENDERING LEGAL, ACCOUNTING, OR OTHER PROFESSIONAL SERVICES. IF PROFESSIONAL ASSISTANCE IS REQUIRED, THE SERVICES OF A COMPETENT PROFESSIONAL PERSON SHOULD BE SOUGHT. NEITHER THE PUBLISHER NOR THE AUTHOR SHALL BE LIABLE FOR DAMAGES ARISING HEREFROM. THE FACT THAT AN ORGANIZATION OR WEBSITE IS REFERRED TO IN THIS WORK AS A CITATION AND/OR A POTENTIAL SOURCE OF FURTHER INFORMATION DOES NOT MEAN THAT THE AUTHOR OR THE PUBLISHER ENDORSES THE INFORMATION THE ORGANIZATION OR WEBSITE MAY PROVIDE OR RECOMMENDATIONS IT MAY MAKE. FURTHER, READERS SHOULD BE AWARE THAT INTERNET WEBSITES LISTED IN THIS WORK MAY HAVE CHANGED OR DISAPPEARED BETWEEN WHEN THIS WORK WAS WRITTEN AND WHEN IT IS READ.

For general information on our other products and services or to obtain technical support, please contact our Customer Care Department within the U.S. at (877) 762-2974, outside the U.S. at (317) 572-3993 or fax (317) 572-4002.

Library of Congress Control Number: 2010925705

Trademarks: Wiley and the Wiley logo are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. AppleScript is a registered trademark of Apple, Inc. All other trademarks are the property of their respective owners. Wiley Publishing, Inc. is not associated with any product or vendor mentioned in this book. AppleScript® Developer Reference is an independent publication and has not been authorized, sponsored, or otherwise approved by Apple, Inc.

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

***To my father,
Philip Conway Munro,
for teaching me the difference between
hardware and software at an early age.***

Credits

Acquisitions Editor

Aaron Black

Executive Editor

Jody Lefever

Project Editor

Katharine Dvorak

Technical Editor

Rob Vanderwerf

Copy Editor

Lauren Kennedy

Editorial Director

Robyn Siesky

Business Manager

Amy Knies

Senior Marketing Manager

Sandy Smith

Vice President and Executive Group Publisher

Richard Swadley

Vice President and Executive Publisher

Barry Pruett

Project Coordinator

Lynsey Stanford

Graphics and Production Specialists

Andrea Hornberger

Erin Zeltner

Quality Control Technician

John Greenough

Proofreading

C. M. Jones

Indexing

WordCo Indexing Services

Media Development Project Manager

Laura Moss

Media Development Assistant Project Manager

Jenny Swisher

Media Development Associate Producer

Doug Kuhn

About the Author

Mark Conway Munro is an entrepreneur who turned a software-writing hobby into a business. Mark was born in Indiana and raised in Ohio. In 1986, Mark moved to New York City. As the manager of Louis Tannen's Magic Store in New York City, he taught himself the Macintosh Computer and FileMaker Pro while building a database of inventory, which eventually led to the development of a networked order processing solution. From there he went on to work for the Jack Morton Company, where he was the Network and Telecommunications Manager. He began developing custom solutions with HyperCard and FileMaker Pro in his spare time. After the release of AppleScript in 1992, he refocused his experience with computer networking, training, technical support, database development, and custom application development. He quickly transitioned from HyperTalk to AppleScript and began building custom workflow automation solutions.

Mark founded Write Track Media in 1994, where he continues to develop innovative solutions that eliminate repetition and optimize computerized workflows. Write Track Media has since become known for its reputation for excellence, and has developed complex automated solutions for companies in a variety of industries all across the country including Adobe Systems, The Associated Press, BMG, Dreyfus, Entertainment Weekly, Epson, KraftMaid, McCann-Erickson, The Miami Herald, MYOB, Nabisco, NASA, Nikon, Random House, Reader's Digest, Sony Music Entertainment, and many other companies.

Currently Mark resides in Pennsylvania, where Write Track Media is located. In his spare time, he enjoys hiking and nature photography.

Contents at a Glance

Foreword	xxi
Acknowledgments.....	xxiii
Introduction	xxv

Part I: AppleScript: The Power of Automation..... 1

Chapter 1: Introduction to AppleScript Programming	3
Chapter 2: Workflow Automation with AppleScript.....	21
Chapter 3: AppleScript Deployment Options.....	31
Chapter 4: Making the Case for Standardization	59

Part II: Learning the AppleScript Language..... 81

Chapter 5: Exploring AppleScript Basics.....	83
Chapter 6: Getting Started with the AppleScript Editor	111
Chapter 7: Working with Text Objects	137
Chapter 8: Working with Numbers and Unit Types	159
Chapter 9: Working with Dates and Times.....	171
Chapter 10: Other AppleScript Data Classes	181
Chapter 11: Working with Lists.....	189
Chapter 12: Working with Records	205
Chapter 13: Logical Branching.....	213
Chapter 14: Looping	227
Chapter 15: Dealing with Script Errors	245
Chapter 16: Getting Started with Scripting Additions.....	267

Part III: Using Scripts to Control Applications..... 311

Chapter 17: Controlling Applications with Scripts.....	313
Chapter 18: Working with Image Events	339
Chapter 19: Working with Database Events.....	361
Chapter 20: Working with System Events.....	385

Part IV: Using Subroutines and Open-Ended Programming 409

Chapter 21: Using Subroutines for Non-Linear Programming.....	411
Chapter 22: Introduction to Open-Ended Programming	433
Chapter 23: Designing a Hierarchical Subroutine Structure	469

Part V: Organizing Code into Modules and Libraries for Multi-Module Solutions 501

Chapter 24: Introduction to Multi-Module Solutions	503
Chapter 25: Designing Open-Ended, Multi-Module Solutions.....	535

Appendix: AppleScript Web Resources	563
---	-----

Index	567
-------------	-----

Contents

Foreword	xxi
Acknowledgments.....	xxiii
Introduction	xxv

Part I: AppleScript: The Power of Automation..... 1

Chapter 1: Introduction to AppleScript Programming..... 3

A Brief History of AppleScript.....	5
Finding AppleScript Resources	7
Applications	8
Scripting additions	9
Components	9
Understanding the Unique Characteristics of AppleScript.....	10
English-like syntax	10
Universally open-ended	13
Deep level of access.....	13
Consistency maintained between updates	14
Exploring the Uses and Users of AppleScript.....	14
Uses for AppleScript	15
Users of AppleScript	18
Respecting the Power of AppleScript.....	18
Summary.....	19

Chapter 2: Workflow Automation with AppleScript 21

Defining Workflow Automation	22
Busting some myths about automation	22
Exploring reasons to automate a workflow	25
Using AppleScript for Workflow Automation.....	26
Understanding the scalability of AppleScript.....	27
Quantifying the return on investment potential of AppleScript.....	29
Summary.....	30

Chapter 3: AppleScript Deployment Options 31

Exploring AppleScript Formats	31
Saving scripts as files	32
Saving scripts as applications	36
Exploring Script Deployment Locations	39
Mac OS X installation locations	39
Mac OS X usage locations	47
Third-party application locations.....	53
Choosing a Format and Location	56
Summary.....	58

Chapter 4: Making the Case for Standardization 59

Understanding the Benefits of Standards	59
Consistency	60
Repurposing	60
Enhanced efficiency	60
Improved quality	60
Collaboration	61
Automation	61
Professionalism	61
Exploring the Flexibility of Standards	61
Applying standards within a context	61
Setting your own standards	62
Defining AppleScript Naming Standards	62
The goals of naming standards	62
Naming variables	65
Naming subroutines	67
Defining AppleScript Usage Standards	71
The goals of usage standards	71
AppleScript usage standards	72
Summary	80

Part II: Learning the AppleScript Language..... 81**Chapter 5: Exploring AppleScript Basics. 83**

Understanding AppleScript Terminology	83
Commands	84
Literals	84
Keywords	85
Operators	87
Object classes	89
Variables	95
Statements	97
Subroutines	99
Scripts	99
Looking at AppleScript Comments	100
Commenting methods	100
Uses for comments	101
Frequency of comments	104
Commenting usage conventions	105
Planning Scripts	107
Pre-coding steps	107
Coding steps	109
Summary	110

Chapter 6: Getting Started with the AppleScript Editor.....111

Exploring the AppleScript Editor User Interface.....	112
The script document window	112
Contextual menus	116
Menus	116
Preferences.....	122
Library window	127
Event Log History window.....	129
Building the “Hello World” Script	132
Creating the script	132
Expanding the script.....	134
Summary.....	136

Chapter 7: Working with Text Objects.....137

Introduction to Text Objects	137
Text object properties.....	137
Special consideration for quotes.....	140
Analyzing Text.....	140
Counting text	140
Searching text	142
Comparing text.....	145
Considering and ignoring text	147
Manipulating Text	150
Merging text.....	150
Splitting text	150
Extracting text	151
Converting text to other data types	154
Using text item delimiters	155
Summary.....	157

Chapter 8: Working with Numbers and Unit Types159

Introduction to Number Objects	159
Looking at types of numbers	159
Putting a number into a variable	160
Comparing numbers.....	161
Manipulating Numbers	162
Performing calculations.....	162
Converting numbers.....	165
Working with Measurement Unit Types	167
Using measurement types.....	168
Converting within the type group	168
Converting to other data types	168
Summary.....	169

Chapter 9: Working with Dates and Times	171
Introduction to Date Objects	171
Date object properties	172
Manipulating Date and Time	175
Comparing dates	175
Performing calculations with dates and times	177
Summary	179
Chapter 10: Other AppleScript Data Classes	181
Working with Booleans	181
Working with RGB Colors	182
Working with Aliases	183
Working with Files	185
Working with References	186
Summary	188
Chapter 11: Working with Lists	189
Introduction to Lists	189
Looking at list properties	190
Looking at specialty lists	191
Analyzing Lists	192
Counting list items	192
Comparing lists	193
Considering and ignoring text properties	196
Searching in lists	197
Manipulating Lists	197
Converting lists to other data types	198
Extracting list items	198
Adding items to a list	201
Replacing items in a list	202
Removing items from a list	202
Summary	203
Chapter 12: Working with Records	205
Introduction to Records	205
Comparing an AppleScript record to a database record	206
Looking at record properties	206
Creating a Record	207
Analyzing Records	208
Counting records	208
Comparing records	208
Considering and ignoring text properties	210

Manipulating Records	210
Converting records to other data types.....	210
Extracting data from a record	210
Adding something to a record	211
Replacing a value in a record	211
Removing a value from a record.....	212
Summary.....	212

Chapter 13: Logical Branching.....213

Looking at the Anatomy of an if-then Statement	213
Building a conditional “Hello World” dialog	214
Expanding the equation	215
Creating a Multiple Condition Statement	217
Adding Additional Conditions.....	219
Using Nested Statements	221
Understanding Common Mistakes.....	222
Missing parenthesis.....	223
Missing conditions.....	223
Conditionally undefined variables	224
Summary.....	225

Chapter 14: Looping.....227

Looking at the Anatomy of a Repeat Statement	228
Defining the Types of Repeat Loops.....	229
Repeat (until exit)	229
Repeat x times.....	230
Repeat while	230
Repeat until.....	231
Repeat with a from x to y {by z}.....	231
Repeat with a in list.....	233
Nesting Repeat Loops.....	233
Using Repeat Loops	235
Creating with repeat loops	235
Modifying with repeat loops	236
Extracting with repeat loops.....	240
Processing files with repeat loops	242
Summary.....	243

Chapter 15: Dealing with Script Errors245

Introduction to Script Errors	245
Defining programming errors.....	246
Defining situational errors.....	249
Exploring Error Management	251
Looking at the anatomy of a try command.....	252
Handling multiple errors	258
Generating your own errors.....	259
Understanding cascading errors.....	260

Recording Errors Into a Log File	261
Writing information to the event log.....	261
Writing errors to text files.....	262
Looking at the AppleScript and Mac OS X Errors	263
Summary	265

Chapter 16: Getting Started with Scripting Additions 267

Finding Scripting Additions	267
System Library folder.....	267
Library folder	268
User's Home Library folder	268
Embedding Scripting Additions.....	268
Working with Standard Additions	269
User Interaction	270
File Commands.....	285
String Commands.....	289
Clipboard Commands	291
File Read/Write	293
Scripting Commands	297
Miscellaneous Commands	300
Folder Actions.....	304
Internet	309
Summary	310

Part III: Using Scripts to Control Applications..... 311

Chapter 17: Controlling Applications with Scripts 313

Introduction to Application Automation.....	313
Looking at the “tell application” statement	314
Managing timeouts	314
Ignoring an application response	316
Respecting hierarchy when nesting disparate control commands	317
Defining different types of AppleScript support.....	318
Exploring an Application's Dictionary	322
Opening a dictionary	322
Exploring the dictionary interface.....	323
Exploring an application's dictionary content.....	324
Using AppleScript to Control Applications	328
Activating, launching, and quitting applications	328
Manipulating the Finder with scripts.....	329
Controlling Inter-Application Communication.....	334

Controlling Remote Applications	335
Configuring Remote Apple Events	335
Understanding eppc computer specifiers	336
Sending commands to a remote application	337
Compiling a script using terms from a local application	337
Summary.....	338
Chapter 18: Working with Image Events	339
Introduction to Image Events	339
Getting started with basic functionality	340
Reading properties of an image file	345
Manipulating an image	348
Creating an Image Batch Processor	354
Summary.....	359
Chapter 19: Working with Database Events	361
Introduction to Database Events.....	361
Getting started with basic functionality	362
Working with database records	366
Working with fields.....	368
Searching a Database	372
Searching for text values	372
Searching for numeric values	375
Searching for date values	376
Searching for multiple values	378
Importing Records from Tab-Separated Files.....	379
Summary.....	384
Chapter 20: Working with System Events	385
Introduction to System Events.....	385
Getting started with basic functionality	386
Exploring the suites of commands.....	387
Controlling Non-Scriptable Applications.....	393
Enabling User Interface Scripting	393
Activating and targeting applications	395
Referencing objects in an application's interface.....	396
Accessing information from an interface.....	400
Performing User Interface Scripting actions	404
Creating a Zipped Archive File with System Events.....	407
Summary.....	408

Part IV: Using Subroutines and Open-Ended Programming 409

Chapter 21: Using Subroutines for Non-Linear Programming..... 411

Working with Subroutines.....	413
Calling a subroutine from a tell application statement.....	414
Exchanging data with subroutines	415
Identifying command handler subroutines	421
Commenting subroutines	424
Exploring the Benefits of Subroutines	425
Easing developer tasks	425
Reusing code	428
Allowing advanced script design	429
Resolving variable names conflicts	429
Designing a Non-Linear Script	430
Understanding when to delimit a script into subroutines.....	430
Looking at the methods of delimiting a script.....	431
Summary.....	432

Chapter 22: Introduction to Open-Ended Programming 433

Understanding the Benefits of Open-Ended Code.....	434
Makes recycling code easy.....	434
Improves script quality.....	434
Encourages consistency	435
Justifies smaller scripts.....	435
Creating Open-Ended Code.....	435
Use repeat loops to remove duplicate code	437
Use the Finder selection	438
Provide for an empty selection	439
Allow folders to be processed	440
Dynamically count the name's length	440
Use text item delimiters	441
Use variables or script properties.....	442
Query user input.....	443
Use subroutines.....	443
Creating Open-Ended Subroutines	445
Divide code with logical groupings.....	445
Make smaller subroutines	446
Name subroutines, parameters, and variables generically	447
Avoid branch-style openness.....	448
Use subroutine parameters for variable input	449
Use records for future parameter expansion.....	450
Keep subroutines as portable as possible	451

Creating an Open-Ended Image Batch Processor	451
Allow a user to select folders	451
Enable two additional selection methods.....	452
Choose manipulations	454
Select a custom scale percentage	455
Allow the user to choose an output format	458
Allow the selection of multiple output formats.....	459
Make a drop application	461
Use subroutines.....	464
Bring it all together	464
Summary.....	468

Chapter 23: Designing a Hierarchical Subroutine Structure.....469

Defining the Goals of Subroutine Hierarchy	472
Produce a flexible and expandable script	473
Maximize reusable code	473
Create portable code	473
Achieve a separation of data from function	473
Facilitate a multi-module solution ideology.....	473
Identifying the Primary Levels of Hierarchy	473
Maintaining flexibility within levels	475
Following proper inter-level communication	475
Identifying Hierarchy-Related Issues.....	475
Project-specific elements	476
Open-ended elements.....	478
Creating a Image Batch Processor with a Hierarchical Subroutine Structure.....	483
Outlining the new subroutine structure	487
Rebuilding the script	488
Summary.....	499

Part V: Organizing Code into Modules and Libraries for Multi-Module Solutions 501

Chapter 24: Introduction to Multi-Module Solutions503

Understanding the Benefits of Multi-Module Solutions.....	504
Eases developer tasks	504
Allows advanced script design.....	504
Designing a Multi-Module Solution	505
Defining types of script files	505
Exploring file structure options for complex solutions	507
Understanding inter-script communication	512
Overcoming the Complexities of Multi-Module Solutions.....	517
Employing good development habits.....	517
Using logs for tracking.....	519

Building a Multi-Module Image Batch Processor	522
Designing the new solution	522
Building the Logging Module file	523
Building the Finder Library file	528
Building the Image Events Library file	529
Building the main Image Batch Processor file	530
Summary	534

Chapter 25: Designing Open-Ended, Multi-Module Solutions 535

Planning for Change	535
Anticipating business changes	535
Anticipating development changes	542
Upgrading the Image Batch Processor to an Open-Ended, Multi-Module Solution	547
Creating the new module template	547
Setting up the Scale module	550
Setting up the Flip module	551
Modifying the image batch processor module	552
Creating a new Rotate module	560
Further expansion and inspiration	562
Summary	562

Appendix: AppleScript Web Resources 563

Apple's Developer Resources	563
Developer Connection main page	563
AppleScript documentation and resource main page	563
Introduction to AppleScript overview page	564
AppleScript language guide	564
AppleScript release notes	564
Mailing Lists	564
Apple's AppleScript Users	564
Mac Scripting Systems (MACSCRIPT)	564
Alternative Script Editor Software	565
Script Debugger	565
Smile	565
Additional Scripting-Related Sites	565
MacScripter	565
UI Browser	565
MacTech's Visual Basic to AppleScript guide	565
Write Track Media	566

Index 567

Foreword

Twelve years ago Mark Munro asked me to write something for the Web site he was building for his company, Write Track Media. I had already been working with him for about three years: me as an editor and in-house FileMaker developer at the country's second-largest record company, and Mark as outside FileMaker and AppleScript developer. He had helped build the departmental Mac-based database of discs, tapes, and videos with up-to-date listings of all their artists, prices, genres, formats, and so on. But the major part of the project, and where Mark was proving so invaluable, was in the system's output.

We needed to produce a monthly pocket-sized catalog of the active product — about 13,000 data records. Mark automated production so that I could generate a 180-page complexly styled catalog — using FileMaker Pro, AppleScript, and Quark — at the push of a button. We needed to constantly produce multipage order forms with elaborate line listings and scannable barcodes. Mark automated these so that one button would trigger the form to build from scratch: Quark firing up, new blank documents opening, text boxes being created and placed, text flying into the boxes, picture boxes being created, AppleScript running off and building barcode images for each product, bringing them back, dropping them into the picture boxes, sizing them to fit.... All this looked like magic to the IT guys that would drop by occasionally, none of them Mac users, with no idea that an application like AppleScript existed that could make the programs all “talk” to each other. It was magic to us, too: the documents were data driven, accuracy was better than it ever had been, and the automation was saving countless hours of typing, page layout, and proofreading every week.

New record formats were coming into being; sales needs and the documents supporting them were changing often. I was pressed for time and got to work with Mark a lot. The thrill was that I could call him and tell him what I needed and he'd never get nervous or show hesitation, and he'd never say it couldn't be done. There was always a way, and for him, always a good way. He worked fast and methodically, kept me briefed, and generally delivered ahead of his target date. I would send lists of fields, find and sort rules, and layout requirements. He would send code. I wrote for his Web site, “There seems to be no limit to the complexity of the scripting and automation jobs they are able to take on, and they do it with an energy, focus, speed, and level-headed aplomb that are to be admired.” “They,” of course was all Mark himself, and I feel the same twelve years later.

It's not surprising to me that Mark, early in his career, was a performing magician who got his start with databases when he decided to computerize the inventory at Tannen's, the New York City magic supply mecca where he was working. I don't mean this in the hokey, “Oh, this guy works magic” sense; I mean that like a skilled performer he has brought to his work in these intervening years a rigorous, practiced, polished, and intensely methodical approach. He has developed a remarkable overview of the effects he wants to achieve. He has codified a highly refined aesthetic and has devoted himself to the rigorous practice that it takes to express it.

I went to the Apple Store last night and happened to mention to a young clerk that my friend was writing a developer guide to AppleScript. I was floored when he said, “Oh AppleScript — nobody talks about that around here — I think there's one guy that knows something about it and was thinking of using it to automatically update his phone, but he only knew how to get it

to turn on and off.” Luckily, that’s just ignorance, even if it’s coming from an Apple employee: AppleScript remains an unparalleled development tool for desktop automation. It’s not easy, and that’s why there are specialists like Mark Munro. Sometimes it’s hard to envision what it’s capable of because it takes effort to zoom out, like a movie camera craning overhead, to get a real sense of what’s possible to automate. Then building the solution can be daunting and time-consuming. It takes an effort to stop what you’re doing by rote, and to devote time to changing direction. The resulting time savings and ease of operation can be thrilling.

I’ve heard a professional developer say, “What is good code? Good code is code that works.” I don’t think Mark would say that. There’s code that works and there’s good code that works. It’s not a fussy, perfectionist approach, either, just a sense of the intrinsic RIGHTness of an approach. He abhors wasted time and repetition of effort and spends remarkable effort in streamlining his FileMaker development practices and templates. This is to save time and money for his clients and, I think, to keep himself moving forward. It’s obvious to me why his company’s motto is: “Write Track Media creates solutions that eliminate repetition and allow our clients to focus on their highest potential.”

As a fledgling FileMaker developer, I have to admit that working with Mark over the years has been an occasional wellspring of inferiority feelings: “Is this the standard? Can I ever be a developer who’s worth his salt if I’m not as good as Mark?” Maybe I can relax a little. I’ve come to believe that in all likelihood Mark’s unique: perhaps there’s no one as methodical and philosophically rigorous at what he does. But now that he’s taking the time to codify his theories and his method, maybe we can get a little closer.

Walker Stevenson
FileMaker Developer

Acknowledgments

Writing this book was a tremendous effort and would not have been possible without the support, contributions, and encouragement of many people.

First, I want to thank John Thorsen, Jr., for the development opportunities he provided me many years ago. He is a constant source of advice, knowledge, and humor that would be difficult to find anywhere within a single human being. Also, thanks goes to Rob Vanderwerf for reviewing the manuscript with a keen eye for technical consistency and accuracy. His feedback led to many key improvements that greatly enhanced the quality of the material. To my friend, Walker Stevenson, for his meticulous nature, a willingness to discuss technical details, and for writing a foreword that makes it sound like I know what I'm doing. And to Aaron Black, Katharine Dvorak, and everyone at Wiley for providing me with this opportunity and for all their hard work.

Finally, a special thanks to all of my wonderful clients, especially those who have worked with me as an extension of their staff for many years. They have presented me with one challenge after another, constantly forcing me to push beyond my comfort zone. Many of the advanced concepts in this book would not exist without them.

Introduction



In the early 1990s, I was working as a Computer Network and Telecommunications Manager at the New York City headquarters of a company now known as Jack Morton Worldwide. In addition to my other responsibilities, I was tasked with creating FileMaker Pro databases to help manage the office's information. At that time, FileMaker was still somewhat crude, and had been since I began using it in the late 1980s.

During this time, I learned how to program HyperCard and was having a lot of fun building custom applications. Then I received a copy of the AppleScript Developer's Toolkit and Scripting Kit, an add-on piece of software for the Macintosh Operating System. I began converting the HyperTalk scripts in my applications to AppleScript. Almost immediately HyperCard was relegated to my archives in favor of AppleScript.

This was still a year or two before the Internet exploded in popularity. Finding documentation and sample scripts was difficult, and very few applications supported scripting. These difficulties led to many frustrating and yet ultimately rewarding, struggles to figure out how to successfully automate one task after another. However, as support for scripting spread out into the Mac OS and in third-party software, it became clear that workflow automation on a Mac was going to be a huge phenomenon.

This was a driving factor for founding Write Track Media in 1994 to specialize in the development of workflow automation solutions. Since that time, I have worked with many wonderful clients in a variety of different industries. Many times since then, I have entertained the notion of writing a book on AppleScript. Some of the techniques and ideas that I used daily, many created out of the necessity of the moment, seemed good enough to share. Believing that good ideas only become great when they are shared with others was a driving factor that led me to avail myself of the opportunity to write this book.

I tried to structure this book to appeal to programmers of any skill level, including those who have never programmed before. I start with the basics and gradually work toward more advanced material to make this book beneficial to all. It is laced with advice, ideas, and techniques gleaned from years of trial and error. Hopefully, you will find these useful and encouraging in your future automation endeavors.

AppleScript is a wonderful language to know and is powerful enough to automate virtually any task. It was a rewarding experience to focus on the language in a systematic fashion while writing this book. I hope you enjoy reading it as much as I have writing it.

Please let me know about issues you find in the book or offer suggestions for subjects you'd like to see covered in a future edition. Visit www.writetrackmedia.com/contact/ to send me an e-mail.

Getting the Most Out of This Book

The chapters in this book are organized into five parts. They are organized with the assumption that a reader will read the book from cover to cover. The material starts with the basics and gradually moves to more advanced material.

Part I, “AppleScript: The Power of Automation” includes an introduction and history of AppleScript as well as discussions of workflow automation with AppleScript and script deployment options. It also includes a presentation of a comprehensive set of naming and usage standards.

Part II, “Learning the AppleScript Language” begins with a chapter on AppleScript basics followed by chapters detailing each class of data that can be manipulated with scripts. From there, chapters discuss logical branching with `if-then` statements, repeat loops, error containment and management, and an in-depth look at how the standard scripting addition extends AppleScript’s functionality.

Part III, “Using Scripts to Control Applications” reveals AppleScript’s capability to tap into the functionality of standard, off-the-shelf software to create powerful multi-application workflows. Also, the three “faceless” applications — Image Events, Database Events, and System Events — included with the Macintosh operating system that provide additional functionality for AppleScript are covered in detail.

Part IV, “Using Subroutines and Open-Ended Programming” contains chapters that discuss the creation of subroutines, writing open-ended code, and an advanced discussion of using a hierarchical method for dividing code into subroutines.

Part V, “Organizing Code into Modules and Libraries for Multi-Module Solutions” continues to push into more advanced topics, discussing the development of complex AppleScript solutions that are spread across more than one script file.

Parts III–V contain several versions of an Image Batch Processing script that evolves over five chapters. Each successive version of the script is improved with concepts that are introduced in that chapter.

All of the code in this book was created and tested in Mac OS X 10.6.

Using the Book’s Icons

There are four margin icons that are used throughout the book to provide additional information, tips, warnings, or indications of where to find additional information.

**NOTE**

Notes highlight useful information that you should take into consideration.

**TIP**

Tips provide additional bits of advice that make particular features quicker or easier to use.

**CAUTION**

Cautions warn you of potential problems before you make a mistake.

**CROSS-REF**

Watch for the Cross-Ref icon to learn where in another chapter you can go to find more information on a particular topic.

Accessing the Book's Web Site

Several of the longer scripts presented in this book are available for download on Wiley's companion Web site. Visit www.wileydevreference.com for more information and to download the script files.

AppleScript: The Power of Automation



In This Part

Chapter 1
**Introduction to
AppleScript
Programming**

Chapter 2
**Workflow Automation
with AppleScript**

Chapter 3
**AppleScript Deployment
Options**

Chapter 4
**Making the Case for
Standardization**

1

Introduction to AppleScript Programming

AppleScript is an Open Scripting Architecture (OSA)–compliant command language that can communicate with and control scriptable applications and scriptable features of Macintosh Operating System (Mac OS) X.

The OSA is a mechanism within the Mac OS that provides a library of functions and allows inter-application communication and control by sending and receiving messages called Apple Events.

An *Apple Event* is a basic message exchange system of the OSA that is used to send instructions to an application and optionally, send back a result. Apple Events can be used to control inter-process communication within an application, between applications on a single computer, and between applications on a remote computer. Figure 1.1 illustrates the path of an Apple Event message.

Since 1994, when System 7.5 was released, most Mac users have been unaware that they send Apple Events every day. For example, each time they double-click a document to open it, an Apple Event is responsible for instructing the appropriate application to launch and to open the file, as shown in Figure 1.2.

AppleScript provides an easy to learn, English-like language that enables users to write scripts that send and receive Apple Events. Each script acts like a new feature of the OS or an application. Scripts can integrate third-party applications, creating custom solutions that perform very specific tasks.

Because of its English-like syntax, even novice programmers can build scripts to perform virtually any function. With pervasive support of AppleScript throughout the Mac OS and many third-party applications, it is the ideal platform to create efficiency-rich, workflow automation solutions.

This amazing and award-winning technology provides a simple and affordable way to automate repetitive computing tasks and leave users free to focus their attention on more creative tasks.

1

In This Chapter

An introduction to AppleScript

Locating AppleScript applications and other resources

Looking at AppleScript's resources and unique characteristics

Who uses AppleScript and what they automate

Looking at AppleScript's influence

Figure 1.1

The path of an Apple Event message sending a command to an application

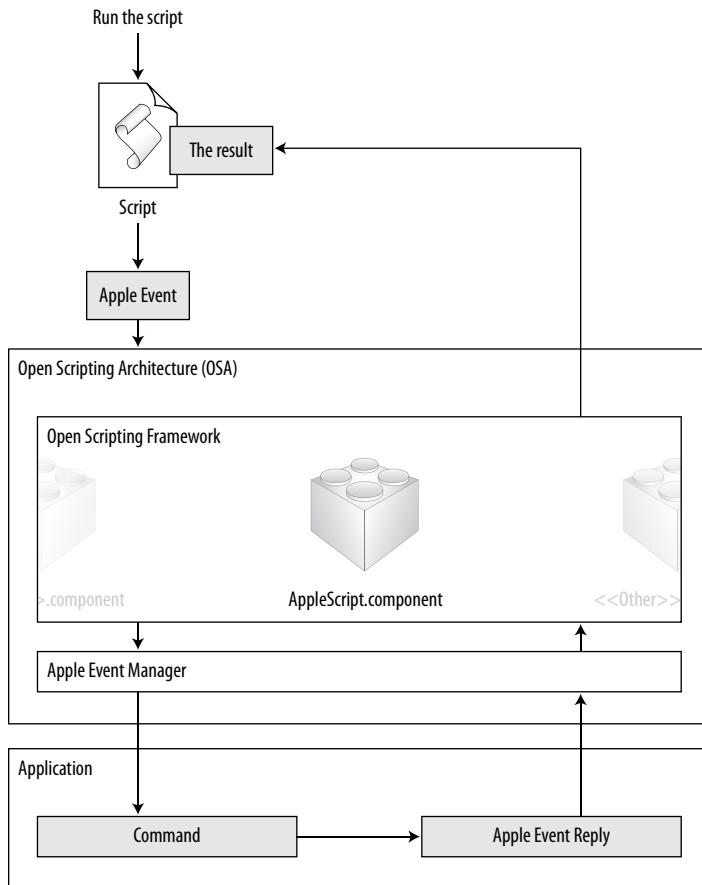
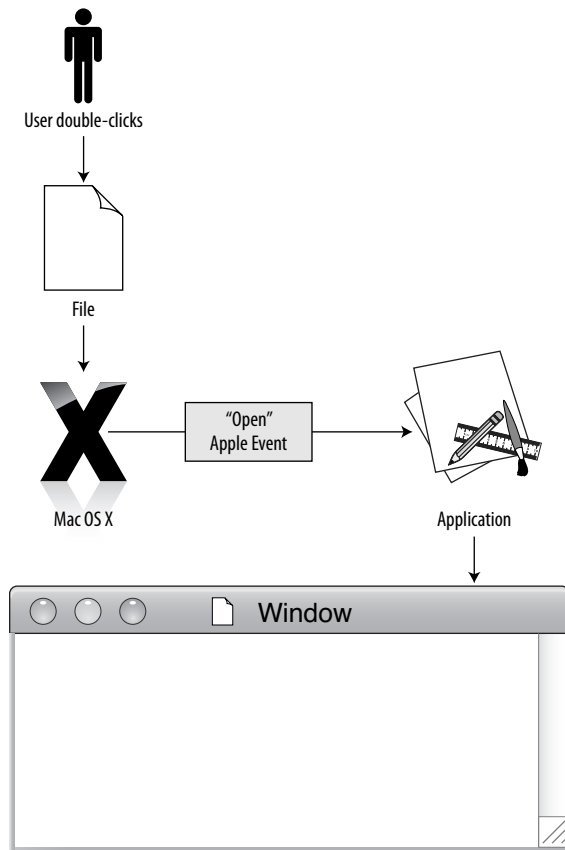


Figure 1.2

The underpinnings of an Apple Event opening a document

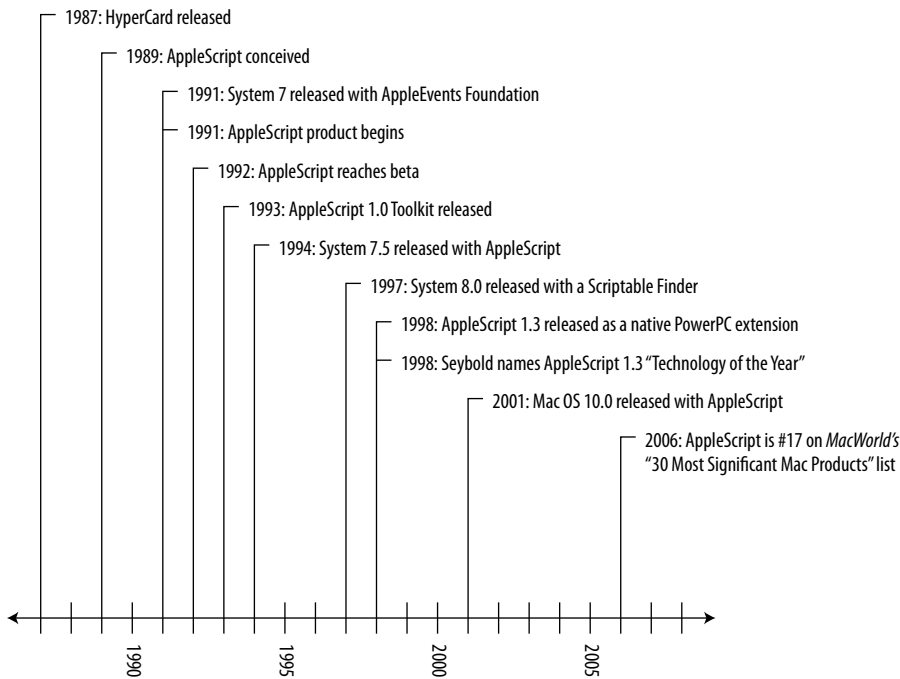


A Brief History of AppleScript

AppleScript and its associated tools were conceived, designed, and implemented between 1989 and 1993. It was a long-term investment in fundamental infrastructure that matured over a span of several years (see Figure 1.3).

Figure 1.3

A timeline of the history of AppleScript



Often considered a precursor to and inspiration for AppleScript, HyperCard was released in 1987. This software enabled novice programmers to rapidly create custom tools that would carry out a set of specific processes. Featuring an easy-to-learn, English-like scripting language called HyperTalk, it was easier to learn and use than other programming languages available at that time.



AppleScript was officially conceived in 1989 as a research project by the Advanced Technology Group (ATG) at Apple Computer and was code-named "Family Farm." The research team was led by Larry Tesler and included Mike Farr, Mitchell Gass, Mike Gough, Jed Harris, Al Hoffman, Ruben Kleiman, Edmund Lai, and Frank Ludolph. Their goal was to create a new system-level development environment for the Mac OS that would allow for inter-application communication and control and provide a user-level language. The original group was disbanded in mid-1990 and new teams were assembled to design and implement the ideas first conceived.

The first step was the development of Apple Events, which is the inter-application communication foundation of AppleScript in the OS. Written in Pascal, like much of the Mac OS at the time, this foundation needed to be in place before the development of AppleScript could begin. The AppleScript project officially began in April 1991, just months before Mac OS 7, when the new Apple Events foundation was released.

**NOTE**

The AppleScript project was code named “Gustav” after a team member’s dog.

In September 1992, AppleScript reached beta. However, in January 1993, the original team was disbanded when several leaders left the project. It wasn’t until April of that year that the AppleScript 1.0 Developer’s Toolkit shipped as a stand-alone product that could be installed on any Mac running System 7. In September, AppleScript version 1.1 was included as part of System 7.1.1 (System 7 Pro). In December, the first “end user” release — AppleScript 1.1 Developer’s Toolkit and Scripting Kit — was released. Finally, in 1994, AppleScript was ready to revolutionize how people use computers when it took its place as an official part of Macintosh System 7.5.

Since that time, AppleScript has slowly evolved into the invaluable tool that we know today. In 1997, the Macintosh Finder finally became scriptable, eliminating the need to use the Finder scripting extension. When Macintosh OS 8.0 was released in July 1997, it included AppleScript version 1.1.2 with many minor improvements.

**NOTE**

In 1997, Apple had plans to eliminate AppleScript in order to cut expenses but, thankfully, this plan was thwarted by a campaign by loyal users of the technology.

In October 1998, AppleScript 1.3 was released, recompiled as a native PowerPC extension and included Unicode support. In that year, Steve Jobs demonstrated AppleScript at Seybold, and *Macworld* magazine named AppleScript 1.3 the “Technology of the Year.” In 2006, AppleScript held position #17 on *Macworld*’s list of the 30 most significant Mac products ever.

**NOTE**

Read the entire history of AppleScript at www.cs.utexas.edu/~wcook/Drafts/2006/ashop1.pdf.

A 1999 technology study by research firm GISTICS estimated that AppleScript produced more than \$100 million in annual savings for North American media firms. Today, Google returns more than two million results when searching for the word “AppleScript.”

In Mac OS 10.6, released in 2009, AppleScript, Standard Additions, and all AppleScript-related system applications, such as System Events, are now 64-bit capable.

The technology has flourished and now boasts a thriving and happily efficient user base.

Finding AppleScript Resources

AppleScript is made up of various elements located on each Mac computer. These elements include applications, scripting additions, and components.

Applications

AppleScript developers use two applications: the AppleScript Editor and the Folder Actions Setup application.



NOTE

Mac OS 10.5 included a folder called “AppleScript” inside the `/Applications/` folder that contained three applications: Script Editor, AppleScript Utility, and Folder Action Setup. Mac OS 10.6 doesn’t include this folder. The Script Editor is now the “AppleScript Editor” and is in the `/Utilities/` folder; the options accessible from the AppleScript Utility are now in the Editor’s preference panel; and Folder Action Setup is now in the `/System/Library/CoreServices` folder.



AppleScript Editor

Probably the most important application in the AppleScript toolbox is the AppleScript Editor, which is located in the `/Applications/Utilities/` folder. This application is used to create, write, edit, compile, run, and save scripts. It contains many features that assist a developer in learning the language, writing scripts, and exploring the command library of scriptable third-party applications.



CROSS-REF

See Chapter 6 for more information about using the AppleScript Editor.



Folder Actions Setup

The Folder Actions Setup application, located in `/System/Library/CoreServices/`, is used to assign script actions to folders. This enables a script to respond to various folder actions, such as the arrival or removal of a file or folder, and perform a sequence of automated tasks on it.



NOTE

You can access the Folder Actions Setup application by clicking a folder while pressing the **Ctrl** key or by clicking the right button on your mouse and selecting the Folder Actions Setup option from the contextual menu.

The Folder Actions Setup window, shown in Figure 1.4, lets you enable and disable folder actions globally as well as add, show, and remove folders on a computer. Once you have added a folder, you can attach one or more scripts to it.



CROSS-REF

See Chapter 16 for more information about using Folder Actions.

Figure 1.4

The Folder Actions Setup window



Scripting additions



A *scripting addition* is used to extend the AppleScript language by providing a set of additional commands. Scripting additions can be stored in several locations on a computer. Apple includes several scripting additions in the OS and you can find additional third-party scripting additions on the Internet.



CROSS-REF

See Chapter 16 for more information about installing and using scripting additions.



TIP

A set of sample scripts provided by Apple and installed as part of the Mac OS 10.6 installation is located at `/System/Library/Scripts/`.

Components

Components are files that provide basic functionality for AppleScript, Apple Events, and other OSA-related languages. While the process of using or developing scripts does not require you to be concerned with these components, they are provided in this book for informational purposes only. Except when adding or removing additional language components, such as JavaScript, you should never attempt to remove, modify, or be concerned with the whereabouts of any of these components.

The Apple Event Manager provides an application programming interface (API) for sending and receiving Apple Events, thereby providing support for the creation of scriptable applications.

It exists as part of the `CoreServices.framework` and is called the `AE.framework`. This is important for those creating scriptable applications but not important for those writing scripts with AppleScript.

Likewise, the `OpenScripting.framework` is a part of the `Carbon.framework` and is not something AppleScript users and developers need to worry about. It defines data structures, routines, and resources that support scripting components regardless of the language. It also compiles, executes, loads, and stores scripts.

The `AppleScript.component` file, the default OSA scripting language component provided by Apple, enables a computer to use the AppleScript language. It is located at `/System/Library/Components`.

Other OSA component files, such as the `JavaScript.component`, can be installed in `~/Library/Components` for each user account that will use it. If your computer is connected to an office network, you may need to contact your network administrator before installing additional OSA components.

Understanding the Unique Characteristics of AppleScript

While old-fashioned macro recording utilities were quite useful in their time — they could simulate a series of literal keystrokes and mouse clicks, respectively — it was difficult to use them in a dynamic and practical manner. With AppleScript you can not only automate a sequence of literal actions, but also you can create a dynamic script that includes logical branches, variable content, and options for different behavior depending on specific conditions. This gives AppleScript the power of a real programming language.

AppleScript possesses more unique characteristics that add to its appeal, such as its English-like syntax, the fact that it is universally open-ended, its deep level of access into the Mac OS framework and the frameworks of third-party applications, and its consistency between OS updates.

English-like syntax

One of the most unique characteristics of AppleScript is its English-like syntax. While some detractors might say it is not even close to “natural spoken English,” most would agree that it is certainly more like a spoken language than most other scripting and programming languages. The difference in syntax can be illustrated with a few simple examples.

The following examples present a sort of Rosetta Stone of programming languages. The code in each example performs exactly the same function: It builds a text-based list of numbers within a range specified by two variables. At the end of each script, the resulting value will be a sequence of numbers from 25 to 30 with a carriage return after each number.