Mark Conway Munro

# AppleScript®


www.**wileydevreference**.com

# AppleScript®

# AppleScript®

Mark Conway Munro


Wiley Publishing, Inc.

**AppleScript®**

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



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



Library of Congress Control Number: 2010925705

*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 Lefevere

**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

# Contents

# 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

# 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.

**In This Chapter**

**Figure 1.1**

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

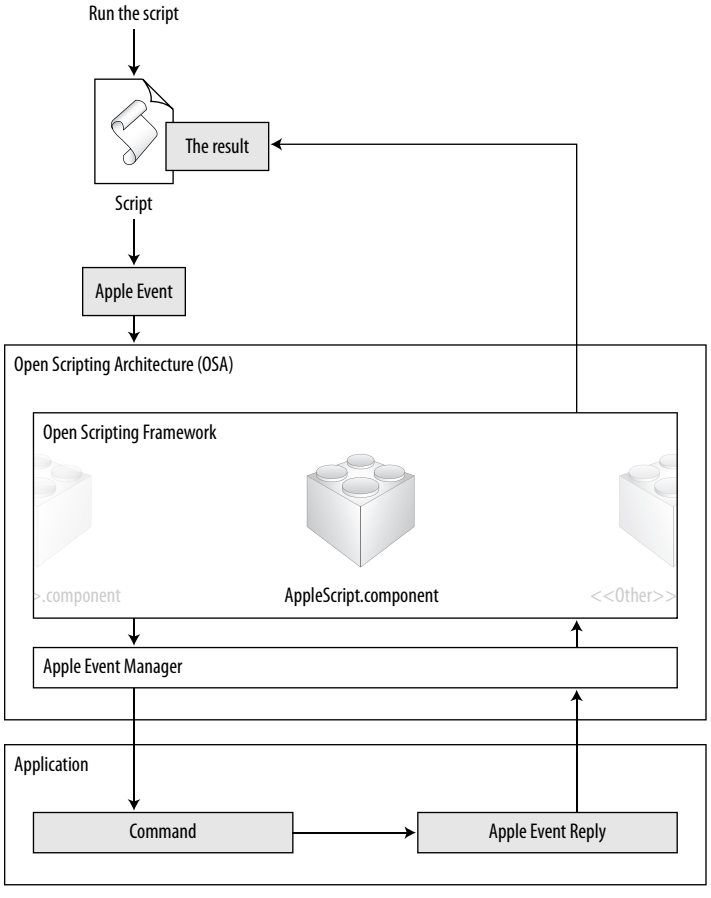
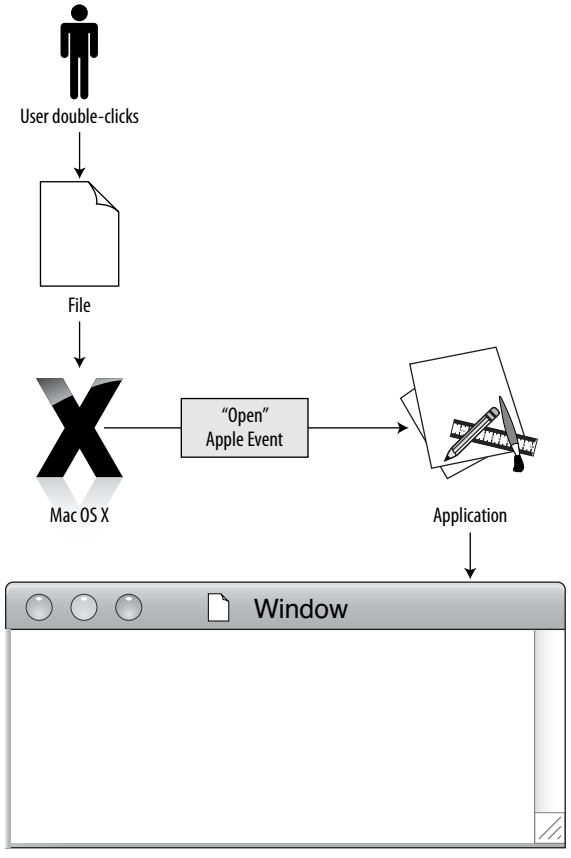


**Figure 1.2**

The underpinnings of an Apple Event opening a document

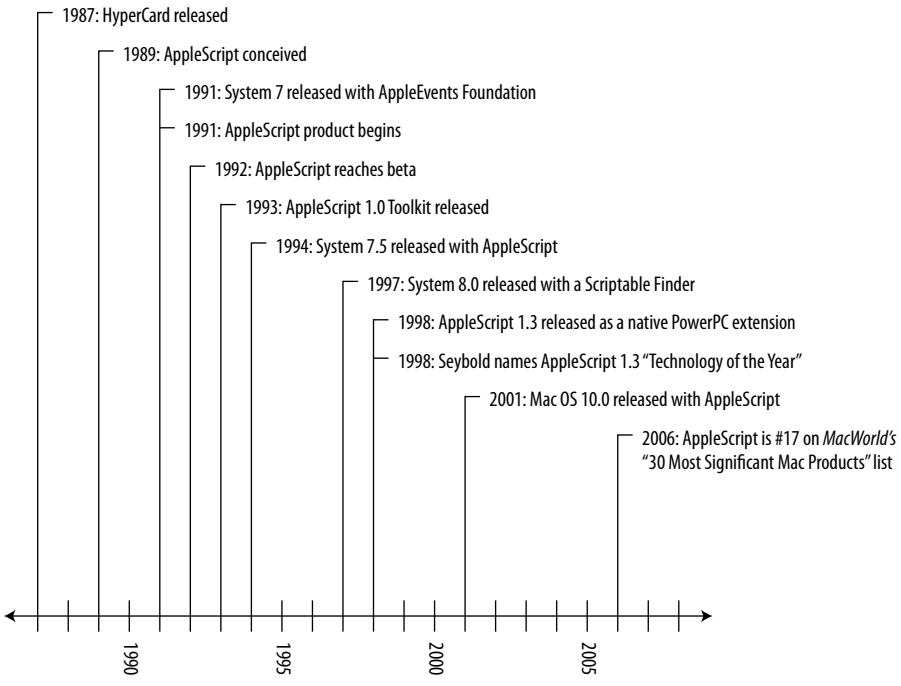



# A Brief History of AppleScript

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/ashopl.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.