

*Effective* SOFTWARE DEVELOPMENT SERIES

Scott Meyers, Consulting Editor



# *Effective* OBJECTIVE-C 2.0

*52 Specific Ways to Improve Your iOS  
and OS X Programs*

Matt Galloway

## **Effective Objective-C 2.0**

---

# Effective Objective-C 2.0: 52 Specific Ways to Improve Your iOS and OS X Programs

## Table of Contents

Contents

Preface

Acknowledgments

About the Author

Chapter 1: Accustoming Yourself to Objective-C

Item 1: Familiarize Yourself with Objective-C's Roots

Item 2: Minimize Importing Headers in Headers

Item 3: Prefer Literal Syntax over the Equivalent Methods

Item 4: Prefer Typed Constants to Preprocessor #define

Item 5: Use Enumerations for States, Options, and Status  
Codes

Chapter 2: Objects, Messaging, and the Runtime

Item 6: Understand Properties

Item 7: Access Instance Variables Primarily Directly When  
Accessing Them Internally

Item 8: Understand Object Equality

Item 9: Use the Class Cluster Pattern to Hide Implementation  
Detail

Item 10: Use Associated Objects to Attach Custom Data to  
Existing Classes

# **Table of Contents**

Item 11: Understand the Role of objc\_msgSend

Item 12: Understand Message Forwarding

Item 13: Consider Method Swizzling to Debug Opaque  
Methods

Item 14: Understand What a Class Object Is

## **Chapter 3: Interface and API Design**

Item 15: Use Prefix Names to Avoid Namespace Clashes

Item 16: Have a Designated\_INITIALIZER

Item 17: Implement the description Method

Item 18: Prefer Immutable Objects

Item 19: Use Clear and Consistent Naming

Item 20: Prefix Private Method Names

Item 21: Understand the Objective-C Error Model

Item 22: Understand the NSCopying Protocol

## **Chapter 4: Protocols and Categories**

Item 23: Use Delegate and Data Source Protocols for  
Interobject Communication

Item 24: Use Categories to Break Class Implementations  
into Manageable Segments

Item 25: Always Prefix Category Names on Third-Party  
Classes

Item 26: Avoid Properties in Categories

Item 27: Use the Class-Continuation Category to Hide  
Implementation Detail

Item 28: Use a Protocol to Provide Anonymous Objects

## **Chapter 5: Memory Management**

# **Table of Contents**

Item 29: Understand Reference Counting

Item 30: Use ARC to Make Reference Counting Easier

Item 31: Release References and Clean Up Observation  
State Only in dealloc

Item 32: Beware of Memory Management with Exception-Safe  
Code

Item 33: Use Weak References to Avoid Retain Cycles

Item 34: Use Autorelease Pool Blocks to Reduce  
High-Memory Waterline

Item 35: Use Zombies to Help Debug Memory- Management  
Problems

Item 36: Avoid Using retainCount

## **Chapter 6: Blocks and Grand Central Dispatch**

Item 37: Understand Blocks

Item 38: Create typedefs for Common Block Types

Item 39: Use Handler Blocks to Reduce Code Separation

Item 40: Avoid Retain Cycles Introduced by Blocks  
Referencing the Object Owning Them

Item 41: Prefer Dispatch Queues to Locks for Synchronization

Item 42: Prefer GCD to performSelector and Friends

Item 43: Know When to Use GCD and When to Use Operation  
Queues

Item 44: Use Dispatch Groups to Take Advantage of  
Platform Scaling

Item 45: Use dispatch\_once for Thread-Safe Single-Time  
Code Execution

Item 46: Avoid dispatch\_get\_current\_queue

# **Table of Contents**

## Chapter 7: The System Frameworks

Item 47: Familiarize Yourself with the System Frameworks

Item 48: Prefer Block Enumeration to for Loops

Item 49: Use Toll-Free Bridging for Collections with  
Custom Memory-Management Semantics

Item 50: Use NSCache Instead of NSDictionary for Caches

Item 51: Keep initialize and load Implementations Lean

Item 52: Remember that NSTimer Retains Its Target

## Index