Extracted from:

Swift Style

An Opinionated Guide to an Opinionated Language

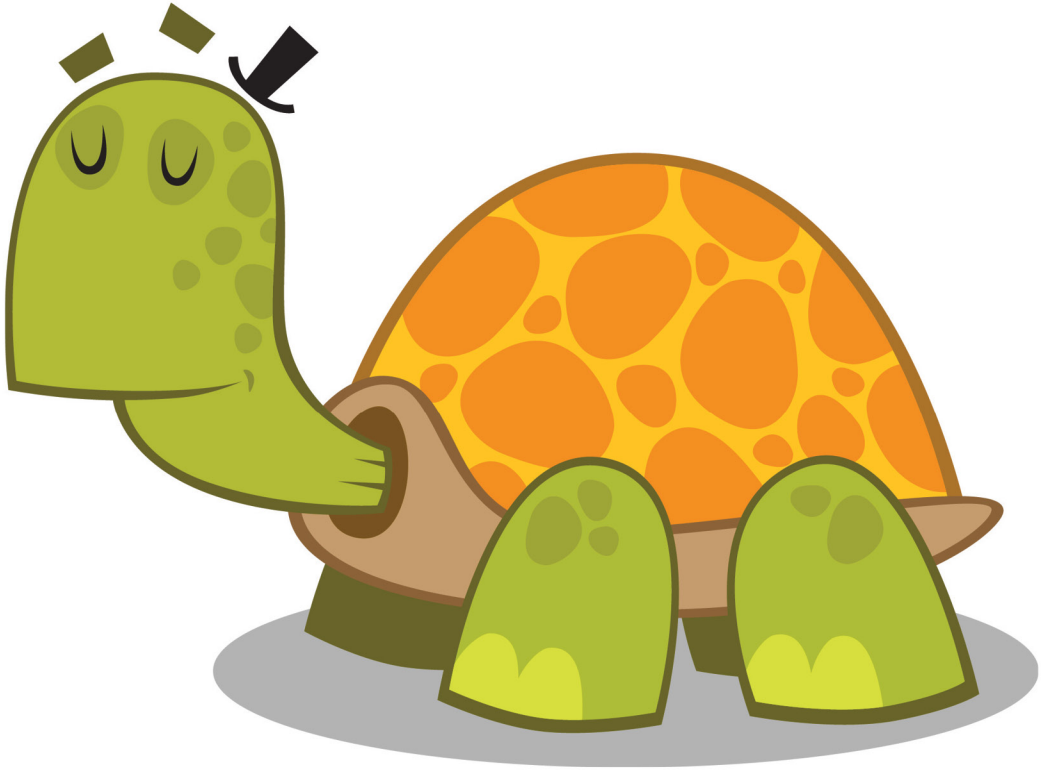
The Pragmatic Bookshelf

Raleigh, North Carolina

The Pragmatic Programmers

Swift Style

An Opinionated Guide to an Opinionated Language



Erica Sadun

edited by Brian MacDonald

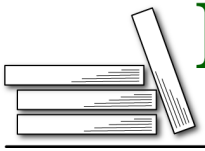
Swift Style

An Opinionated Guide to an Opinionated Language

Erica Sadun

The Pragmatic Bookshelf

Raleigh, North Carolina



Pragmatic Bookshelf

Our Pragmatic books, screencasts, and audio books can help you and your team create better software and have more fun. Visit us at *https://pragprog.com*.

The team that produced this book includes:

Publisher: Andy Hunt
VP of Operations: Janet Furlow
Executive Editor: Susannah Davidson Pfalzer
Development Editor: Brian MacDonald
Indexing: Potomac Indexing, LLC
Copy Editor: Linda Recktenwald
Layout: Gilson Graphics

For sales, volume licensing, and support, please contact *support@pragprog.com*.

For international rights, please contact *rights@pragprog.com*.

Printed in the United States of America.
ISBN-13: 978-1-68050-235-0
Encoded using the finest acid-free high-entropy binary digits.
Book version: P1.0—March 2017

Welcome to *Swift Style*

This book offers a practical, powerful, and opinionated guide to coding style. It incorporates a multitude of best practices, guiding you to work successfully in this equally opinionated programming language.

How This Book Got Here

This book didn't start out as a book. It began its life as a blog post ("Swift: Don't do that"[1]) and then as a GitHub repository.[2] At the time, I was inspired by one of Natasha Murashev's links[3] about the new open source SwiftLint[4] project, developed by JP Simard[5] and the guys at Realm.[6] Murashev writes a weekly curated Swift newsletter that covers news and articles about the language.

At that time, SwiftLint was built on what can be kindly described as extreme hacking by hooking into Swift's then-private SourceKit services. As I wrote at the time,[7] I loved the idea behind SwiftLint but I needed a better tool for immediate use. I wanted something good enough for real code, even in imperfect form. It had to run from the command line and work with playgrounds. It had to be able to scan files added to Xcode projects by reference as well as those physically stored in the project root. Leveraging some of the code I had developed for a much earlier `Xcode Manifest` application, I built an Objective C scanner[8] with simple matching rules based on regular expressions.

My linter worked line by line instead of parsing because I didn't want to use inter-process hacking. (Its implementation relied on regular expression

1. http://ericasadun.com/2015/05/05/swift-dont-do-that/
2. https://github.com/erica/testlint
3. https://swiftnews.curated.co/search?q=SwiftLint
4. https://github.com/realm/SwiftLint
5. https://github.com/jpsim
6. https://github.com/realm
7. http://ericasadun.com/2015/05/18/swift-alternative-lintage/
8. https://github.com/erica/testlint

matching, waving my hands, and sacrificing chicken entrails.) It wasn't a great linter but it was an amazing way to start thinking about the way I do and should write Swift. Over time, my rule base grew and grew as my interest evolved from "how do I automate simple code checks?" to "what are the right things to check in my code?"

I kept working on my repo to incorporate rules that I found valuable. The project quickly evolved from a project about linting into a way I could *explore* style rules for this new language. As I worked on my style preferences, I found myself talking to Swift developers, asking for opinions and feedback, and incorporating their experiences into my writing.

My style sheet kept growing, maturing from a few pages of directives to a pamphlet to a small book. At a certain point, I realized I had something of more general utility than a personal style guide on my hands. And that was when I approached Pragmatic about the possibility of transforming this material into a proper book: the book you are now reading.

From the time I started work on this project (way back in Swift 1!) to today, Swift settled down. It decided exactly what kind of language it was going to be and has finally realized and expressed that language in a much more stable form. A significant developer base has adopted the language and now has several years of Swift development experience behind them.

Today's Swift opens itself to evaluation. It allows you to weigh preferred coding styles even though the community as a whole has not yet adopted fixed conventions. You can finally build house guidelines without having the rug pulled out from under you every few months. Swift has *arrived*.

This book is written for Swift coders both new and experienced. The material in these pages ranges from "Captain Obvious" good practices to power strategies, spanning the full range between those points. I solicited opinions and preferences along the way, taking into account varied experience, programming domains, and conventions, to build a strong base of practical and practicing advice.

If there's one core message I want to pass along, it's this: there are always ways to enhance your code and coding practices. While code can have issues, those issues should be evaluated with respect to strong design principles, not odor. Smell the roses, not the code, and embrace your personal style.

Your Code Doesn't Smell

Your style may not match my style. Your best practices may not be mine. Every coder and development group have their own way of doing things. A code base evolves a dialect drawn from the background, experiences, and requirements of its development participants.

Code is more often read than written. Its crafting involves a process of review and clarification apart from developing algorithms and ensuring correctness. You who are writing code, the future you who reads that code, your extended team, and anyone who visits and maintains your repositories are all first-class consumers of the code you write today. Code that's readable, consistent, and comprehensible lowers costs in terms of life span, maintainability, and error prevention.

Style is not about code correctness. Code may be poorly implemented or incorrect or fragile. That's why testing, debugging, and review play such critical roles in development. What style does is create a consistent experience that enables a reader to abstract away a file's line-by-line layout and focus on the underlying meaning, intent, and implementation.

I have my own set of rules. They're ever changing, flexible, and adaptable. They, like yours, depend on circumstance and utility. Any style that produces readable, maintainable, and well-documented code is, by definition, successful.

The Swift community has begun to settle on certain standards, not all in agreement, that create *conventional* coding styles. I've been collecting and distilling these conventions: first to create a personal style for myself and then later curating them into this book.

This book doesn't offer canonical answers. It incorporates many conventional answers, which you're welcome to adopt, ignore, ridicule, or embrace. It explores the areas of Swift where structure comes into play. Whether you're developing personal style or house style, there are always ways to enhance your code choices.

Style guides should be firm, definitive, and prescriptive. This book isn't *the* ultimate Swift style guide and it can't be. I'm writing this outside of the specifics that propel the needs of a particular project or mission. Instead, it's a book about incorporating good Swift style and conventions into your daily programming practices to create your *own* house guidelines and preferred practices.

This book lays out ideas and principles for you to draw from and adopt. Use these topics to establish and enhance house rules and then start living by the rules you created. The results will be cleaner, consistent, and readable code. Style, no matter how individual or corporate, how quirky or conventional, how personalized or typical, always creates an improved coding, reading, and maintenance experience.

What's in This Book

Apple's Swift programming language has finally reached a level of stability that opens its use to a much wider audience. *Swift Style* guides you through the ins and outs of some Swift programming best practices. This book is written for Swift programmers both experienced and new to this language who want to explore the art of crafting code in this language.

Code style matters. Critical dos and don'ts for writing readable Swift code guide you to better craftsmanship. This book explores common coding challenges and the best practices that address them. From spacing, bracing, and semicolons to proper API style, discover both *what* you should do and the *whys* behind each recommendation.

A style guide establishes a consistent experience of well-crafted code that lets you focus on the code's underlying meaning, intent, and implementation. This book explores the areas of Swift where coding structure comes into play. Whether you're developing a personal style or a house style, there are always ways to enhance your code choices. You'll find here the ideas and principles to establish or evolve your own best style practices.

Begin with simple syntactical styling. Strengthen code bracing for easy readability. Style your closures for safety and resilience. Perfect your spacing and layout. Master literal initialization and typing. Optimize control flow layout and improve conditional style choices. Transition from Objective-C and move code into Swift the right way. Improve API design using proper naming and labeling. Elevate defaulted arguments and variadics to their right places. *Swift Style* covers it all.

Here's a chapter-by-chapter breakdown of what you'll find in this book:

- Chapter 1, *Structure Your Code for Readability*, on page ? Swift code faces wrapping challenges you don't encounter in C-like languages. Swift's top-heavy style of output magnifies the importance of braces, wrapping, and line composition. This chapter explores ways you choose to build coding backbones. You'll learn to enhance code readability and emphasize code meaning and design intent.

- Chapter 2, *Adopt Conventional Styling*, on page ? Conventional style is like a window. It allows you to look through to see the code intent or landscape that lies beyond it. When there's a streak on the window or an unconventional use in code, programmers naturally fixate on the wrong thing. This chapter explores common house style elements. It surveys Swift language features you'll want to standardize and lock down for better and more consistent code.
- Chapter 3, *Establish Preferred Practices*, on page ? Like any language, there's always more than one way to achieve your goals in Swift. Making good choices is a critical part of coding style. This chapter surveys common design decision points and guides you through refactoring opportunities. Step beyond simple linting to explore the architectural design points that affect your development.
- Chapter 4, *Design the Right APIs*, on page ? An API establishes a contract for calling code. It describes the types, methods, and results produced by an implementation and how these will behave. A well-designed API provides a clear and understandable set of tools, with well-chosen names and thoughtful consideration to resilience and long-term code evolution. Learn the "Swifty" ways to design your APIs by leveraging the principles of clarity, concision, and utility. From access control to naming, from nesting to defaults, this chapter shows you how to present your functionality safely and meaningfully.
- Chapter 5, *Look to the Past and the Future*, on page ? Time plays a key role in Swift style. Seeing Swift in this larger scope is important in both adopting new practices and supporting code beyond its moment of creation. This short chapter explores time, guiding you into routines that enhance, document, and support the Swift code you're now writing. You'll learn ways you can best move on from your code's past and how you will support its future
- Chapter 6, *Good Code*, on page ? This book wraps up with a short meditation about what good code *means* and how you can recognize what good coding *is*.

Contributing to This Book

When reading through this volume, if you see something that's missing or that's wrong, or if you just want to suggest an alternative viewpoint, drop me a note at erica@ericasadun.com and I will consider it for a future update. Thank you in advance for being part of this effort.

Online Resources

This book has its own web page,[9] where you can find more information about the book and interact in the following ways:

- Participate in a discussion forum with other readers, Swift enthusiasts, and me
- Help improve the book by reporting errata, including content suggestions and typos

Other valuable online resources include these:

- *The Swift Programming Language* is a free ebook offered by Apple on the iBooks store and on its website. The documentation can usually be found somewhere around here,[10] but I warn you Apple often moves its resources, so this URL can and will change.
- A second Apple ebook, *Using Swift with Cocoa and Objective-C*, is just as essential. It provides an overview of the topics related to interoperability between the two languages and the details of API calls. It's a much shorter volume than *The Swift Programming Language* because of its tight focus.
- Apple's online *Swift Standard Library Reference*[11] provides an indispensable overview of Swift's base functionality layer. It offers an overview of fundamental data types, common data structures, functions, methods, and protocols. If you stop learning Swift at the language basics, you'll be missing out on this critical portion of core language expressiveness.
- *SwiftDoc*[12] offers auto-generated documentation of Swift's standard library. This is the same documentation you find when you Command-click symbols in Xcode but presented in easier-to-read web pages. It's a terrific resource.
- Also amazing is *ASCII WWDC*.[13] Enter keywords and search through years of WWDC talks. This website helps you track down presentations specific to the Swift language and the Xcode tools that support Swift development.

9. https://pragprog.com/book/esswift/swift-style
10. https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/Initialization.html
11. https://developer.apple.com/library/ios/documentation/General/Reference/SwiftStandardLibraryReference/index.html
12. http://swiftdoc.org
13. http://asciiwwdc.com

- The two Apple *Swift*[14] *blogs*[15] are updated about once a month, but coverage includes can't-miss topics related to language features and case studies. Their mission statements speak about behind-the-scenes peeks into language design, but their focus over time has been more practical how-to articles. The blogs' resources page[16] further provides links to a set of essential Swift and Xcode materials, including iTunes U courses, videos, sample code, and a link to the official Swift Standard Library Reference.
- I post regular updates on Twitter[17] and at my personal website[18] about the current state of Swift.
- Natasha Murashev offers a superb curated weekly *This Week in Swift* newsletter[19] with lots of great articles for coders.
- Kenny Leung's weekly *TWISt-shout* newsletter[20] wraps up each week of Swift language development news.
- The redesigned *Apple Developer Forums*[21] provide access to Swift engineers, lively discussions, and up-to-date how-to information. You'll also find important language information archived at the old forums site[22]; navigate to Developer Tools > Language > Swift.
- Apple's *Swift Users* mailing list[23] offers help with Swift and its related tools. The conversation, at least at this time, tends to be thoughtful and the traffic is light, with good access to high-level Swift developers. As with any email list, this one is meant to discuss the language with other users and is not for debugging support.
- Reddit's *Swift sub*[24] offers a community-curated list of interesting topics and peer support. You'll also find excellent community connections at ios-developers.slack.com's #swift channel and swift-lang.slack.com. (Visit http://swift-lang.schwa.io for invitations to the latter.)

14. https://developer.apple.com/swift/blog/
15. http://swift.org/blog
16. https://developer.apple.com/swift/resources/
17. http://twitter.com/ericasadun
18. http://ericasadun.com
19. https://swiftnews.curated.co
20. https://github.com/pepperdog/TWISt-shout
21. https://forums.developer.apple.com/community/xcode/swift
22. https://devforums.apple.com/index.jspa
23. https://swift.org/community/#mailing-lists
24. http://reddit.com/r/swift/

- IBM's Linux-based *Swift Sandbox*[25] provides a live way to paste, test, and share code when discussing Swift with other developers. As a non-Apple website, it has some minor nontrivial differences in APIs and does not provide you with access to Cocoa and Cocoa Touch.

Ready to Get Going?

You're about to take a deep dive into Swift style. This book explores the fine details of language use in a fascinatingly pedantic (and possibly diagnostically obsessive) fashion. This book won't change your understanding of Swift and it won't reduce the pain of writing and debugging your code, but it should take a lot of the pain out of reading and reviewing that code at some later date. This book offers new ways to think about Swift, to structure Swift, and to make good coding choices.

Thank you for purchasing this book. Let's get started!

Credits

Thanks

My thanks to everyone who helped provide technical information and feedback during the development of this book. I'm deeply indebted to my technical review team: Kevin Ballard(!), Tony Gray, Maurice Kelly, Stephen Wolff, Stefan Turalski, Peter Wood, Nick Watts, Alex Henry, Arun Kumar, Daivid Morgan, Derek Graham, Dianne Siebold, Gábor László Hajba, Kenneth Geisshirt, Nick McGinness, Peter Hampton, and Phillip Apley. Additional shout-outs go to Steve Streza, Mike Ash, Alex Kempgen, Seth Willets, Gwynne Raskind, Xiaodi Wu, Kenny Leung, Paul Cantrell, Sebastian Celis, Jeremy Tregunna, Nate Cook, Pyry Jahkola, Russ Bishop, Jacob Bandes-Storch, Soroush Khanlou, Tim Vermeulen, Anthony Mattox, Alfonso Urdaneta, Brent Royal-Gordan, Zev Eisenberg, Cezary Wojcik, Seivan Heidari, Jessy and Catie Catterwaul, Dave DeLong, Rainer Brockerhoff, Peter Livesey, Joe Fabisevich, Mark Rowe,

25. https://swiftlang.ng.bluemix.net

Olivier Halligon, Jeff Forbes, and Greg Titus. Thanks go out to the Swift Evolution and Swift Users' Group members and everyone at Apple who has provided inspiration, support, and insight, including but not limited to Chris Lattner, Joe Groff, Ted Kremenek, John McCall, David Smith, Doug Gregor, Dave Abrahams, Jack Lawrence, Brian Gesiak, and Jordan Rose.

Thanks also to my Pragmatic Team: Brian MacDonald, my development editor who managed and oversaw with patience, insight, and humor; Susannah Davidson Pfalzer, executive editor; Andy Hunt, publisher; and everyone else at The Pragmatic Bookshelf who helped make this book a reality.