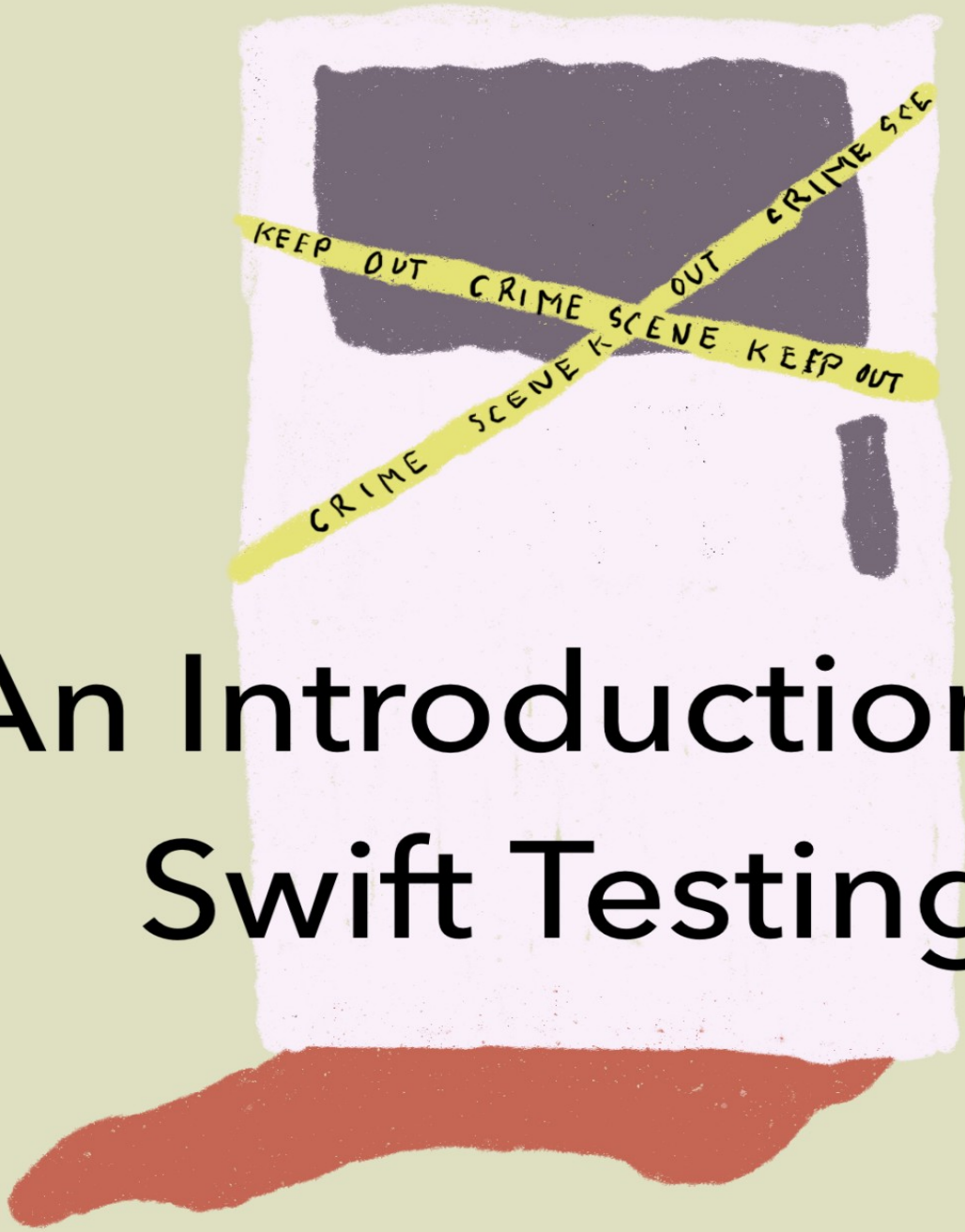
The Case of the Crimson Test Suite



An Introduction to Swift Testing

Daniel H Steinberg

Copyright

ISBN-13: 978-1-944994-07-5

Book Version

This is version 1.0 for Swift 6, Xcode 16 beta 6 or later, macOS Sonoma/Sequoia, and iOS 18 released August 2024.

Code Download

Visit https://github.com/editorscut/ec016swifttesting for all of the code for this book.

Run it in Xcode 16 or higher. All code is written in Swift.

Recommended Settings

The ePub is best viewed in scrolling mode using the original fonts. On smaller devices I also choose landscape. For some reason that I don't understand, scrolling mode is supported by Apple's Books app on the iPad but not on the Mac. If you view this book in Apple's Books app, choose "Let lines break naturally".

Submit Errata

Submit your errata here for the book or for the source code by selecting New Issue. Please provide the book version listed above, chapter, section, and page number in your issue so that I can find it and, if possible, resolve it quickly.

Official Links

Please check http://developer.apple.com for additional resources including videos, sample code, documentation, and forums. You'll also find information on what is required to take advantage of these resources.

Apple has posted videos, slides, and sample code from the Worldwide Developers Conference.

Legal

CHAPTER 1

Getting Started

There's an old joke that begins with the question, "how can you tell that someone is a vegan?"

The answer is, "Don't worry, they'll tell you."

My wife and I would try going vegan about once a year.

After a while she'd say, "what about vegetarian."

So we'd do that for a while.

Then she'd have a taste for a hamburger and that would be that.

Except it wasn't. We weren't full on vegans, we weren't even vegetarians, but we were eating a lot less meat and we felt better for it.

We have an analogous situation in the world of people who write code for Apple platforms.

Someone who religiously tests their code is like the vegan of iOS development.

100% test coverage? Not likely. Besides, is that even good for you?

What about Test Driven Development?

I don't know. Now and then I just want to write some code. I'll test it later.

"Oh no," you say, "I hope this isn't a book that's going to lecture me on writing tests. I may as well buy a book on why I should floss my teeth every day."

No. No lectures here. In fact, I understand.

If you've tried to write unit tests for iOS or macOS apps before, you know how painful it was.

The XCTest framework was an old system based on an older system that was modeled on an even older system.

Swift Testing is so much nicer and easier to use.

It's a modern testing framework that makes it easy to write a test.

This book is an invitation to give it a try. We'll start with a small test here or there.

As each test passes, we feel pretty good and write another. Before we know it, we've written enough tests to gather them into a suite.

Oh look, these tests belong together but they're not in the same suite.

That's ok - add a tag for them and you can run any combination you want.

But I'm getting ahead of myself. I'm clearly excited about Swift Testing and this short book is a quick, introductory dive into what is possible.

In this chapter we look at examples of creating a project or Swift package along with a testing target for Swift Testing. We also look at examples of adding a testing target to an existing project or Swift package.

Also, this is the fourth book in the series that features stories of the great detective Chamfered Edges and sidekick, Captain Swiftly. If you have not met them yet, you may want to read this introduction from "The Curious Case of the Async Cafe".

In this book we jump back in time to learn of how Edges was introduced to the world of detection while apprenticing to the Agile Detective.

We begin our journey with the revelation that Edges had once apprenticed to the Agile Detective.

The Agile Detective

Edges and I had been working together for half a year. I'd been shadowing them since our investigation of the "Case of the Vanishing Bodies" trying to learn how to be a detective.

"Edges," I asked the great detective, "how do you know when a case is completed?"

"Well, mon ami," said Edges, sipping their espresso, "sometimes it is obvious and sometimes, as with the case we wrapped up this morning, it is subtle."

I looked across our usual table at the Async Cafe. That answer was both true and completely unhelpful. I raised an eyebrow.

Edges nodded and continued, "in either case, I always have a criteria for when a case will be complete."

I added, "I've observed that you have criteria of every step along the way."

"Swiftly," said Edges, "it is the only way to track our progress. It's something I learned from my mentor, The Agile Detective."

"The Agile Detective?" I repeated. "Was that their name?"

"No, of course not. His real name was James and that's what we called him. But we always referred to him as the Agile Detective."

"Was he particularly good at getting into small places or crawling through small openings?"

"No," laughed Edges, "not that sort of agile. He was actually a very large and physically inflexible man. But his mind. That's where we found his agility. And his methodologies - they were informed by some of the better known practices of agile programmers."

I paused a moment, noticing a stranger who entered the cafe and was looking around. As the stranger headed our way, I said to Edges, "I'd very much like to hear some of these stories."

"I can do better than that," my friend said agreeably, "I think you should consider spending time working with him to learn some of his methodologies."

"Does he live far?" I asked.

"Non, just a short bus ride away," said Edges. "It will take me a few days to set things up."

I suppose this is good news as it should give you enough time to read the rest of this chapter to learn how to set up our environment for the remainder of this short book on Swift Testing.

In the next section, you will learn to create a new project with Xcode that includes a testing target.

Note that Swift Testing requires Swift 6 and so you will need Xcode 16 or above.

This is all you will need to participate in the remainder of the book. However, there will be times that you need to add a testing target to an existing project or you wish to test a package. You can skip those sections for now if you want and they're here if you need them in your own work later on.

Road Map

Here's an overview of the contents of the chapters in this book.

Chapter One: Getting Started

In this first chapter we looked at various ways to set up Swift Testing. We created a project that included a testing target and added a testing target to an existing application. We repeated the process for Swift Packages.

Chapter Two: The Basics of Swift Testing

The running example for this chapter is a Swift Package. We build out the functionality using Swift Testing tests which we initially mostly run from the command line and later run from Xcode.

A test consists of a method with the attached macro `@Test`. A test can have one or more expectations which are constructed using the freestanding macro `#expect()`. We'll run more than one expectation in a single test when its appropriate.

We take advantage of the `@Test` macro to provide nicer display names for tests and to assign tags so that we can identify tests that

conceptually fall under a heading. This allows us to run a subset of the tests by running those with a specific tag. A test can have multiple tags so it can be considered to belong to more than one collection of tests.

At the end we take our well tested code and easily connect it to a UI for converting Celsius to Fahrenheit.

Chapter Three: Test Organization and Flow

In this chapter we start with the skeleton of an RPN Calculator app. The code I provide includes all of the buttons and the display for the calculator but none of the buttons do anything yet. They won't until well into Chapter Four.

We work on test driving the stack that numbers are pushed on to and popped off of.

Along the way we test methods that can throw an error using `#expect(throws:)` and learn to restrict the flow of one expectation into another by using `#require()` instead of `#expect()`. Unsurprisingly, `#require()` also has a variant `#require(throws:)`.

The remainder of this chapter considers different ways in which we can organize our tests. We look at `@Suite` and bundling tests together into a struct and sharing test data. Finally, we see how tags and suites work well together.

Chapter Four: The Finer Points of Swift Testing

This final chapter is a collection of techniques for improving our test suites.

By default Swift Testing tests run in parallel in random order. This is what we usually want.

In rare cases, however, we want to run our tests in a specific order. We look at how to do that and the problems that may arise.

We then round out the functionality of our calculator and call out issues that can arise with any testing framework.

Then we eliminate repeated tests that differ only in the output that is passed to them with a fun feature of Swift Testing called parameterized tests.

Finally, sometimes a type is too verbose when it appears in the Tests Navigator or elsewhere, so we look at using `testDescription` to customize this output.

Enough with the roadmap. Let's get going.