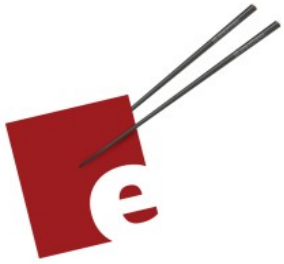


First Edition

# A SwiftUI Kickstart

DANIEL H STEINBERG

Introducing the
SwiftUI User Interface Framework

Editors Cut

# A SwiftUI Kickstart

## Introducing The SwiftUI User Interface Framework

by Daniel H Steinberg

Editors Cut

# Copyright



ISBN-13: 9 978-1-944994-00-6

# Book Version

This is version 0.6 for Swift 5.3, Xcode 12, and iOS 14 released October 10, 2020.

# Recommended Settings

The ePub is best viewed in scrolling mode using the original fonts. On smaller devices I also choose landscape. If you view this book in Apple's Books app, choose "Let lines break naturally".

# Legal

Every precaution was taken in the preparation of this book. The publisher and author assume no responsibility for errors and omissions, or for damages resulting from the use of the information contained herein and in the accompanying code downloads.

The sample code is intended to be used to illustrate points made in the text. It is not intended to be used in production code.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks or service marks. Where those designations appear in this book, and Dim Sum Thinking, Inc. was aware of the trademark claim, the designations have been printed with initial capital letters or in all capitals.

This book uses terms that are registered trademarks of Apple Inc. for which the terms of use don't permit rendering them in all caps or initial caps. You can view a complete list of the trademarks and registered trademarks of Apple Inc. at http://www.apple.com/legal/trademark/appletmlist.html.

The Editor's Cut name and logo are registered trademarks of Dim Sum Thinking, Inc.

CHAPTER 1

# Let's Get Started

Sections:
- Hello, World!
- Previews
- Content View
- Links and Credits
- Version History
- Road Map

Welcome to this introductory book about SwiftUI.

This short book will introduce you to the basic concepts and techniques of SwiftUI along with some of the new features of Swift that were created to enable this technology.

I assume that you are an experienced developer who know Swift but is new to SwiftUI. In fact, this book has been updated to take advantage of the very latest versions of Swift and SwiftUI running on the current versions of Xcode and macOS.

This isn't a comprehensive book covering every nook and cranny of SwiftUI. This is a kickstart. It's designed to get you up and coding and I will try not to get lost in the weeds. That said, there's a lot of

material in this short book. By the end, you'll know enough to go to the docs, forums, and other resources to learn the finer points.

Apple introduced SwiftUI in June 2019 with the first release scheduled for September. Within hours of the first WWDC session you could find videos and blog posts everywhere about the topic. Within days there were several books and entire video courses available online. In the month that followed, I presented tutorials and conference sessions in Portugal, Spain, England, and the United States. So perhaps this book is too late. Perhaps you know everything.

On the other hand, the technology is new enough that we don't really have a sense of the best ways to use it. The APIs are still evolving. Apple is trying not to break anything but in the 2020 release we got new components that support grids and more extensive text input. There are some additions to the way we work with data flow as well.

I decided that as with Swift, I would just present my understanding of best practices and key ideas at this point and as things change I will continue to update the book.

To follow along with the examples, download the code for this book from https://github.com/editorscut/ec009-SwiftUIKickstart. You'll need to have access to a Mac and Xcode 12. You can run Catalina or Big Sur but you'll find the experience much nicer if you're able to use Big Sur. Xcode 12 is available for free from the App Store.

I've added support in this edition for light and night modes in Apple Books if you are reading the ePub. I find the code reads best in Books on an iPad in landscape with scrolling turned on. This was the code stays together and there are no awkward page breaks. In addition, readers asked that I support PDF and Mobi and so I have. I don't think either of these is ideally suited to reading technical books but you have the option if you want or need it.

So where do we start?

Sadly, our first app doesn't do anything. At least not in this chapter.

We begin our hero's journey, as all such stories begin: our hero (that would be you), is going about their business in their ordinary world unaware that everything is about to change.

In the past edition, I began with a simple UIKit example to use as a common starting point. Now that people are coming to SwiftUI as their first experience in coding for Apple platforms, we don't share this background anymore. If you are one of the people who come from UIKit, you can find the original first chapter and examples archived here.

You don't need it to embark on our new adventure. We'll start this chapter by creating a new project and doing a little visual programming just to get the lay of the land before we start writing code.

I can't keep you from skipping ahead to the next chapter where we start writing SwiftUI code, but I encourage you to relax and enjoy

this journey.

After all, you're the hero.

# Hello, World!

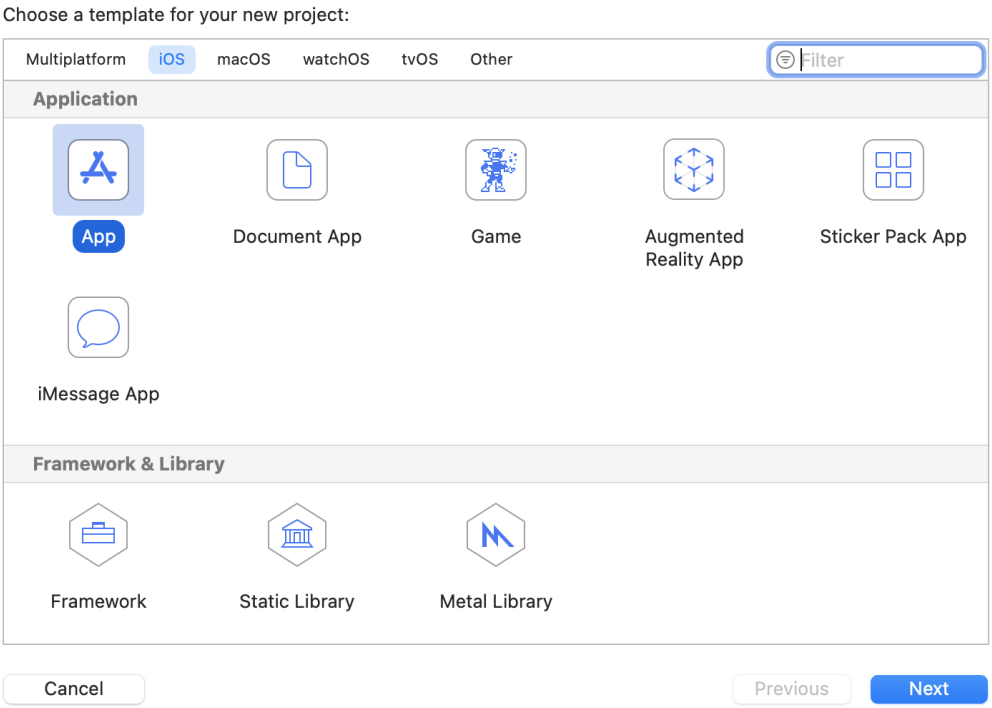
We start with the classic, "Hello, world!" app.

As you'll see, there's almost nothing to do, but it gives us an excuse to look around and see what Xcode creates for us.

## Our first project

In Xcode choose File > New Project menu item or use the keyboard combination Shift - Command - N.

From the template chooser, select the iOS tab and under Application choose App.

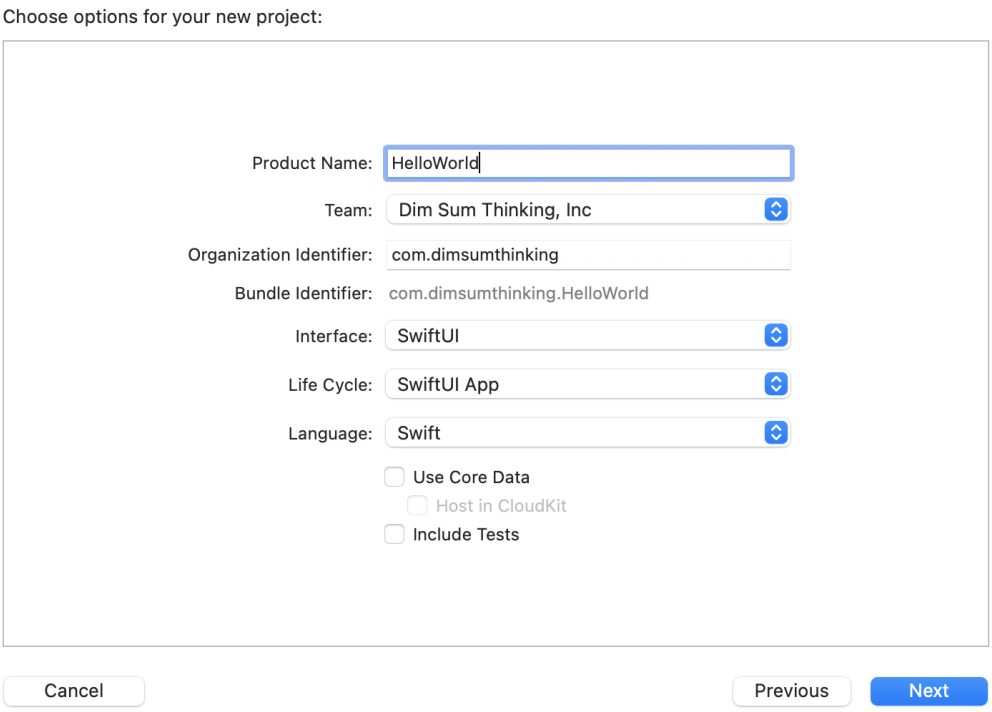



Click the Next button.

Name the product, *HelloWorld* and select your team and enter your Organization Identifier as a reverse DNS. I use com.dimsumthinking.

Here's the important part.

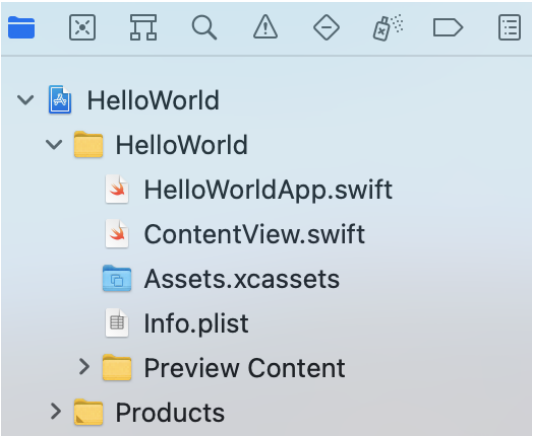
Make sure your Interface is set to SwiftUI, your Life Cycle is set to SwiftUI App, and your Language is Swift.




Click the Next button and choose a location to save your project. It doesn't matter where you choose so long as you remember where it is. Maybe put it on the *Desktop* for now.

## File navigation

You should see a file hierarchy on the left side of your Xcode window. If you don't, use the keyboard combination Command - 0 to toggle the Navigators. You can always type Command - 1 to open the Project navigator or use the sidebar affordance you can see at the top left below along with the navigator tabs.




We'll look at this hierarchy in more depth as the book progresses. For now let's start where your app starts.

## HelloWorldApp

One of the choices we made when we created our app was to set the Life Cycle to SwiftUI App.

Previously, the life cycle used something called an `AppDelegate` with a ton of methods stubbed out to use at various points in the app life cycle.

The current template simplifies things in something that will be named *<YourProjectName>App*. In our current case, the file is *HelloWorldApp.swift*.

Here are its contents. Note that the final version of this code is in the code download in a folder that corresponds to the directory corresponding to the section we are in. Currently we are in section 1 of chapter 1 so look in the folder *01/01*. Inside you'll see our project

folder *HelloWorld*. The current file is *HelloWorldApp.swift*. I'll display the path to the current file above the code listings.

*01/01/HelloWorld/HelloWorld/HelloWorldApp.swift*

```
import SwiftUI

@main
struct HelloWorldApp: App {
  var body: some Scene {
        WindowGroup {
          ContentView()
        }
  }
}
```

There's not much to this code. I'm going to describe it for now in a hand-wavy way. I don't want to get overwhelmed by the code.

- We import `SwiftUI` because that's where `App`, `Scene`, and `WindowGroup` are defined.
- Every app has to begin somewhere. `@main` says that this app begins here. You've seen variations on `main` in every language you've coded in on every platform you've targeted. Swift says, "instead of making you type in a lot of boilerplate or even have to read through it, we'll just tuck it all into `@main`."
- That brings us to the struct named `HelloWorldApp`. I'll make a big fuss about this fact throughout the book, but most of the types we use in SwiftUI are structs not classes.
- `HelloWorldApp` conforms to the `App` protocol. For now, all we need to understand about `App` is that it requires that we provide

a computed property named `body` that returns some sort of `Scene`.

- This particular `Scene` defines a `Window` that contains an instance of `ContentView()`. If we were running an app that supported multiple documents, each one would start off with the same contents. That's what we mean when we say the `WindowGroup` contains its own instance of `ContentView`.

Let's run our app.

## Running HelloWorld

Select a simulator to run the app on. Here I've chosen iPhone 11 Pro. Click to the right of where the interface says HelloWorld in this image to Set your active scheme. Select any iOS simulator you like.


We can run the app by clicking the Play button you see in the screenshot above, with the keyboard combination Command - R, or by selecting the menu item Product > Run.

It will take a while for the app to build, the simulator to launch, the app to be installed on the simulator, and finally for it to start up. When at long last it does, you'll see "Hello, world!" in the center of the screen.


Ta daaaahhhhhhh.

Displaying "Hello, world!" is the default behavior of the basic SwiftUI project template.

## One more thing

Note that the text is centered perfectly both horizontally and vertically.

Check this out.

Rotate the screen either using the keyboard combination Command - RightArrow, the simulator menu item Device > Rotate Right, or the top right button in the chrome above the simulator. You should see the app in landscape.


Again the text is perfectly centered.

I'm making a big deal about this because it used to require that we take some explicit steps to get this placement. In SwiftUI we get this for free. Later we'll look at how elements are sized and placed on the screen.

For now, I don't know about you, but I need to rest after all that work.

Just kidding.

I'll see you in the next section where we'll take a first look at what `ContentView` renders without running our app on device or in a simulator.