EXPERT INSIGHT

iOS 18 Programming for Beginners

Learn iOS development with Swift 6, Xcode 16, and iOS 18 – your path to App Store success

Ninth Edition

Ahmad Sahar

‹packt›

iOS 18 Programming for Beginners

Ninth Edition

Learn iOS development with Swift 6, Xcode 16, and iOS 18 – your path to App Store success

Ahmad Sahar

‹packt›

iOS 18 Programming for Beginners

Ninth Edition

Senior Publishing Product Manager: Larissa Pinto
Acquisition Editor – Peer Reviews: Swaroop Singh
Project Editor: K. Loganathan
Content Development Editor: Matthew Davies
Copy Editor: Safis Editing
Technical Editor: Simanta Rajbangshi
Proofreader: Safis Editing
Indexer: Pratik Shirodkar
Presentation Designer: Rajesh Shirsath
Developer Relations Marketing Executive: Sohini Ghosh

First published: December 2016
Second edition: January 2018
Third edition: December 2018
Fourth edition: January 2020
Fifth edition: November 2020
Sixth edition: December 2021
Seventh edition: November 2022
Eighth edition: October 2023
Ninth edition: November 2024

Production reference: 1041224

Published by Packt Publishing Ltd.
Grosvenor House
11 St Paul's Square
Birmingham
B3 1RB, UK.

ISBN 978-1-83620-489-3

www.packt.com

Contributors

About the author

Ahmad Sahar is a trainer, presenter, and consultant at Tomafuwi Productions. He is an Apple Certified Trainer for app development in Swift, and he has conducted training courses on for over ten years iOS mobile application development, Flutter mobile application development, and how to use Apple software and hardware. He also currently holds App Development with Swift Associate and Certified User certifications.

He is a member of the DevCon iOS online community in Malaysia and has conducted presentations and talks for this group. In his spare time, he likes building and programming LEGO MINDSTORMS® robots.

To my mother, Sharifah; my father, Sahar; my two sisters, Ainol Shareha and Ainol Shaharina; and my beloved wife, Oni – thank you for your love and support.

About the reviewer

Ian Lockett is a mobile application developer with over 18 years of experience in the software industry. He has spent the last decade developing iOS and Android apps for a wide range of clients and industries. Most of Ian's experience is in iOS development using Swift and Objective-C, but he has also created and maintained Android apps over the last few years using Kotlin.

Ian prides himself on attention to detail and delivering high-quality software. Ian lives in the UK with his wife and two children and now works for a large finance company having worked as a freelance/contract developer for the past seven years.

Join us on Discord!

Read this book alongside other users, experts, and the author himself. Ask questions, provide solutions to other readers, chat with the author via Ask Me Anything sessions, and much more. Scan the QR code or visit the link to join the community.

`https://packt.link/ios-Swift`

Table of Contents

Preface

Welcome to *iOS 18 Programming for Beginners*. This book is the ninth edition of the *iOS Programming for Beginners* series, and has been fully updated for iOS 18, macOS 15.0 Sequioa, and Xcode 16.

In this book, you will build a journal app called *JRNL*. You will start off by exploring Xcode, Apple's programming environment, also known as its **Integrated Development Environment** (**IDE**). Next, you will start learning the foundations of Swift, the programming language used in iOS apps, and see how it is used to accomplish common programming tasks.

Once you have a solid foundation of using Swift, you will start creating the user interface of the *JRNL* app. During this process, you will work with storyboards and connect your app's scenes together using segues.

With your user interface complete, you will then add code to implement your app's functionality. To start, you'll learn how to display data using a table view. Next, you'll learn how to add data to your app, and how to pass data between view controllers. After that, you'll learn how to determine your device location and display annotations on a map. You'll then learn how to persist app data using JSON files create custom views, and add photos from the camera or photo library. Finally, you'll make your app work on devices with larger screens, such as an iPad or Mac, by implementing a collection view in place of a table view.

You now have a complete app, but how about adding the latest iOS 18 features? You'll start by learning about SwiftData, which allows you to describe data models and manipulate model instances using regular Swift code. Next, you will learn how to develop apps using SwiftUI, a great new way of developing apps for all Apple platforms. After that, you'll learn how to test your code using Swift Testing, and how to bring Apple Intelligence features into your apps.

Finally, you'll learn how to test your app with internal and external testers and get it into the App Store.

Who this book is for

This book is tailored for individuals with minimal coding experience who are new to the world of Swift and iOS app development. A basic understanding of programming concepts is recommended.

What this book covers

Chapter 1, *Exploring Xcode*, takes you through a tour of Xcode and talks about all the different parts that you will use throughout the book.

Chapter 2, *Simple Values and Types*, deals with how values and types are implemented by the Swift language.

Chapter 3, *Conditionals and Optionals*, shows how `if` and `switch` statements are implemented, and how to implement variables that may or may not have a value.

Chapter 4, *Range Operators and Loops*, shows how to work with ranges and the different ways loops are implemented in Swift.

Chapter 5, *Collection Types*, covers the common collection types, which are arrays, dictionaries, and sets.

Chapter 6, *Functions and Closures*, covers how you can group instructions together using functions and closures.

Chapter 7, *Classes, Structures, and Enumerations*, talks about how complex objects containing state and behavior are represented in Swift.

Chapter 8, *Protocols, Extensions, and Error Handling*, talks about creating protocols that complex data types can adopt, extending the capabilities of existing types, and how to handle errors in your code.

Chapter 9, *Swift Concurrency*, introduces you to the concepts of parallel and asynchronous programming, and shows you how you can implement them in your app.

Chapter 10, *Setting Up the User Interface*, deals with creating the *JRNL* app and setting up the initial screen the users will see.

Chapter 11, *Building Your User Interface*, covers setting up the main screen for the *JRNL* app.

Chapter 12, *Finishing Up Your User Interface*, covers setting up the remaining screens for the *JRNL* app.

Chapter 13, *Modifying App Screens*, is about configuring each screen of the app in a storyboard.

Chapter 14, *Getting Started with MVC and Table Views*, covers working with a table view and how you can use it to display a list of items.

Chapter 15, *Getting Data into Table Views*, concerns the incorporation of data into table views using an array as a data source.

Chapter 16, *Passing Data between View Controllers*, teaches you how to add data entered using a view controller to an array, and how to pass data from the array to another view controller.

Chapter 17, *Getting Started with Core Location and MapKit*, deals with working with Core Location and MapKit to determine your device location and add annotations to a map.

Chapter 18, *Getting Started with JSON Files*, involves learning how to store and retrieve user data using a JSON file.

Chapter 19, *Getting Started with Custom Views*, teaches you how to create and use a custom view that displays a star rating.

Chapter 20, *Getting Started with the Camera and Photo Library*, talks about how to get photos from your camera or photo library into your app.

Chapter 21, *Getting Started with Search*, teaches you how to implement a search bar for your main screen.

Chapter 22, *Getting Started with Collection Views*, shows you how to implement collection views in place of table views to suit devices with larger screens, such as a Mac or iPad.

Chapter 23, *Getting Started with SwiftData*, deals with implementing Apple's new SwiftData framework to persist data on your app.

Chapter 24, *Getting Started with SwiftUI*, introduces building an app using Apple's new SwiftUI technology.

Chapter 25, *Getting Started with Swift Testing*, teaches you how to test your code using Swift Testing.

Chapter 26, *Getting Started with Apple Intelligence*, shows you how to add Apple Intelligence features to your app.

Chapter 27, *Testing and Submitting Your App to the App Store*, concerns how to test and submit your apps to the App Store.

To get the most out of this book

This book has been completely revised for iOS 18, macOS 15.0 Sequioa, Xcode 16, and Swift 6. *Part 4* of this book also covers the latest technologies introduced by Apple during WWDC 2024, which are SwiftData, SwiftUI, Swift Testing, and Apple Intelligence.

To complete all the exercises in this book, you will need:

- A Mac computer running macOS 14.0 Sonoma, macOS 15.0 Sequioa, or later
- Xcode 16.0 or later

To check if your Mac supports macOS 15.0 Sequioa, see this link: `https://www.apple.com/my/macos/macos-sequoia-preview/`. If your Mac is supported, you can update macOS using **Software Update** in **System Preferences**.

To get the latest version of Xcode, you can download it from the Apple App Store. Most of the exercises can be completed without an Apple Developer account and use the iOS Simulator. If you wish to test the app you are developing on an actual iOS device, you will need a free or paid Apple Developer account.

The following chapter requires a paid Apple Developer account: *Chapter 27*, *Testing and Submitting Your App to the App Store.* Instructions on how to get a paid Apple Developer account are included.

Download the example code files

You can download the example code files for this book from GitHub at `https://github.com/PacktPublishing/iOS-18-Programming-for-Beginners-Ninth-Edition`. If there's an update to the code, it will be updated in the GitHub repository.

We also have other code bundles from our rich catalog of books and videos available at `https://github.com/PacktPublishing/`. Check them out!

Code in Action

Visit the following link to check out videos of the code being run:

`https://www.youtube.com/playlist?list=PLeLcvrwLe185EJSoURfHhSHfbPFkiZl6m`

Download the color images

We also provide a PDF file that has color images of the screenshots/diagrams used in this book. You can download it here: `https://packt.link/gbp/9781836204893`.

Conventions used

There are a number of text conventions used throughout this book.

`CodeInText`: Indicates code words in text, database table names, folder names, filenames, file extensions, pathnames, dummy URLs, user input, and Twitter handles. Here is an example: "So, this is a very simple function, named `serviceCharge()`."

A block of code is set as follows:

```
class ClassName {
   property1
   property2
   property3
   method1() {
      code
   }
   method2() {
      code
   }
}
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
let cat = Animal()
cat.name = "Cat"
cat.sound = "Mew"
cat.numberOfLegs = 4
cat.breathesOxygen = true
print(cat.name)
```

Bold: Indicates a new term, an important word, or words that you see onscreen. For example, words in menus or dialog boxes appear in the text like this. Here is an example: "Launch **Xcode** and click **Create a new Xcode project:**"

Important notes

appear like this.

Tips

appear like this.

Get in touch

Feedback from our readers is always welcome.

General feedback: If you have questions about any aspect of this book, mention the book title in the subject of your message and email us at `customercare@packtpub.com`.

Errata: Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you have found a mistake in this book, we would be grateful if you would report this to us. Please visit `www.packtpub.com/support/errata`, selecting your book, clicking on the Errata Submission Form link, and entering the details.

Piracy: If you come across any illegal copies of our works in any form on the Internet, we would be grateful if you would provide us with the location address or website name. Please contact us at `copyright@packt.com` with a link to the material.

If you are interested in becoming an author: If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, please visit `authors.packtpub.com`.

Leave a Review!

Thank you for purchasing this book from Packt Publishing—we hope you enjoy it! Your feedback is invaluable and helps us improve and grow. Once you've completed reading it, please take a moment to leave an `Amazon review`; it will only take a minute, but it makes a big difference for readers like you.

Scan the QR code below to receive a free ebook of your choice.

`https://packt.link/NzOWQ`

Download a free PDF copy of this book

Thanks for purchasing this book!

Do you like to read on the go but are unable to carry your print books everywhere?

Is your eBook purchase not compatible with the device of your choice?

Don't worry, now with every Packt book you get a DRM-free PDF version of that book at no cost.

Read anywhere, any place, on any device. Search, copy, and paste code from your favorite technical books directly into your application.

The perks don't stop there, you can get exclusive access to discounts, newsletters, and great free content in your inbox daily.

Follow these simple steps to get the benefits:

1. Scan the QR code or visit the link below:

`https://packt.link/free-ebook/9781836204893`

2. Submit your proof of purchase.
3. That's it! We'll send your free PDF and other benefits to your email directly.

Part 1

Swift

Welcome to *Part 1* of this book. In this part, you will begin by exploring Xcode, Apple's programming environment, which is also known as the **Integrated Development Environment** (**IDE**). After that, you will start learning the foundations of Swift 6, the programming language used in iOS apps, and see how it is used to accomplish common programming tasks.

This part comprises the following chapters:

- *Chapter 1*, *Exploring Xcode*
- *Chapter 2*, *Simple Values and Types*
- *Chapter 3*, *Conditionals and Optionals*
- *Chapter 4*, *Range Operators and Loops*
- *Chapter 5*, *Collection Types*
- *Chapter 6*, *Functions and Closures*
- *Chapter 7*, *Classes, Structures, and Enumerations*
- *Chapter 8*, *Protocols, Extensions, and Error Handling*
- *Chapter 9*, *Swift Concurrency*

By the end of this part, you'll understand the process of creating an app and running it on Simulator or a device, and you'll have a working knowledge of how to use the Swift programming language in order to accomplish common programming tasks. This will prepare you for the next chapter and will also enable you to create your own Swift programs. Let's get started!

1

Exploring Xcode

Welcome to *iOS 18 Programming for Beginners*. I hope you will find this a useful introduction to creating and publishing iOS 18 apps on the App Store.

In this chapter, you'll download and install **Xcode** on your Mac. Then, you'll explore the Xcode user interface. After that, you'll create your first **iOS app** and run it in **Simulator**. Finally, you'll run your app on an **iOS device**.

By the end of this chapter, you will know how to create an iOS app, how to run it in Simulator, and how to run it on an iOS device.

The following topics will be covered in this chapter:

- Downloading and installing Xcode from the App Store
- Exploring the Xcode user interface
- Running your app in Simulator
- Running your app on an iOS device

Technical requirements

To do the exercises for this chapter, you will need the following:

- An Apple Mac computer (Apple Silicon or Intel) running macOS 14 Sonoma or macOS 15 Sequoia
- An Apple Account (if you don't have one, you will create one in this chapter)
- Optionally, an iOS device running iOS 18

The Xcode project for this chapter is in the `Chapter01` folder of the code bundle for this book, which can be downloaded here:

`https://github.com/PacktPublishing/iOS-18-Programming-for-Beginners-Ninth-Edition`

Check out the following video to see the code in action:

`https://youtu.be/g3mNosIoR8E`

You'll start by downloading Xcode, Apple's **integrated development environment** (**IDE**) for developing iOS apps from the App Store, in the next section.

The total size of the download is very large (2.98 GB for Xcode and 8.36 GB for iOS 18 Simulator), so it may take a while to download. Ensure that you have enough disk space prior to downloading.

Downloading and installing Xcode from the App Store

Xcode is Apple's IDE for developing macOS, iOS, iPadOS, watchOS, tvOS, and visionOS apps. You'll need to download and install Xcode on your Mac prior to writing your first app. Follow these steps:

1. On your Mac, choose **App Store** from the **Apple** menu.
2. In the search field in the top-right corner, type `Xcode` and press the *Return* key.
3. You'll see **Xcode** in the search results. Click **Get** and then click **Install**.
4. If you have an Apple Account, type it in the text field and enter your password when prompted. If you don't have one, click the **Create Apple Account** button and follow the step-by-step instructions to create one:

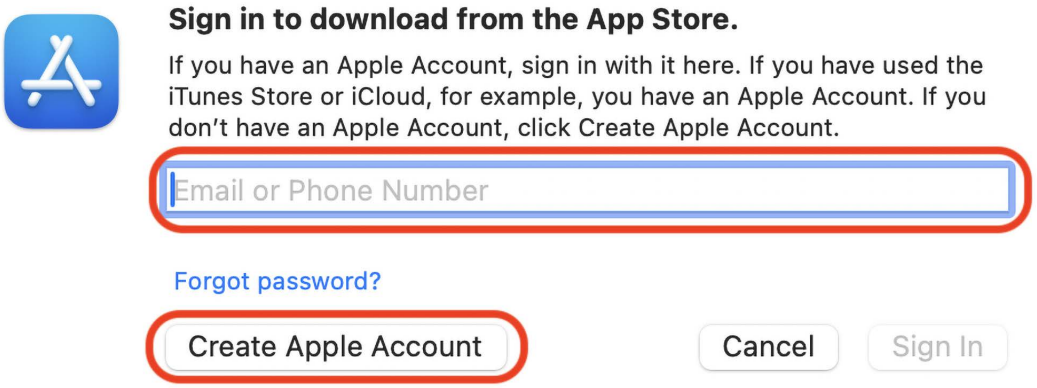


Figure 1.1: Apple account creation dialog box

You can see more information on how to create an Apple Account using this link: `https://support.apple.com/en-us/108647#appstore`.

5. Once Xcode has been installed, launch it. You'll see a license agreement screen. Click **Agree**:

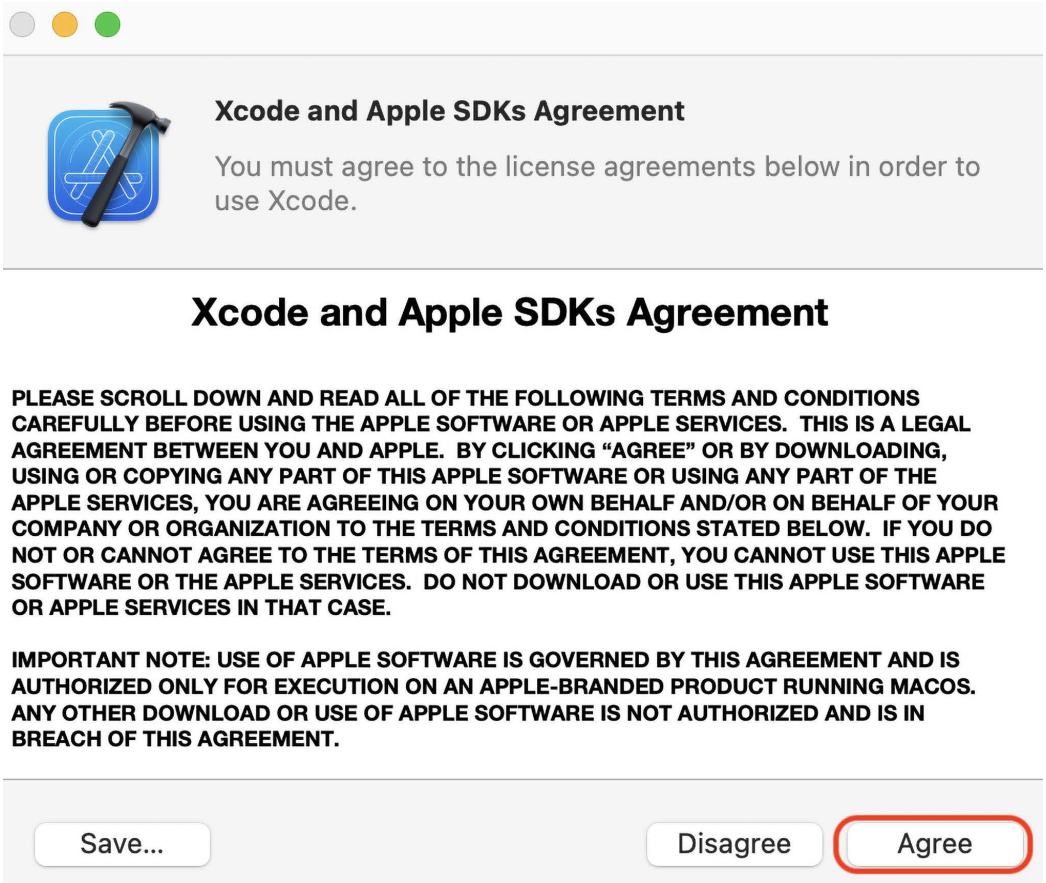


Figure 1.2: License agreement screen

6. You'll be prompted to enter your Mac's administrator **username** and **password**. Once you have done so, click **OK**:

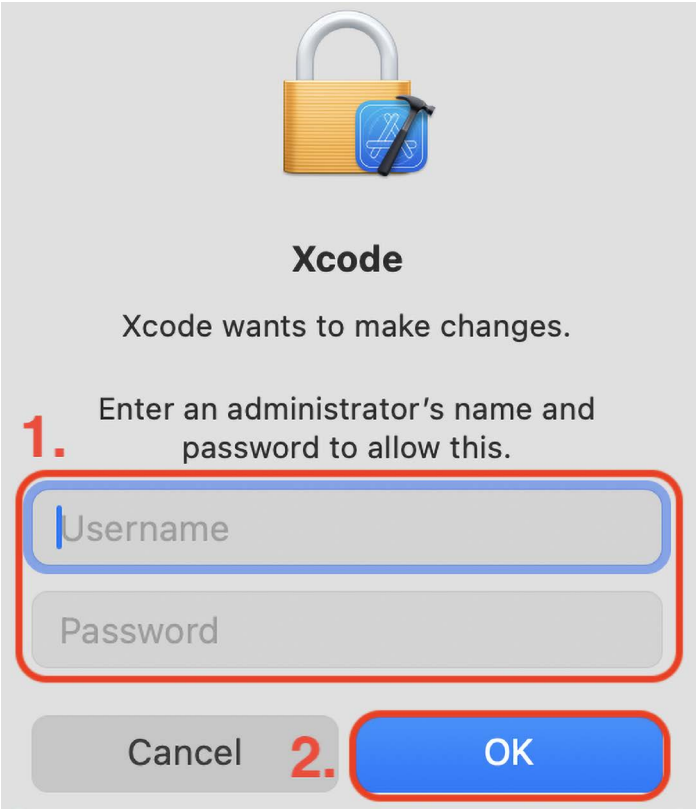


Figure 1.3: Prompt for administrator username and password

7. You'll see a screen showing you the available development platforms. You just need macOS and iOS for now. Tick **iOS 18.0**, leave all other options unticked, and click **Download & Install**:

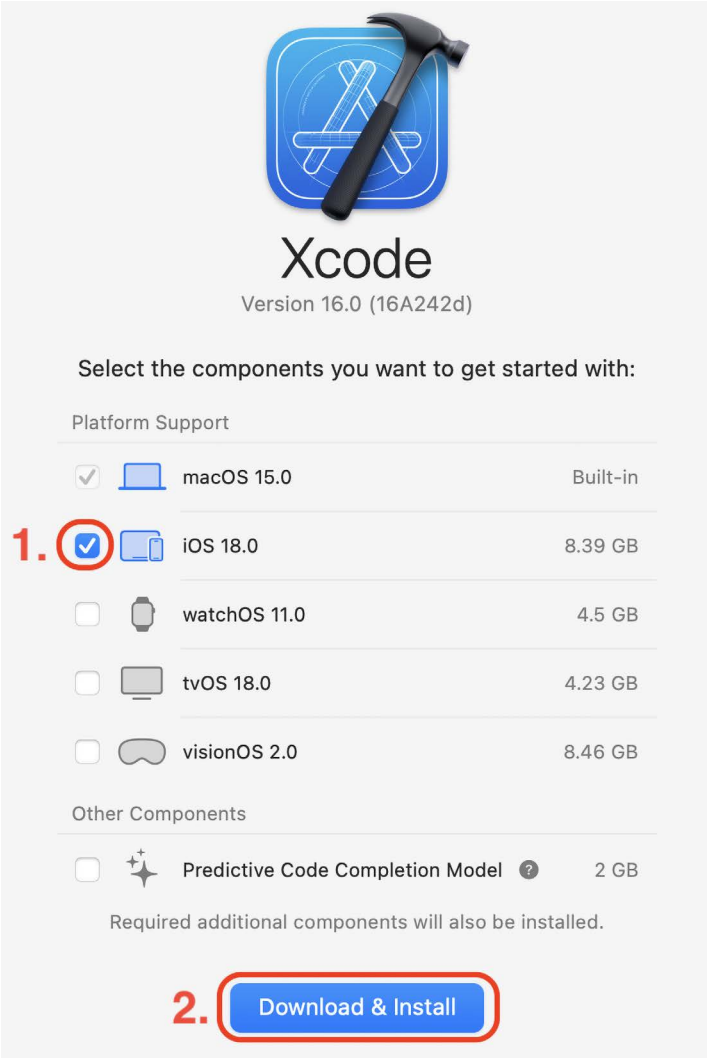
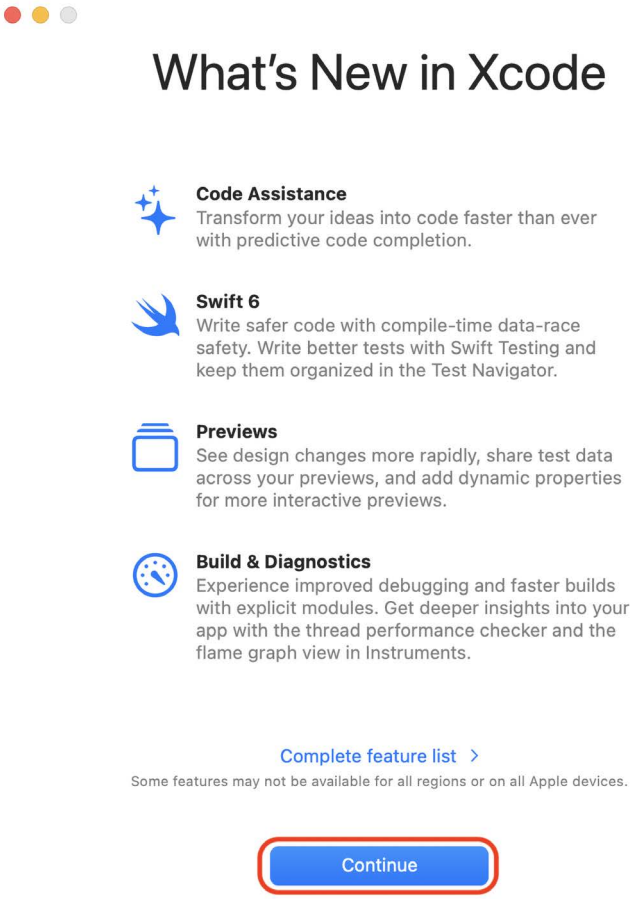


Figure 1.4: Development platforms screen

8. Xcode will prompt you to relaunch to use updated frameworks. Click **Relaunch Xcode**:

Figure 1.5: Relaunch Xcode prompt

9. You'll see a **What's New in Xcode** screen. Click **Continue**:



f

Figure 1.6: What's New in Xcode screen

10. You'll see the **Welcome to Xcode** screen. Click **Create New Project...** in the left-hand pane:

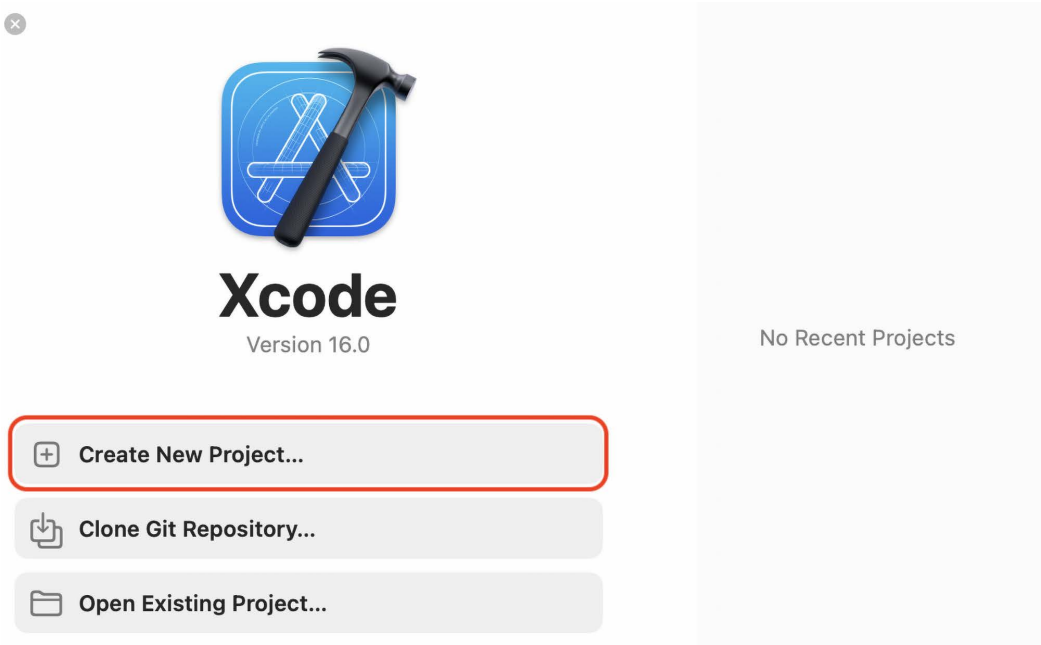


Figure 1.7: Welcome to Xcode screen

11. Xcode will start to download **iOS 18.0 Simulator** automatically. Note that you will not be able to run any apps on Simulator until this process has been completed:

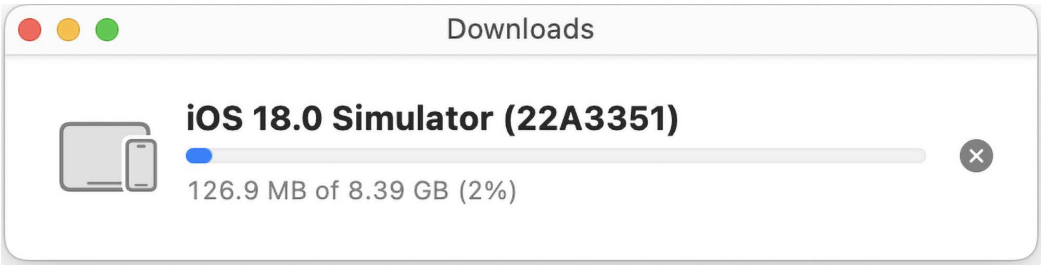


Figure 1.8: Simulator download progress bar

12. You'll see the new project screen as follows. In the **Choose a template for your new project** section, select **iOS**. Then choose **App** and click **Next:**

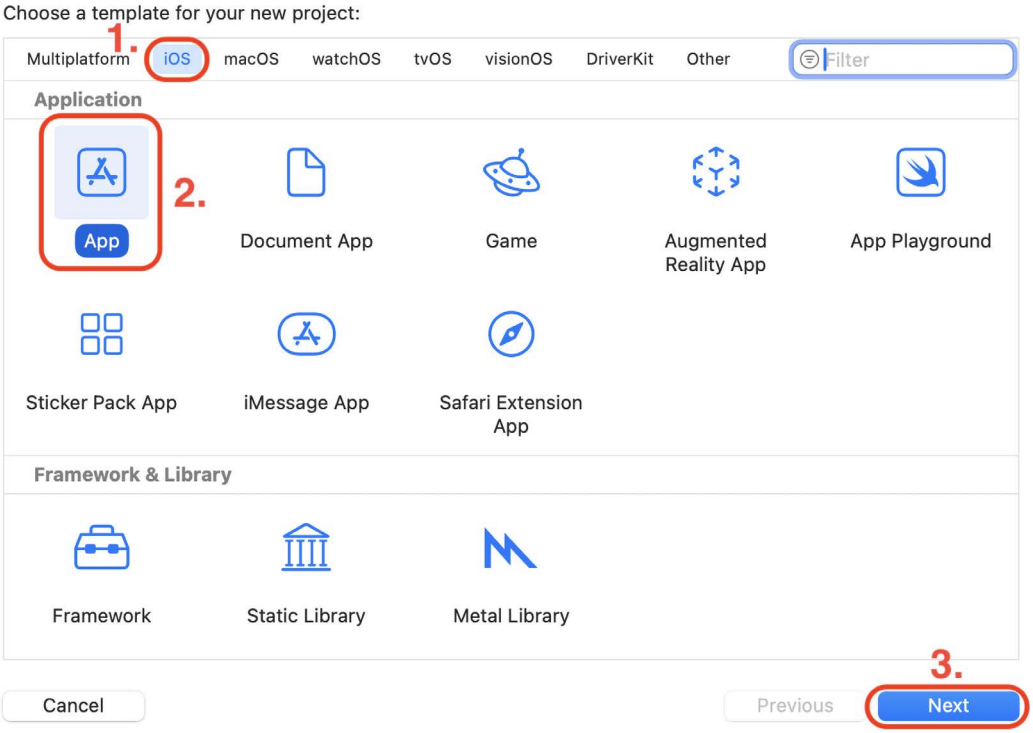


Figure 1.9: New project screen

13. You'll see the **Choose options for your new project** screen:

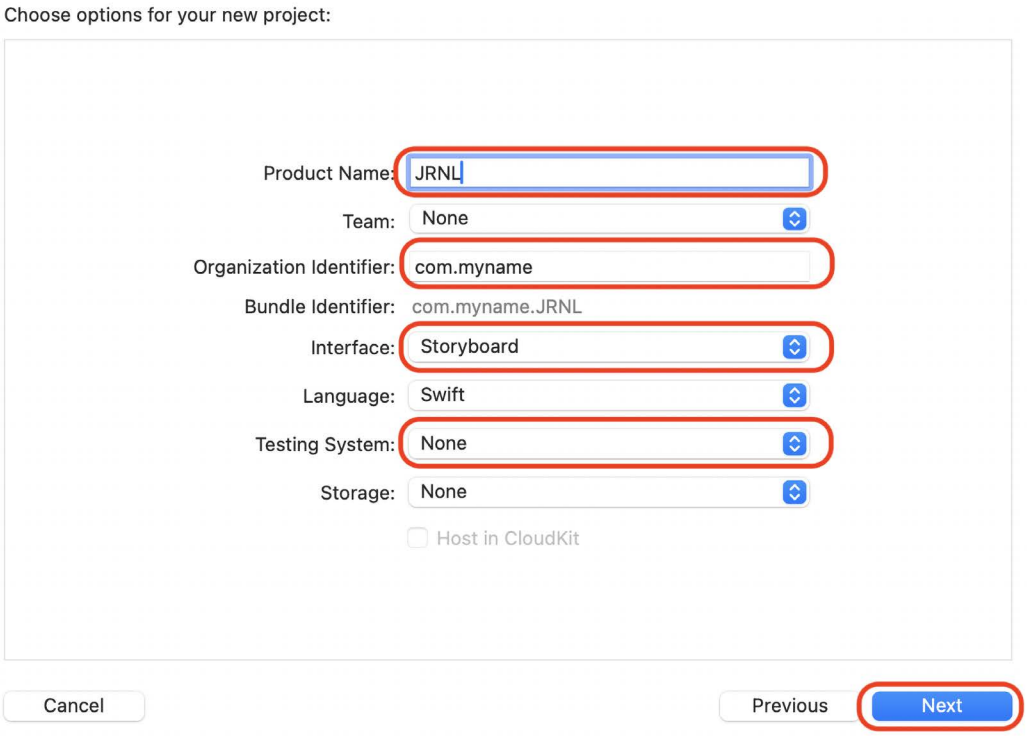


Figure 1.10: Choose options for your new project screen

Configure the options as follows:

- **Product Name:** The name of your app. Enter `JRNL` in the text field.
- **Organization Identifier:** Used to create a unique identifier for your app on the App Store. Enter `com.myname` for now. This is known as the reverse domain name notation format and is commonly used by iOS developers.
- **Interface:** The method used to create the user interface for your app. Set this to **Storyboard**.
- **Testing System:** The testing system you will use. You will learn about this in *Chapter 25, Getting Started with Swift Testing*. Set it to **None** for now.

Leave the other settings at their default values. Click **Next** when done.

14. You'll see a **Save** dialog box. Choose a location to save your project, such as the **Desktop** or **Documents** folder, and click **Create**:

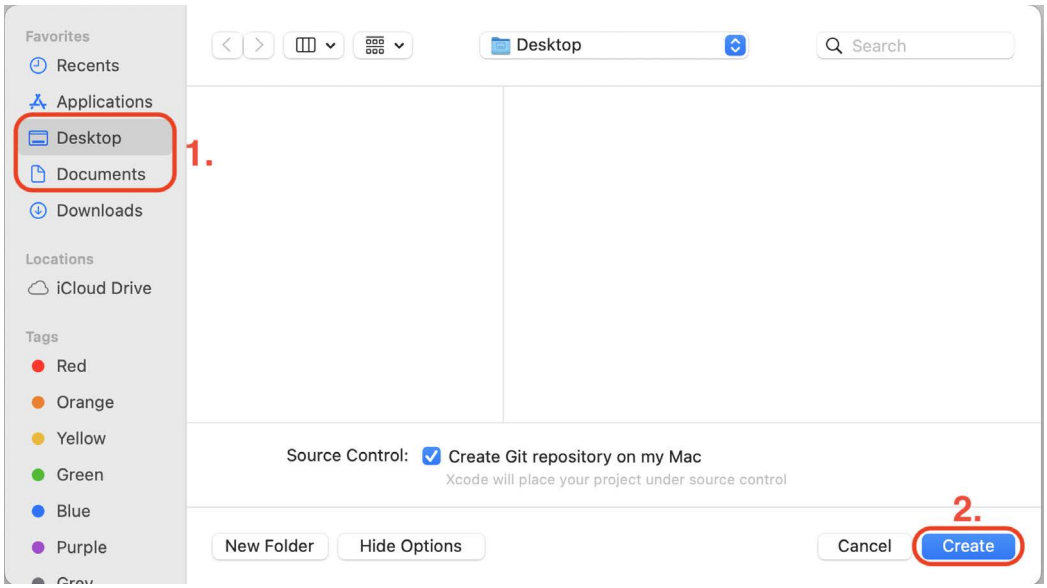


Figure 1.11: Save dialog box

15. You'll see a dialog box saying **Git Repository Creation Failed**. Click **Fix**.

The reason why you see this dialog box is because the **Source Control** checkbox in the **Save** dialog box was ticked. Apple recommends that **Source Control** be turned on. **Source Control** is outside the scope of this book but if you wish to learn more about version control and Git, see this link: https://git-scm.com/video/what-is-version-control.

16. You will see the **Source Control** screen as follows:

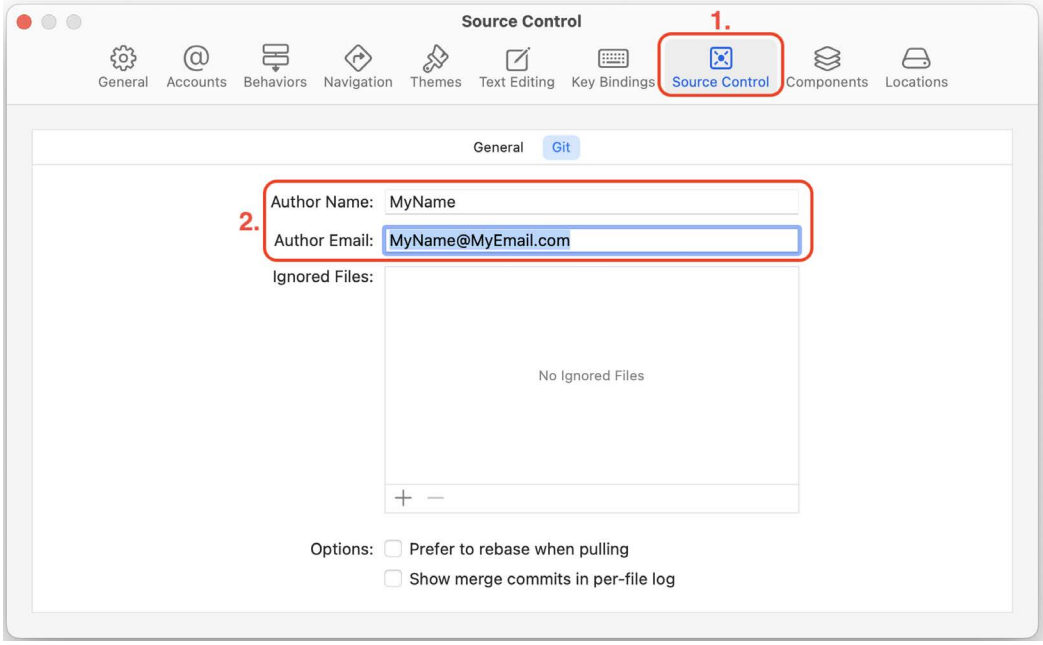


Figure 1.12: Source Control preference screen

Enter the following information:

- **Author Name:** Your own name
- **Author Email:** Your email address

Close the **Source Control** screen by clicking the close button in the top-left corner when done. The Xcode main window will appear.

Fantastic! You have now successfully downloaded and installed Xcode and created your first project. In the next section, you will learn about the Xcode user interface.

Exploring the Xcode user interface

You've just created your first Xcode project! As you can see, the Xcode user interface is divided into several distinct parts, as shown here:

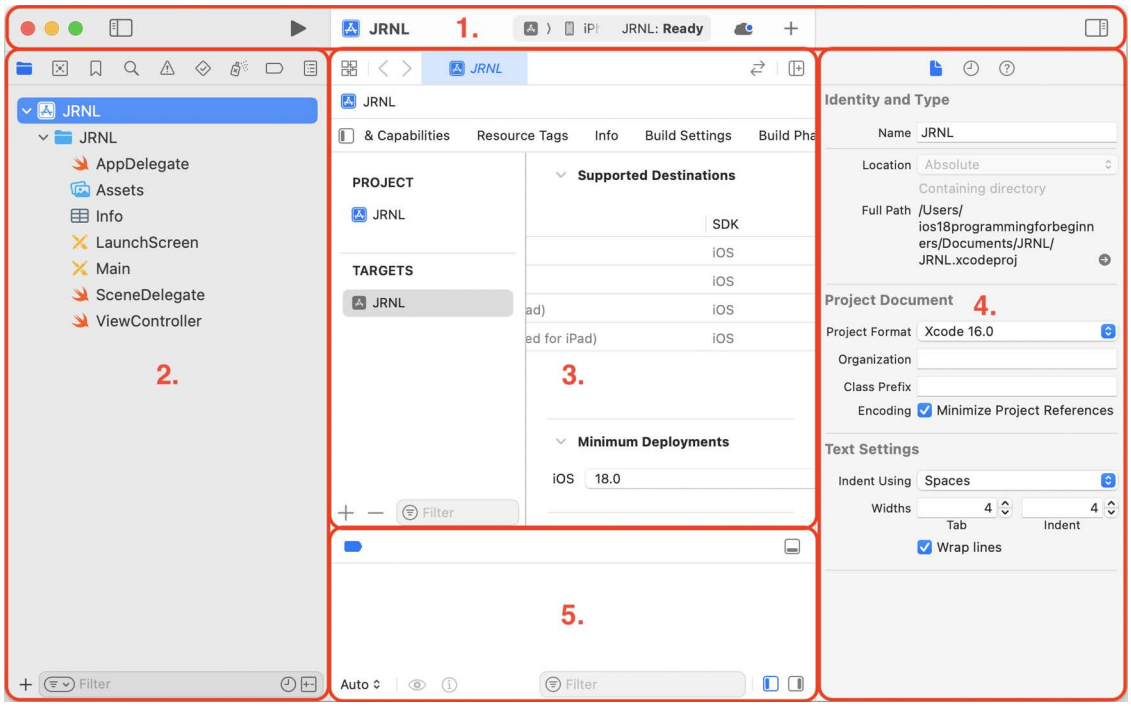


Figure 1.13: Xcode user interface

Let's look at each part in more detail. The following description corresponds to the numbers shown in the preceding screenshot:

- **Toolbar (1)**: Used to build and run your apps, and view the progress of running tasks.
- **Navigator area (2)**: Provides quick access to the various parts of your project. The **Project navigator** is displayed by default.
- **Editor area (3)**: Allows you to edit source code, user interfaces, and other resources.
- **Inspector area (4)**: Allows you to view and edit information about items selected in the **Navigator area** or **Editor area**.
- **Debug area (5)** – Contains the **debug bar**, the **variables view**, and the **Console**. The **Debug area** is toggled by pressing *Shift* + *Command* + *Y*.

Next, let's examine the toolbar more closely. The left side of the toolbar is shown here:

Figure 1.14: Xcode toolbar (left side)

Let's look at each part in more detail. The following descriptions correspond to the numbers shown in the preceding screenshot:

- **Navigator button (1)** – Used to display and hide the Navigator area.
- **Stop button (2)** – Only appears next to the Run button when the app is running. Stops the currently running app.
- **Run button (3)** – Used to build and run your app.
- **Scheme menu (4)** – Shows the specific scheme to build your project (**JRNL**) and the destination to run your app on (**iPhone SE (3rd generation)**). Schemes and destinations are distinct. Schemes specify the settings for building and running your project. Destinations specify installation locations for your app and exist for Simulator and physical devices.
- **Activity view (5)** – Displays the progress of running tasks.

The right side of the toolbar is shown here:

Figure 1.15: Xcode toolbar (right side)

Let's look at each part in more detail. The following descriptions correspond to the numbers shown in the preceding screenshot:

- **Xcode Cloud button (1)** – Allows you to sign in to Xcode Cloud, a continuous integration and delivery service built into Xcode.
- **Library button (2)** – Displays user interface elements, code snippets, and other resources.
- **Inspector button (3)** – Used to display and hide the Inspector area.

Don't be overwhelmed by all the different parts, as you'll learn about them in more detail in the upcoming chapters. Now that you are familiar with the Xcode interface, you will run the app you just created in Simulator, which displays a representation of an iOS device.

Running your app in Simulator

Simulator is downloaded and installed after you install Xcode. It provides a simulated iOS device so that you can see what your app looks like and how it behaves, without needing a physical iOS device. It can model all the screen sizes and resolutions for both iPad and iPhone so you can test your app on multiple devices easily.

To run your app in Simulator, follow these steps:

1. Click the Destination pop-up menu to view a list of simulated devices. Choose **iPhone SE (3rd generation)** from this menu:

Figure 1.16: Xcode Destination pop-up menu with iPhone SE (3rd generation) selected

In your own projects, you should pick whichever simulator you require. That said, if you want to match the screenshots in this book exactly, use the **iPhone SE (3rd generation)** simulator. This simulator also has a home button, so it is easier to get to the home screen.

2. Click the Run button to install and run your app on the currently selected simulator. You can also use the *Command* + *R* keyboard shortcut.

3. Simulator will launch and show a representation of an iPhone SE (3rd generation). Your app displays a white screen, as you have not yet added anything to your project:

Figure 1.17: Simulator displaying your app

4. Switch back to Xcode and click on the Stop button (or press *Command* + .) to stop the currently running project.

You have just created and run your first iOS app in Simulator! Great job!

The Destination menu has a section showing physical devices connected to your Mac and a **Build** section. You may be wondering what they are used for. Let's look at them in the next section.

Understanding the Build section

You learned how to choose a simulated device in the Destination menu to run your app in the previous section. In addition to the list of simulated devices, this menu also has a section showing physical devices connected to your Mac, and a **Build** section.

These allow you to run apps on actual Mac or iOS devices and prepare apps for submission to the App Store.

Click the Destination menu in the toolbar to see the physical device and **Build** sections at the top of the menu:

Figure 1.18: Xcode Destination menu showing device and Build sections

If you have an Apple Silicon Mac, the physical device section will display text stating **My Mac (Designed for iPad)**, because Apple Silicon Macs can run iOS apps. Otherwise, **No Devices** will be displayed. If you were to plug in an iOS device, it would appear in this section, and you would be able to run the apps you develop on it for testing. Running your apps on an actual device is recommended as Simulator will not accurately reflect the performance characteristics of an actual iOS device and does not have hardware features that actual devices have.

The **Build** section has two menu items, **Any iOS Device(arm64)** and **Any iOS Simulator Device (arm64, x86_64)**. These are used when you need to archive your app prior to submitting it to the App Store. You'll learn how to do this in *Chapter 27*, *Testing and Submitting Your App to the App Store*.

Now let's see how to build and run your app on an actual iOS device. Most of the instructions in this book do not require you to have an iOS device though, so if you don't have one, you can skip the next section and go straight to *Chapter 2*, *Simple Values and Types*.

Running your app on an iOS device

Although you'll be able to go through most of the exercises in this book using Simulator, it is recommended to build and test your apps on an actual iOS device, as Simulator will not be able to simulate some hardware components and software APIs.

For a comprehensive look at all the differences between Simulator and an actual device, see this link: `https://help.apple.com/simulator/mac/current/#/devb0244142d`.

In addition to your device, you'll need an Apple Account (used to automatically create a free Apple developer account) or a paid Apple developer account to build and run your app on your device. You can use the same Apple Account that you used to download Xcode from the App Store. To run your app on an iOS device, follow these steps:

1. Use the cable that came with your iOS device to connect your device to your Mac, and make sure the iOS device is unlocked.
2. Your Mac will display an **Allow Accessory to Connect** alert. Click **Allow**.
3. Your iOS device will display a **Trust This Computer** alert. Tap **Trust** and key in your device passcode when prompted. Your iOS device is now connected to your Mac and will appear in Xcode's Destination menu.
4. Choose **Window | Devices and Simulators** in the Xcode menu bar. You will see a window displaying a message saying **Developer Mode disabled**:

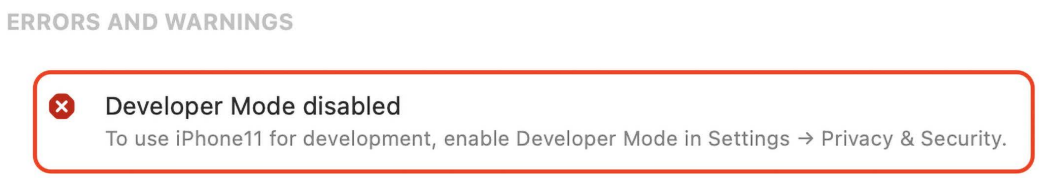


Figure 1.19: Xcode Devices and Simulators window showing Developer Mode disabled

Developer Mode was introduced by Apple during their Worldwide Developers Conference in 2022 (WWDC 2022) and is required to install, run, and debug your apps on devices running iOS 16 or greater.

To watch a WWDC 2022 video on Developer Mode, click this link: https://developer.apple.com/videos/play/wwdc2022/110344/.

5. To enable Developer Mode on your iOS device, go to **Settings** | **Privacy & Security**, scroll down to the **Developer Mode** item, and tap it:

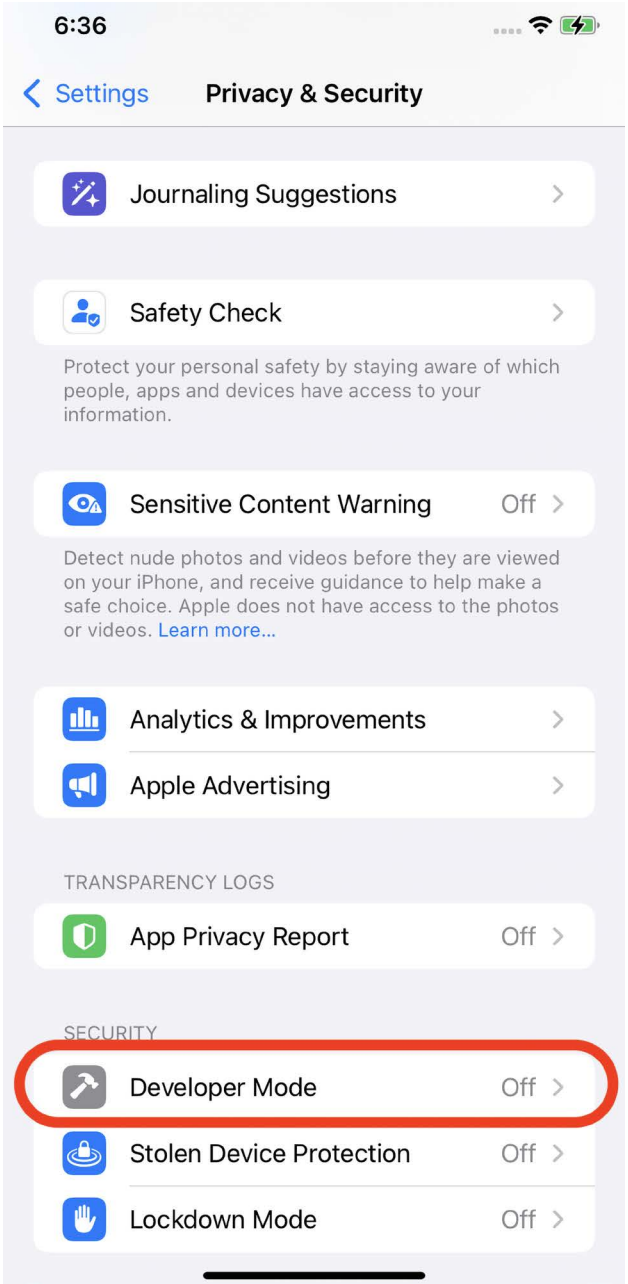


Figure 1.20: Privacy & Security screen showing Developer Mode

6. Turn the **Developer Mode** switch on:

Figure 1.21: Developer Mode switch

7. An alert will appear to warn you that Developer Mode reduces the security of your iOS device. Tap the alert's **Restart** button.
8. After your iOS device restarts and you unlock it, confirm that you want to enable **Developer Mode** by tapping **Enable** and entering your iOS device's passcode.
9. The Devices and Simulators window will display a **Preparing iPhone** message. Wait a few minutes, then verify that the Devices and Simulators window no longer displays the **Developer Mode disabled** text:

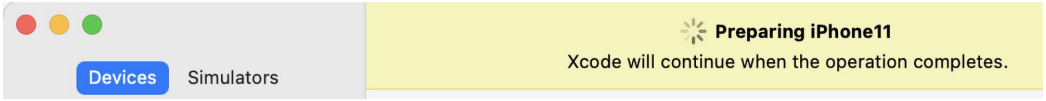


Figure 1.22: Xcode Devices and Simulators window showing Preparing iPhone message

Your iOS device is now ready to install and run apps from Xcode.

10. In Xcode, choose your iOS device from the Destination menu.

11. Run the project by clicking the Run button (or use *Command* + *R*). You will get the following error in Xcode's **Signing & Capabilities** panel: **Signing for "JRNL" requires a development team:**

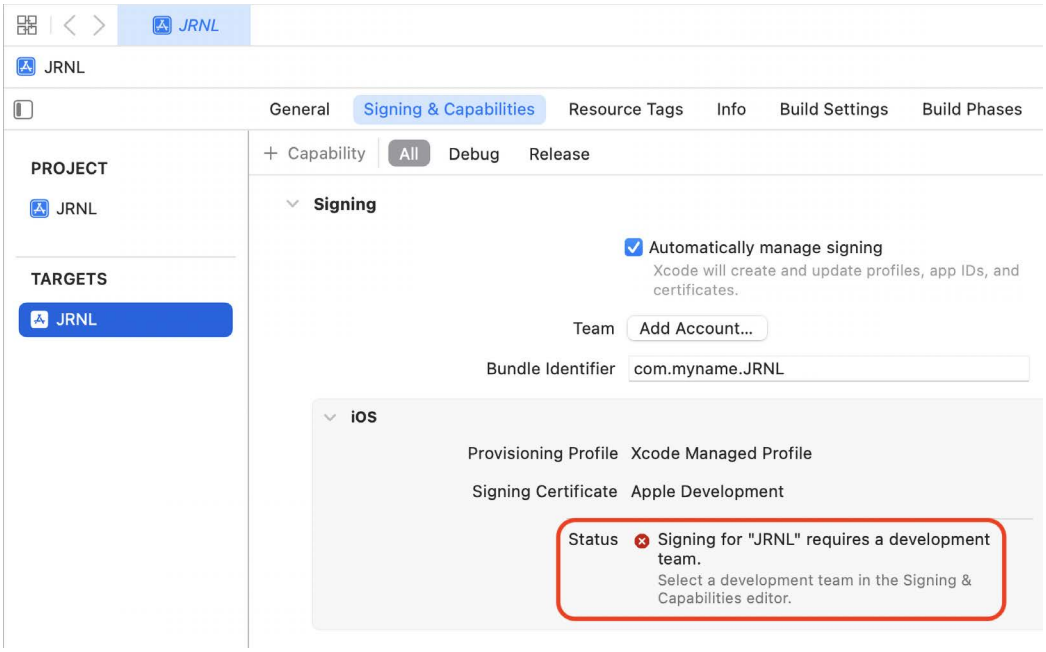


Figure 1.23: Xcode Signing & Capabilities panel

This is because a digital certificate is required to run the app on an iOS device, and you need to add a free or paid Apple developer account to Xcode so the digital certificate can be generated.

Using an Apple Account to create a free developer account will allow you to test your app on an iOS device, but it will only be valid for 7 days. Also, you will need a paid Apple developer account to distribute apps on the App Store. You'll learn more about this in *Chapter 27*, *Testing and Submitting Your App to the App Store*.

Certificates ensure that the only apps that run on your device are the ones you authorize. This helps to protect against malware. You can also learn more about them at this link: `https://help.apple.com/xcode/mac/current/#/dev60b6fbbc7`.

12. Click the **Add Account...** button:

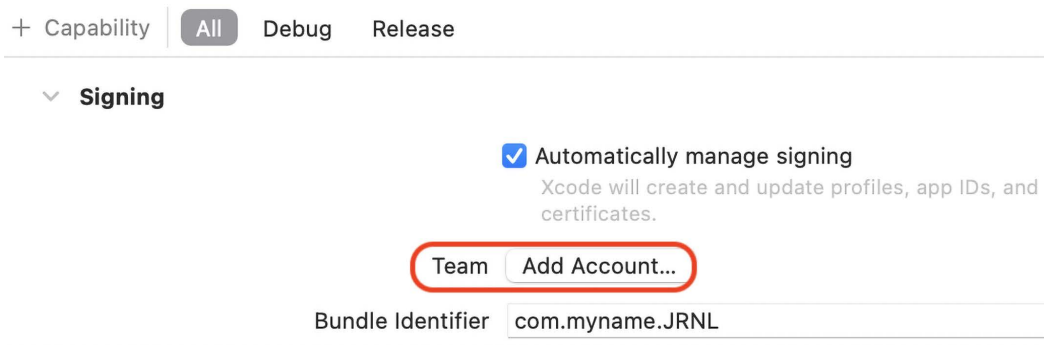


Figure 1.24: Xcode Signing & Capabilities pane with the Add Account… button selected

13. The Xcode **Settings** window appears with the **Accounts** pane selected. Enter your Apple Account and click **Next:**

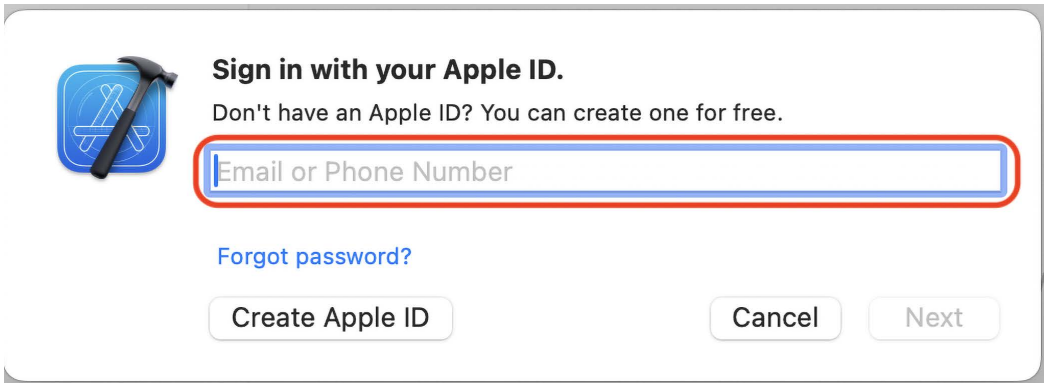


Figure 1.25: Apple Account creation dialog box

Note that you can create a different Apple Account if you wish, using the **Create Apple ID** button.

You can also access the Xcode settings by choosing **Settings** in the Xcode menu.

14. Enter your password when prompted. After a few minutes, the **Accounts** pane will display your account settings:

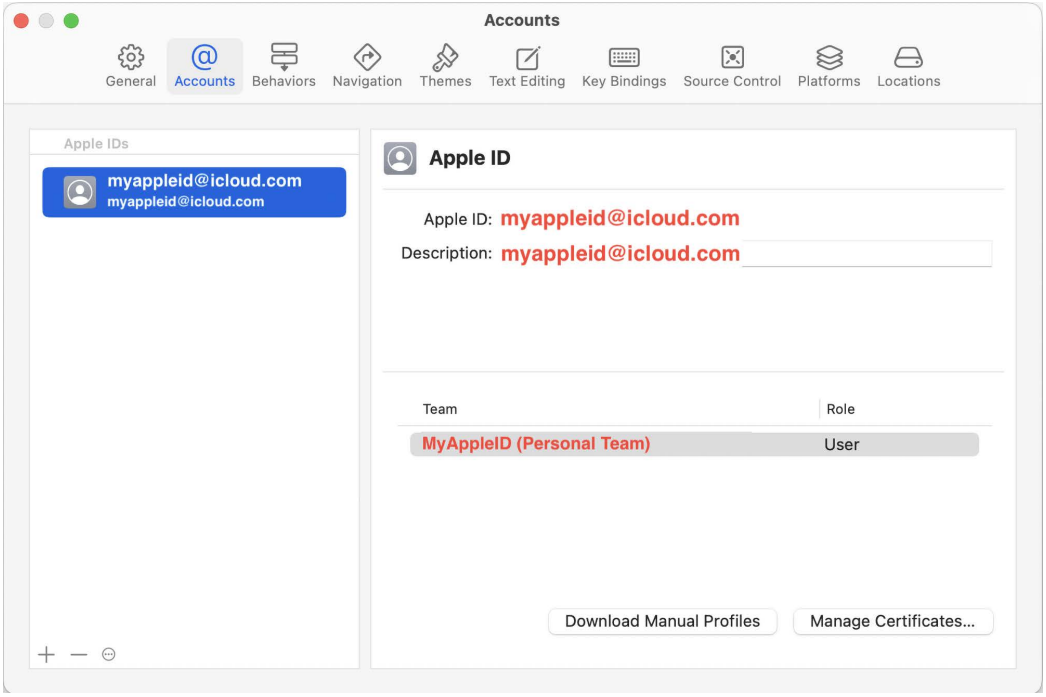


Figure 1.26: Accounts pane in Xcode preferences

15. Close the **Settings** window when you're done by clicking the red close button in the top-left corner.
16. In Xcode's editor area, click **Signing & Capabilities**. Make sure **Automatically manage signing** is ticked and **Personal Team** is selected from the **Team** pop-up menu:

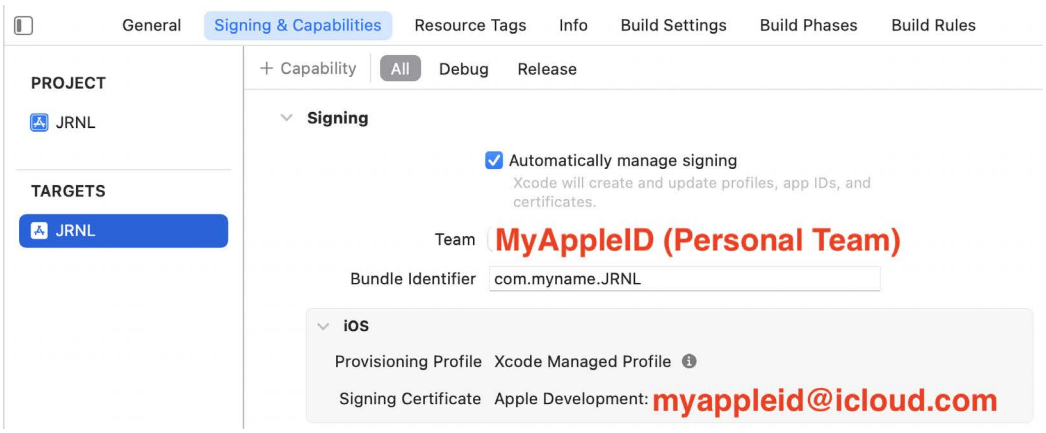


Figure 1.27: Xcode Signing & Capabilities pane with account set

17. If you still see errors on this screen, try changing your **Bundle Identifier** by typing some random characters into it, for example, `com.myname6712.JRNL`.
18. Build and run your app. If you are prompted for a password, enter your Mac's login password and click **Always Allow**.
19. Your app will be installed on your iOS device. However, it will not launch, and you will see the following message:

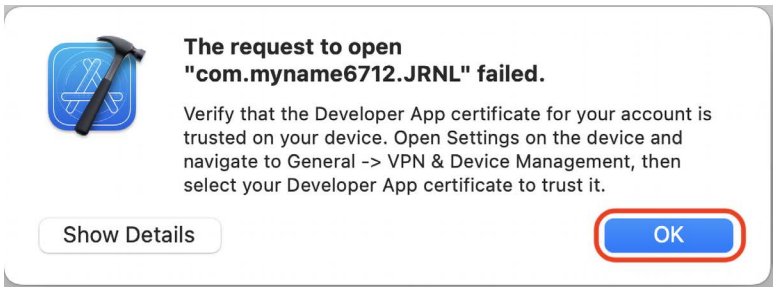


Figure 1.28: Could not launch "JRNL" dialog box

20. This means you need to trust the certificate that has been installed on your device. You'll learn how to do this in the next section.

Trusting the Developer App certificate on your iOS device

A **Developer App certificate** is a special file that gets installed on your iOS device along with your app. Before your app can run, you need to trust it. Follow these steps:

1. On your iOS device, tap **Settings** | **General** | **VPN & Device Management**:

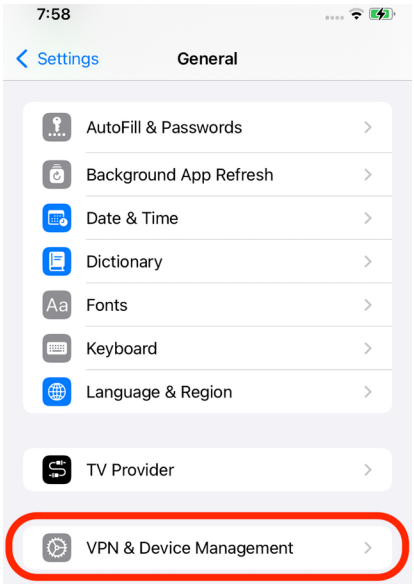


Figure 1.29: VPN & Device Management setting in Settings

2. Tap your Apple Account:

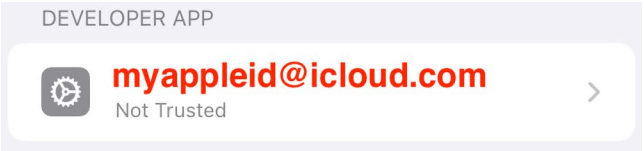


Figure 1.30: Your Apple Account in Device Management settings

3. Tap **Trust:**

Figure 1.31: Trust button

4. Tap **Allow:**

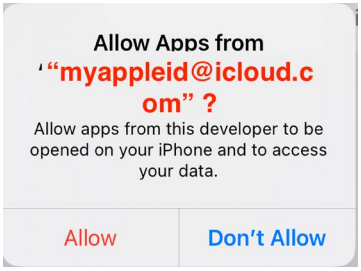


Figure 1.32: Allow dialog box

You should see the following text, which shows the app is now trusted:

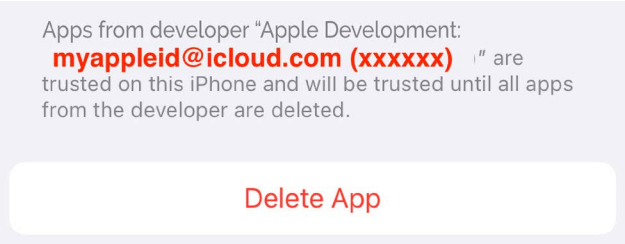


Figure 1.33: Device management section with trusted certificate

5. Click the Run button in Xcode to build and run again. You’ll see your app launch and run on your iOS device.

Congratulations! You have successfully run your app on an actual iOS device!

Summary

In this chapter, you learned how to download and install Xcode on your Mac. Then, you familiarized yourself with the different parts of the Xcode user interface. After that, you created your first iOS app, selected a simulated iOS device, and built and ran the app in Simulator. Finally, you learned how to connect an iOS device to Xcode via USB so that you can run your app on it.

In the next chapter, we'll start exploring the Swift language using Swift Playgrounds, and learn how simple values and types are implemented in Swift.

Join us on Discord!

Read this book alongside other users, experts, and the author himself. Ask questions, provide solutions to other readers, chat with the author via Ask Me Anything sessions, and much more. Scan the QR code or visit the link to join the community.

`https://packt.link/ios-Swift`

2

Simple Values and Types

Now that you have had a short tour of Xcode in the previous chapter, let's look at the Swift programming language, which you will use to write your app.

First, you'll explore **Swift playgrounds**, interactive environments where you can type in Swift code and have the results displayed immediately. Then, you'll study how Swift represents and stores various types of data. After that, you'll look at some cool Swift features, such as **type inference** and **type safety**, which help you to write code more concisely and avoid common errors. Finally, you'll learn how to perform common operations on data and how to print messages to the Debug area to help you troubleshoot issues.

By the end of this chapter, you should be able to write simple programs that can store and process letters and numbers.

The following topics will be covered:

- Introducing Swift playgrounds
- Exploring data types
- Exploring constants and variables
- Understanding type inference and type safety
- Exploring operators
- Using the `print()` statement

For more information about the latest version of the Swift language, visit `https://docs.swift.org/swift-book/documentation/the-swift-programming-language/`.

Technical requirements

To do the exercises in this chapter, you will need the following:

- An Apple Mac computer running macOS 14 Sonoma or macOS 15 Sequoia
- Xcode 16 installed (refer to *Chapter 1*, *Exploring Xcode*, for instructions on how to install Xcode)

The Xcode playground for this chapter is in the `Chapter02` folder of the code bundle for this book, which can be downloaded here:

`https://github.com/PacktPublishing/iOS-18-Programming-for-Beginners-Ninth-Edition`

Check out the following video to see the code in action:

`https://youtu.be/lSBD_Wacpdc`

In the next section, you'll create a new playground, where you can type in the code presented in this chapter.

Introducing Swift playgrounds

Playgrounds are interactive coding environments. You type code in the left-hand pane, and the results are displayed immediately in the right-hand pane. It's a great way to experiment with code and explore the iOS SDK.

SDK is an acronym for software development kit. To learn more about the iOS SDK, visit `https://developer.apple.com/ios/`.

Let's start by creating a new playground and examining its user interface. Follow these steps:

1. To create a playground, launch Xcode and choose **File** | **New** | **Playground...** from the Xcode menu bar:

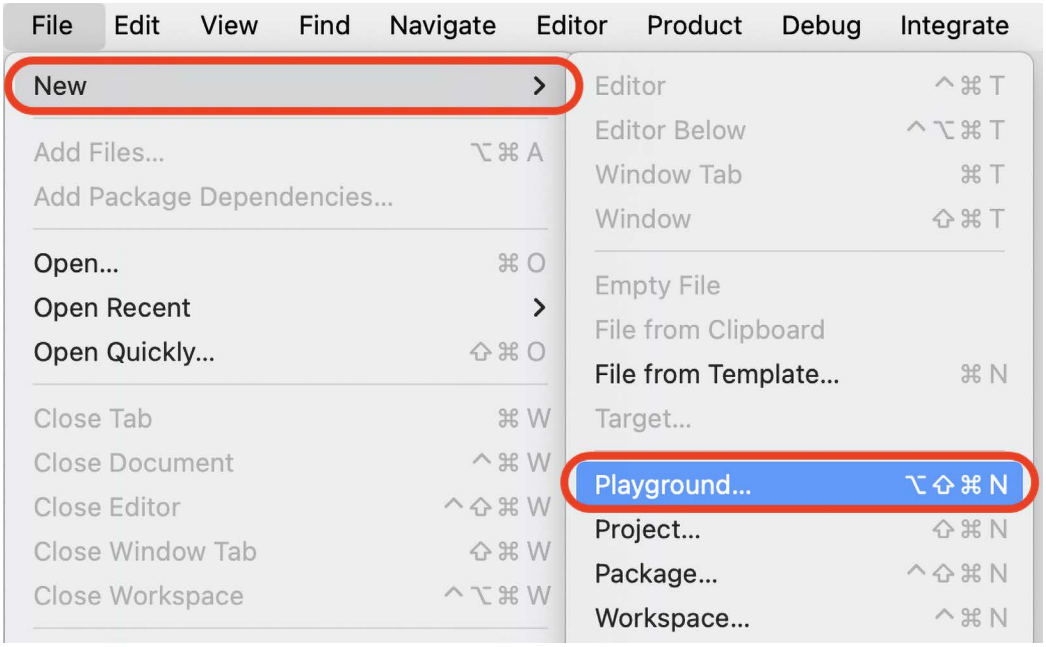


Figure 2.1: Xcode menu bar with File | New | Playground... selected

2. The **Choose a template for your new playground:** screen appears. **iOS** should already be selected. Choose **Blank** and click **Next:**

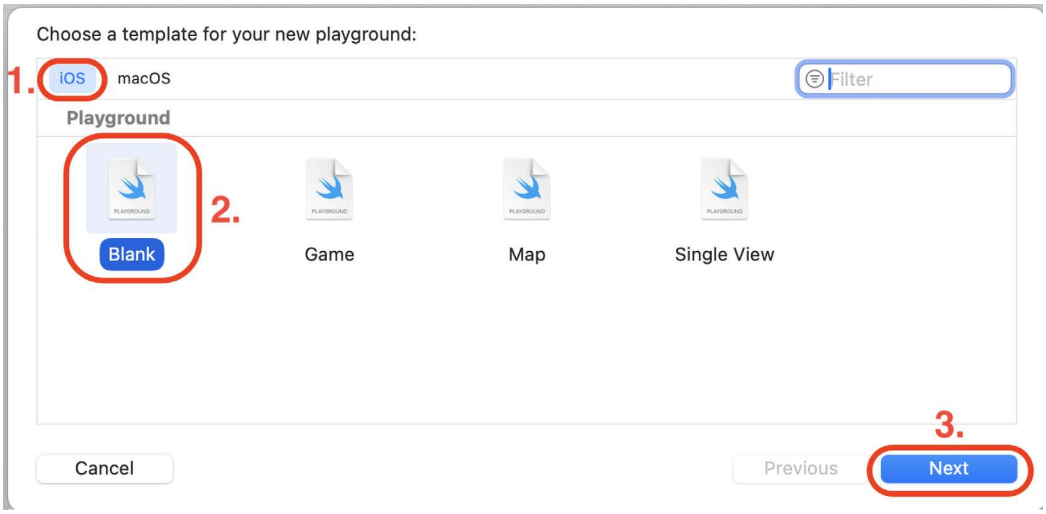


Figure 2.2: The Choose a template for your new playground: screen

3. Name your playground `SimpleValues` and save it anywhere you like. Click **Create** when done:

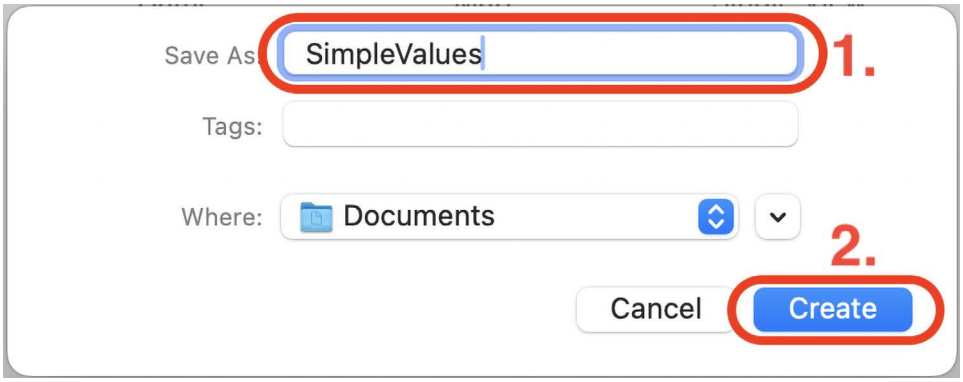


Figure 2.3: Save dialog box

4. You'll see the playground on your screen:

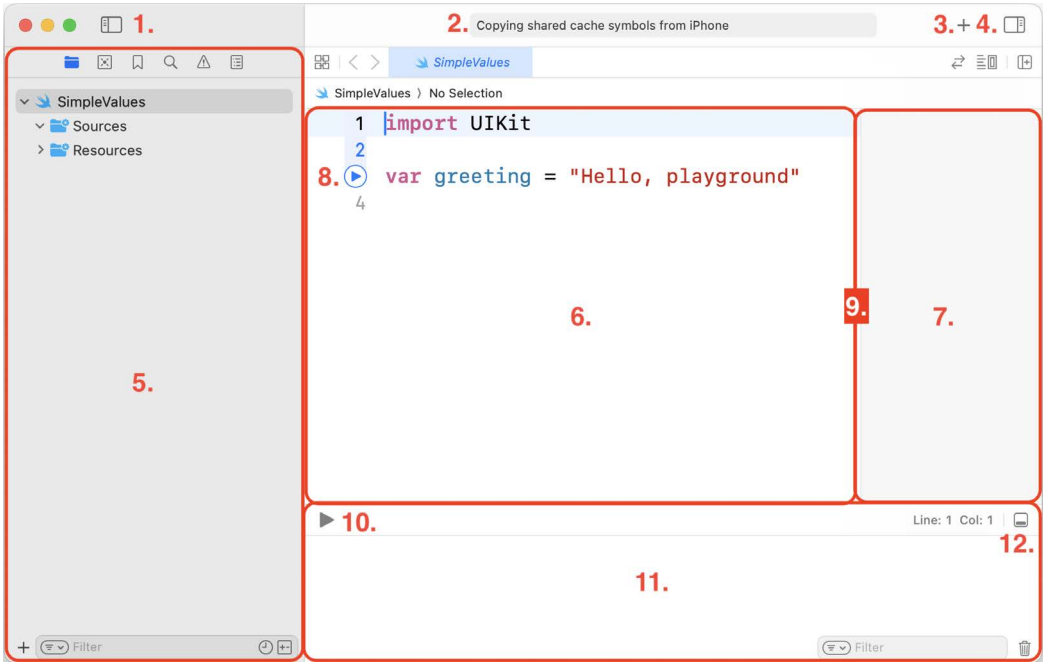


Figure 2.4: Xcode playground user interface

As you can see, it's much simpler than an Xcode project. Let's look at the interface in more detail:

- **Navigator button (1):** This shows or hides the Navigator area.
- **Activity View (2):** This shows the current operation or status.
- **Library button (3):** This displays code snippets and other resources.
- **Inspector button (4):** This shows or hides the Inspector area.

- **Navigator area (5):** This provides quick access to various parts of your project. The Project navigator is displayed by default.
- **Editor area (6):** You write code here.
- **Results area (7):** This provides immediate feedback on the code you write.
- **Run button (8):** This executes code from a selected line.
- **Border (9):** This border separates the Editor and Results areas. If you find that the results displayed in the Results area are truncated, drag the border to the left to increase its size.
- **Run/Stop button (10):** This executes or stops the execution of all code in the playground.
- **Debug area (11):** This displays the results of the `print()` command.
- **Debug button (12):** This shows and hides the Debug area.

You may find the text in the playground too small and hard to read. Let's see how to make it larger in the next section.

Customizing fonts and colors

Xcode has extensive customization options available. You can access them in the **Settings...** menu. If you find that the text is small and hard to see, follow these steps:

1. Choose **Settings...** from the Xcode menu to display the Settings window.
2. In the Settings window, click **Themes** and choose **Presentation (Light)** to make the text larger and easier to read:

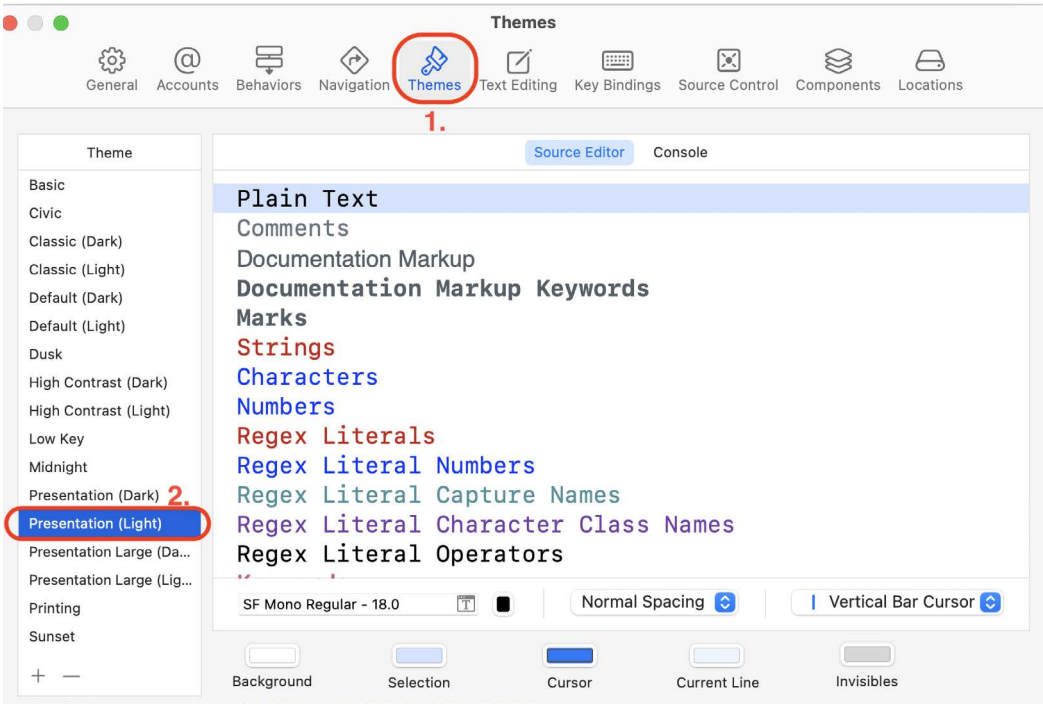


Figure 2.5: Xcode settings window with the Themes pane selected

3. Close the Settings window to return to the playground. Note that the text in the playground is larger than before. You can also try the other themes if you wish.

Now that you've customized the fonts and colors to your liking, let's see how to run playground code in the next section.

Running playground code

Your playground already has an instruction in it. To execute the instruction, follow these steps:

1. Click the Run button to the left of the instruction. After a few seconds, you will see `"Hello, playground"` displayed in the Results area:

Figure 2.6: Playground showing "Hello, playground" in the Results area

You can also use the Run/Stop button in the bottom-left corner or use the keyboard shortcut *Command* + *Shift* + *Return* to run all the code in your playground.

2. To prepare the playground for use in the remainder of this chapter, delete the `var greeting = "Hello, playground"` instruction from the playground. As you go along, type the code shown in this chapter into the playground, and click the Run button to the left of the last line to run it.

Let's dive into the simple data types used in Swift in the next section.

Exploring data types

All programming languages can store numbers, words, and logic states, and Swift is no different. Even if you're an experienced programmer, you may find that Swift represents these values differently from other languages that you may be familiar with.

For more information on data types, visit `https://docs.swift.org/swift-book/documentation/the-swift-programming-language/thebasics`.

Let's walk through the Swift versions of **integers**, **floating-point numbers**, **strings**, and **Booleans** in the next sections.