Getting Started with Xcode

Prachi Pai Asnodkar, Xcode Engineer
Holly Borla, Xcode Engineer
Honza Dvorsky, Xcode Engineer
Creating a New Project
Writing and Navigating Source Code
Running and Debugging
Working with Packages and Frameworks
Testing and Distribution
Creating a New Project

Prachi Pai Asnodkar, Xcode Engineer
// AppDelegate.swift
// Mind

import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplicationLaunchOptionsKey : Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:.
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {  
    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole:  
    }
Project Editor
Project Editor
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UI Scene Session Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UISceneSessionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
Inspector
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sceneMode:)
    }
}
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UIScreenSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UISceneSession.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication,
                      didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication,
                     configurationForConnecting connectingSceneSession: UISceneSessionConfiguration, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
// AppDelegate.swift
// Mind

import UIKit

UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UISceneSessionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground.
    }

    // MARK: UI(Scene) Session Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

  func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    return true
  }

  func applicationWillTerminate(_ application: UIApplication) {
    // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
  }

  // MARK: UIApplication Lifecycle

  func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
    // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
    return UISceneConfiguration(name: "Default Configuration", sessionRole:)
  }
}
// AppDelegate.swift
// Mind

import UIKit

@UIApplicationMain class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UISceneSessionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

@UIApplicationMain class AppDelegate: UIResponder, UIApplicationDelegate {
   func application(_ application: UIApplication,
                   didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
      // Override point for customization after application launch.
      return true
   }

   func applicationWillTerminate(_ application: UIApplication) {
      // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
   }

   // MARK: UISceneSession Lifecycle

   func application(_ application: UIApplication,
                   configurationForConnecting connectingSceneSession: UISceneSession, options: UISceneOptions) -> UISceneConfiguration {
      // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
      return UISceneConfiguration(name: "Default Configuration", sessionRole:
   }
import UIKit

@UIApplicationMain class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole:)
    }
}
import UIKit

UIApplicationMain

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UI Scene Session Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
// AppDelegate.swift
// Mind

import UIKit

@UIApplicationMain class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UIInterfaceOrientation Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UISceneSessionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created.
        // Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole:)
    }
}
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        // Use completion handler to indicate that otherSceneSession has been chosen.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: nil)
    }
}
import UIKit

@UIApplicationMain
class AppDelegate: UIResponder, UIApplicationDelegate {

    func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
        // Override point for customization after application launch.
        return true
    }

    func applicationWillTerminate(_ application: UIApplication) {
        // Called when the application is about to terminate. Save data if appropriate. See also applicationDidEnterBackground:
    }

    // MARK: UISceneSession Lifecycle

    func application(_ application: UIApplication, configurationForConnecting connectingSceneSession: UISceneSession, options: UIScene.ConnectionOptions) -> UISceneConfiguration {
        // Called when a new scene session is being created. Use this method to select a configuration to create the new scene with.
        return UISceneConfiguration(name: "Default Configuration", sessionRole: .unspecified)
    }
}
Welcome to Xcode
Demo
Summary

Creating a project using Xcode
Summary

Creating a project using Xcode

Choosing a template
Summary

Creating a project using Xcode
Choosing a template
Adding an app icon
Summary

Creating a project using Xcode
Choosing a template
Adding an app icon
Using source control
Summary

Creating a project using Xcode

Choosing a template

Adding an app icon

Using source control
Writing and Navigating Source Code

Holly Borla, Xcode Engineer
// MeditationController.swift
// Mind

import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {

    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization

let healthStore = HealthStoreFactory.makeHealthStore()
init(healthStore: HealthStore = healthStore) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
}
class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 35
        case thirtySeconds = 60
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return
            MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration
init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
}
// MeditationController.swift
// Mind

import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 35
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }

    // MARK: - Initialization
    let healthStore: HealthStore
    var duration: Duration

    init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
        self.healthStore = healthStore
        self.duration = .threeMinutes
    }
/// MeditationController.swift
/// Mind

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return .init(rawValue: .timeFormatter.string(from: round(value)))
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
}
import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
  enum Duration: TimeInterval, CaseIterable, Hashable {
    case fiveSeconds = 5
    case fifteenSeconds = 15
    case thirtySeconds = 30
    case oneMinute = 60
    case threeMinutes = 180
    case fiveMinutes = 300
    case tenMinutes = 600
    case fifteenMinutes = 900
  }

  var formattedDuration: String {
    return
      MeditationController.timeFormatter
        .string(from: rawValue)!
  }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
  self.healthStore = healthStore
  self.duration = .threeMinutes
}
Jump Bar

```swift
// MeditationController.swift
import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 35
        case thirtySeconds = 60
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
}
```
import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
}
```swift
// MeditationController.swift
// Mind

import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }

    // MARK: - Initialization
    let healthStore: HealthStore
    var duration: Duration

    init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
        self.healthStore = healthStore
        self.duration = .threeMinutes
    }
}
```
class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return
            MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return 
        MeditationController.timeFormatter
        .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
Jump Bar
```swift
import Foundation
import SwiftUI
import Combine

class MeditationController: BindableObject {
    enum Duration: TimeInterval, CaseIterable, Hashable {
        case fiveSeconds = 5
        case fifteenSeconds = 15
        case thirtySeconds = 30
        case oneMinute = 60
        case threeMinutes = 180
        case fiveMinutes = 300
        case tenMinutes = 600
        case fifteenMinutes = 900
    }

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration

init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
```
class MeditationController: BindableObject {
enum Duration: TimeInterval, CaseIterable, Hashable {
    case fiveSeconds = 5
    case fifteenSeconds = 15
    case thirtySeconds = 30
    case oneMinute = 60
    case threeMinutes = 180
    case fiveMinutes = 300
    case tenMinutes = 600
    case fifteenMinutes = 900

    var formattedDuration: String {
        return MeditationController.timeFormatter
            .string(from: rawValue)!
    }
}

// MARK: - Initialization
let healthStore: HealthStore
var duration: Duration
init(healthStore: HealthStore = HealthStoreFactory.makeHealthStore()) {
    self.healthStore = healthStore
    self.duration = .threeMinutes
}
A Swift Tour

Tradition suggests that the first program in a new language should print the words “Hello, world!” on the screen. In Swift, this can be done in a single line:

```
1. print("Hello, world!")
2. // Prints "Hello, world!"
```

If you have written code in C or Objective-C, this syntax looks familiar to you—in Swift, this line of code is a complete program. You don’t need to import a separate library for functionality like input/output or string handling. Code written at global scope is used as the entry point for the program, so you don’t need a `main()` function. You also don’t need to write semicolons at the end of every statement.

This tour gives you enough information to start writing code in Swift by showing you how to accomplish a variety of programming tasks. Don’t worry if you don’t understand something—everything introduced in this tour is explained in detail in the rest of this book.

Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn’t need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```
1. var myVariable = 42
```
A Swift Tour

Tradition suggests that the first program in a new language should print the words “Hello, world!” on the screen. In Swift, this can be done in a single line:

```swift
print("Hello, world!")
// Prints "Hello, world!"
```

If you have written code in C or Objective-C, this syntax looks familiar to you—in Swift, this line of code is a complete program. You don’t need to import a separate library for functionality like input/output or string handling. Code written at global scope is used as the entry point for the program, so you don’t need a `main()` function. You also don’t need to write semicolons at the end of every statement.

This tour gives you enough information to start writing code in Swift by showing you how to accomplish a variety of programming tasks. Don’t worry if you don’t understand something—everything introduced in this tour is explained in detail in the rest of this book.

**NOTE**

For the best experience, open this chapter as a playground in Xcode. Playgrounds allow you to edit the code listings and see the result immediately.

[Download Playground](#)

Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn’t need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```swift
var myVariable = 42
```
A Swift Tour

Tradition suggests that the first program in a new language should print the words “Hello, world!” on the screen. In Swift, this can be done in a single line:

```swift
1. print(“Hello, world!”)
2. // Prints “Hello, world!”
```

If you have written code in C or Objective-C, this syntax looks familiar to you—in Swift, this line of code is a complete program. You don’t need to import a separate library for functionality like input/output or string handling. Code written at global scope is used as the entry point for the program, so you don’t need a `main()` function. You also don’t need to write semicolons at the end of every statement.

This tour gives you enough information to start writing code in Swift by showing you how to accomplish a variety of programming tasks. Don’t worry if you don’t understand something—this chapter is just a brief introduction to the rest of this book.

For the best experience, open this chapter as a playground in Xcode. Playgrounds allow you to edit the code listings and see the result immediately.

Download Playground

Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn’t need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```swift
1. var myVariable = 42
```
A Swift Tour

Tradition suggests that the first program in a new language should print the words “Hello, world!” on the screen. In Swift, this can be done in a single line:

```swift
print("Hello, world!")
// Prints "Hello, world!"
```

If you have written code in C or Objective-C, this syntax looks familiar to you—in Swift, this line of code is a complete program. You don’t need to import a separate library for functionality like input/output or string handling. Code written at global scope is used as the entry point for the program, so you don’t need a `main()` function. You also don’t need to write semicolons at the end of every statement.

This tour gives you enough information to start writing code in Swift by showing you how to accomplish a variety of programming tasks. Don’t worry if you don’t understand something—everything introduced in this tour is explained in detail in the rest of this book.

**NOTE**

For the best experience, open this chapter as a playground in Xcode. Playgrounds allow you to edit the code listings and see the result immediately.

Download Playground

Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn’t need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```swift
var myVariable = 42
```
A Swift Tour

Tradition suggests that the first program in a new language should print the words “Hello, world!” on the screen. In Swift, this can be done in a single line:

```swift
1. print("Hello, world!")
2. // Prints "Hello, world!"
```

If you have written code in C or Objective-C, this syntax looks familiar to you—in Swift, this line of code is a complete program. You don’t need to import a separate library for functionality like input/output or string handling. Code written at global scope is used as the entry point for the program, so you don’t need a `main()` function. You also don’t need to write semicolons at the end of every statement.

Note: For the best experience, open this chapter as a playground in Xcode. Playgrounds allow you to edit the code listings and see the result immediately.

Simple Values

Use `let` to make a constant and `var` to make a variable. The value of a constant doesn’t need to be known at compile time, but you must assign it a value exactly once. This means you can use constants to name a value that you determine once but use in many places.

```swift
1. var myVariable = 42
```
Demo
Summary
Summary

Interacting with the Canvas
Summary

Interacting with the Canvas

Using code completion
Summary

Interacting with the Canvas

Using code completion

Editing structures from the Action Menu
Summary

Interacting with the Canvas

Using code completion

Editing structures from the Action Menu

Using Fix-Its
Summary

Interacting with the Canvas

Using code completion

Editing structures from the Action Menu

Using Fix-Its

What’s New in Swift

Introducing SwiftUI: Building Your First App
Running and Debugging
Demo
Summary
Running and debugging with the Simulator
Summary

Running and debugging with the Simulator

Running on a device
Summary

Running and debugging with the Simulator

Running on a device

Adding a Swift Package
Summary

Running and debugging with the Simulator

Running on a device

Adding a Swift Package

---

Advanced Debugging with Xcode and LLDB

WWDC 2018

Adopting Swift Packages in Xcode

WWDC 2019
Working with Packages and Frameworks

Holly Borla, Xcode Engineer
Demo
API Design Guidelines

Table of Contents

- Fundamentals
- Naming
  - Promote Clear Usage
  - Strive for Fluent Usage
  - Use Terminology Well
- Conventions
  - General Conventions
  - Parameters
  - Argument Labels
- Special Instructions

Fundamentals

- **Clarity at the point of use** is your most important goal. Entities such as methods and properties are declared only once but used repeatedly. Design APIs to make those uses clear and concise. When evaluating a design, reading a declaration is seldom sufficient; always examine a use case to make sure it looks clear in context.
API Design Guidelines

Table of Contents

• Fundamentals
  • Clarity at the point of use is your most important goal. Entities such as methods and properties are declared only once but used repeatedly. Design APIs to make those uses clear and concise. When evaluating a design, reading a declaration is seldom sufficient; always examine a use case to make sure it looks clear in context.
Summary
Summary

Importing a Swift Package
Summary

Importing a Swift Package

Creating a framework
Summary

Importing a Swift Package
Creating a framework
Reading Developer Documentation
Summary

Importing a Swift Package
Creating a framework
Reading Developer Documentation
Writing documentation comments
Testing and Distribution

Honza Dvorsky, Xcode Engineer
Tests
Tests

Unit
Summary
Summary

Writing unit and UI tests
Summary

Writing unit and UI tests

Editing Test Plans
Summary

Writing unit and UI tests
Editing Test Plans
Viewing the Test Report
Summary

Writing unit and UI tests

Editing Test Plans

Viewing the Test Report

Archiving and uploading using the Organizer
Summary

Writing unit and UI tests

Editing Test Plans

Viewing the Test Report

Archiving and uploading using the Organizer

Testing in Xcode

Thursday, 11:00AM

What’s New in Signing for Xcode and Xcode Server

WWDC 2017
Creating a New Project
Writing and Navigating Source Code
Running and Debugging
Working with Packages and Frameworks
Testing and Distribution
More Information

developer.apple.com/wwdc19/404