10-Day iOS Development Bootcamp

Swift Basics to App Publishing

Ravi Shankar

iOS/Swift/SwiftUI Trainer and Developer

- 25+ years in tech industry
- 9 years focussed on iOS/Swift
- <> Freelance iOS Developer
- Co-author of Udemy Course

Published Apps Across Apple Platforms

iPhone Mac Apple Watch

Aspiring Indie App Developer

Working towards success in independent app development

10-Day iOS Development Training

<> Swift Fundamentals

- Swift Basics
- Collections
- Functions & Closures
- Object-Oriented Programming

iOS & UI Development

- iOS Core Concepts
- UIKit Essentials
- SwiftUI Fundamentals
- Navigation & Multi-View Apps

Data & Networking

- Data Storage Options
- Core Data
- Networking & API Integration
- Concurrency & Multitasking

Advanced Topics

- Design Patterns (MVC, MVVM)
- Performance Optimization
- Debugging & Profiling
- Testing (Unit & UI)

“From Swift basics to publishing your app, we’ll cover it all!”

Our Training Approach

<> Hands-on Coding

Extensive practical exercises
and real-world projects

Coding Challenges

Regular challenges to test
and reinforce your skills

☑ MCQ Assessments

Comprehensive online testing
of key concepts

Individual App Assessment

Personal project to
demonstrate individual skills

Final App Showcase

Present your group's MVP to demonstrate applied learning

"From interactive learning to collaborative app development - build your iOS skills through practical application!"

Day 1 Agenda

- ✅ Apple Ecosystem
- ✅ Playground Overview
- ✅ Swift Basics
- ✅ Enums
- ✅ Optionals and Guard
- ✅ Closures

Day 2 Agenda

- ✅ Class
- ✅ Struct
- ✅ Protocol
- ✅ Extension
- ✅ Generics

Day 3 Agenda

- ✅ Memory Management
- ✅ Type Casting
- ✅ Error Handling
- ✅ Access Control
- ✅ Introduction to IOS
- ✅ UIKit

Memory Management

weak

- Creates a reference that doesn't increase the reference count
- Automatically set to `nil` when the referenced object is deallocated
- Used to avoid retain cycles, especially in parent-child relationships
- Example: `weak var artifact: CulturalArtifact?`

unown

- Similar to weak, but doesn't change to `nil`
- Use when you're sure the reference will never be nil during its lifetime
- Can cause crashes if used incorrectly (accessing after deallocation)
- Example: `unowned let camera: Photographer`

[weak self]

- Prevents retain cycles in closures that capture `self`
- Allows `self` to be deallocated even if the closure is still alive

Swift Type Casting

Operator	Description	Usage
is	• Used to check if an instance is of an certain type.	If item is Monument {}
as	• Upcasting (casting to a superclass)	let culturalItem = monument as CulturalItem
as?	• Downcast to a subclass • Returns an optional of the target type	If let monument = item as? Monument
as!	• Forces a downcast • if not possible, it triggers a runtime error.	let knownMonument = culturalItems[0] as! Monument

Error Handling

- Defining Errors using enums
- Throwing Errors
- Handling errors with do-catch blocks
- Different variations of the try keyword (try, try?, try!)
- Rethrowing errors
- Using the Result type
- Asynchronous error handling

Error Handling

try and try?

```swift
// Using do-catch to handle errors
do {
    let result = try recognizeMonument(image: "taj_mahal")
    print(result)
} catch SwiftLensError.unrecognizedMonument {
    print("Oops! We couldn't recognize this monument.")
} catch SwiftLensError.networkError {
    print("Oops! Network Error.")
} catch {
    print("An unexpected error occurred: \(error)")
}

// Using try? to handle errors (returns an optional)
print("\nNaina: We can also use try? for a simpler approach.")
if let result = try? recognizeMonument(image: "eiffel_tower") {
    print(result)
} else {
    print("Failed to recognize the monument.")
}
```

Error Handling

Result Type

```swift
// API call function
func fetchIPInfo(completion: @escaping (Result<IPInfo, Error>) -> Void) {
    let apiUrl = URL(string: "http://ip-api.com/json/")!
    let session = URLSession.shared
```

API Error Handling

```swift
fetchIPInfo { result in
    switch result {
    case .success(let ipInfo):
        print("API Response:")
        print("Status: \(ipInfo.status)")
        print("Country: \(ipInfo.country)")
        print("City: \(ipInfo.city)")
        print("ISP: \(ipInfo.isp)")
        print("Latitude: \(ipInfo.lat)")
        print("Longitude: \(ipInfo.lon)")
        // ... print other fields as needed
    case .failure(let error):
        print("Error: \(error.localizedDescription)")
    }
}
```

Swift Access Control

Access Level	Description	Scope
open	Least restrictive; allows access from any source file in any module. Can be subclassed and overridden outside its defining module.	Class only
public	Allows access from any source file in any module. Cannot be subclassed or overridden outside its defining module.	Any declaration
internal	Allows access from any source file within the same module. This is the default level.	Any declaration
fileprivate	Restricts access to the current source file.	Any declaration
private	Most restrictive; limits access to the enclosing declaration and extensions in the same file.	Any declaration

UIKit Storyboard and Xcode Basics

UI Storyboard Basics

Label: Used to display text (e.g., welcome message)

Button: Triggers actions when tapped

@IBOutlet: Connects UI elements to code @IBOutlet weak var welcomeLabel: UILabel!

@IBAction: Connects button taps to code @IBAction func buttonTapped(_ sender: UIButton) { }

Xcode Interface Elements

Project Navigator: Shows project files and structure

Inspector Pane:

- Attributes Inspector: Modify properties of UI elements
- Connection Inspector: Manage outlets and actions

Debug/Console Window: Displays output and errors

Interface Builder: Visual editor for storyboards

Assistant Editor: Split view to see code and storyboard simultaneously

UIKit Storyboard and Xcode Basics

Creating a Basic UI

- Open storyboard in Interface Builder
- Drag and drop a Label and a Button onto the view
- Use Assistant Editor to create outlets and actions
- Implement button action to update label text: @IBAction func buttonTapped(_ sender: UIButton) {
  welcomeLabel.text = "Welcome to UIKit!"
  }

Running and Testing

- Use the play button or Cmd+R to build and run the app
- Test the UI in the simulator
- Check the console for any errors or debug information

Common Errors

IBOutlet not set

Error: "Key-value coding-compliant" error

Cause: IBOutlet connection is missing or incorrect in Interface Builder

Missing ViewController in Assistant Editor

Issue: When the Assistant Editor is open, the ViewController is not shown

Cause: Incorrect file linking or naming conventions

Crash when accessing an unconnected IBOutlet

Error: "Unexpectedly found nil while unwrapping an Optional value"

Cause: Attempting to use an IBOutlet that hasn't been properly connected

Debugging in iOS: Expectations vs. Reality

Expectation
Methodical detective work

Reality
Why is this optional nil?!

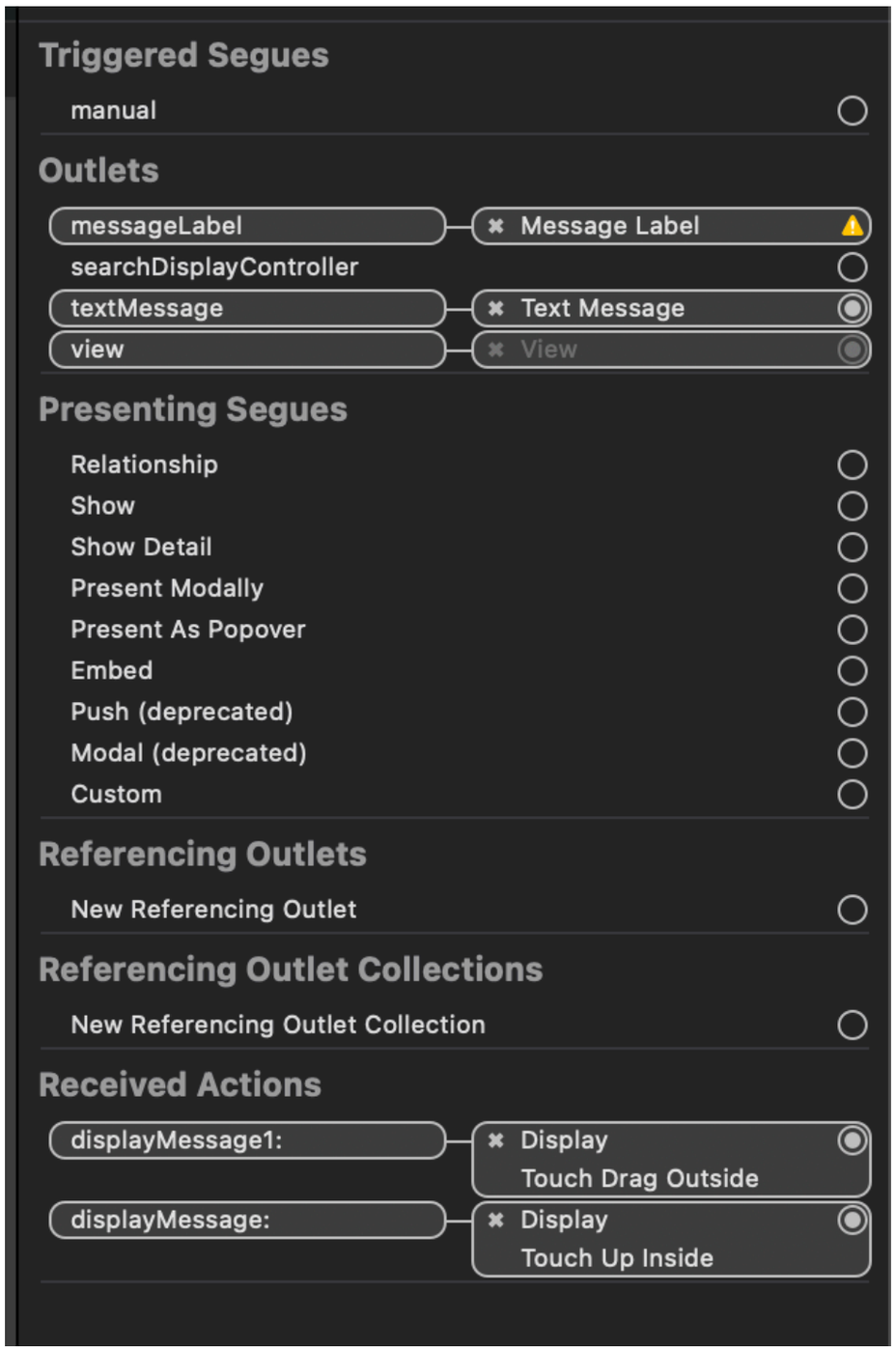
When you spend hours debugging only to realize you forgot to connect an IBOutlet...

Common Errors

IBOutlet not set

Error: "Key-value coding-compliant" error

Cause: IBOutlet connection is missing or incorrect in Interface Builder

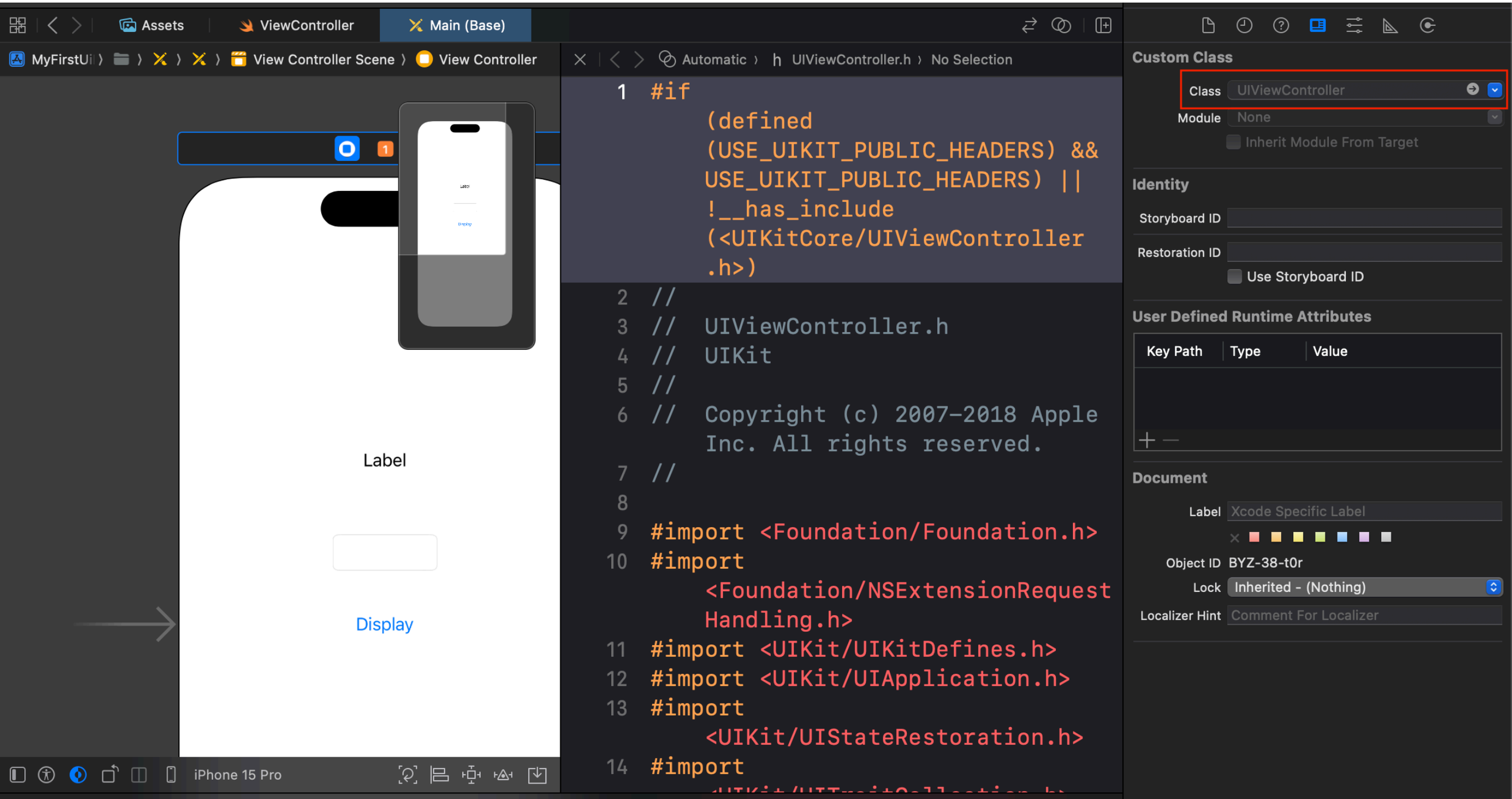


Common Errors

Missing ViewController in Assistant Editor

Issue: When the Assistant Editor is open, the ViewController is not shown

Cause: Incorrect file linking or naming conventions

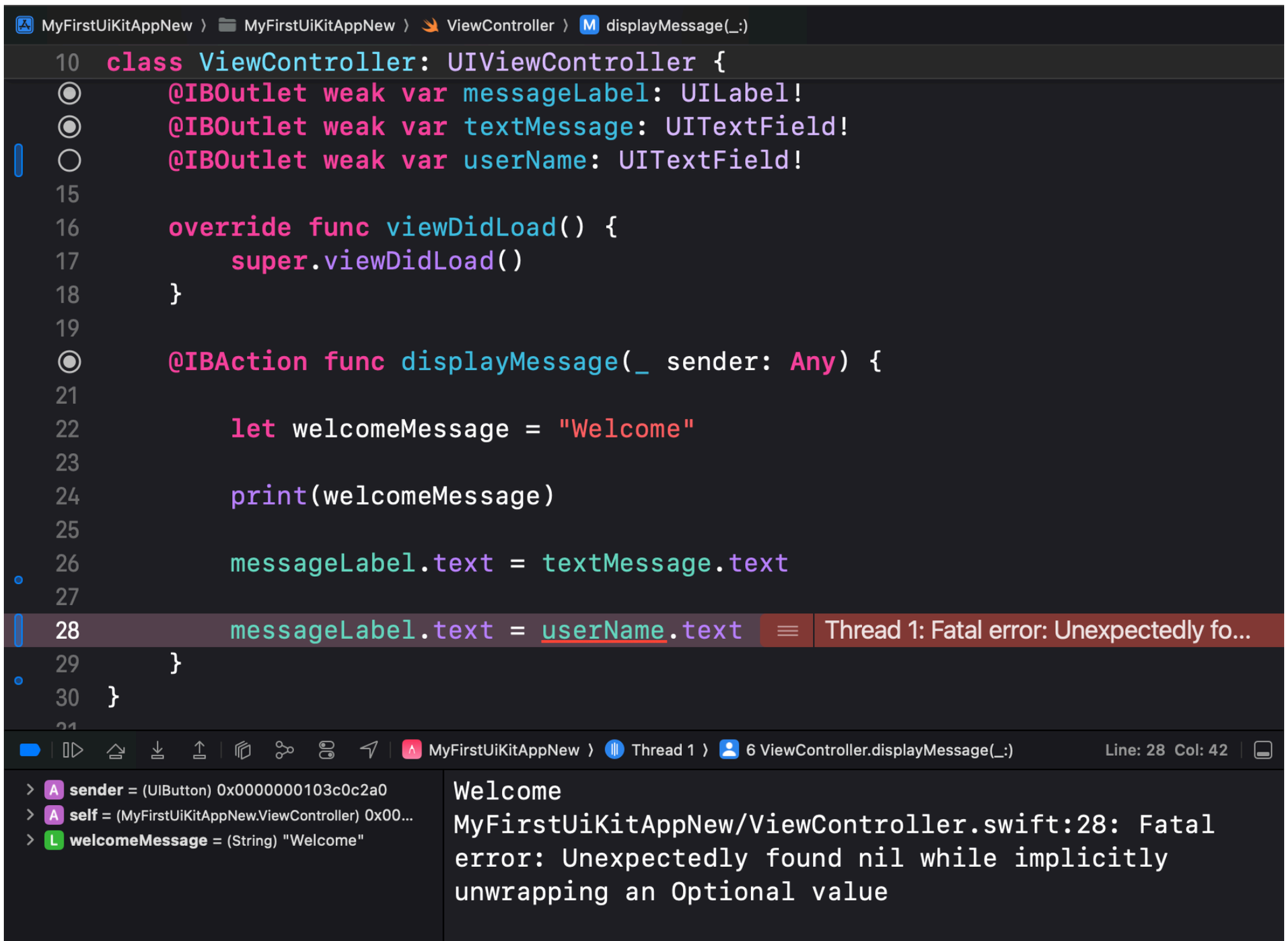


Common Errors

Crash when accessing an unconnected IBOutlet

Error: "Unexpectedly found nil while unwrapping an Optional value"

Cause: Attempting to use an IBOutlet that hasn't been properly connected



Day 4 Agenda

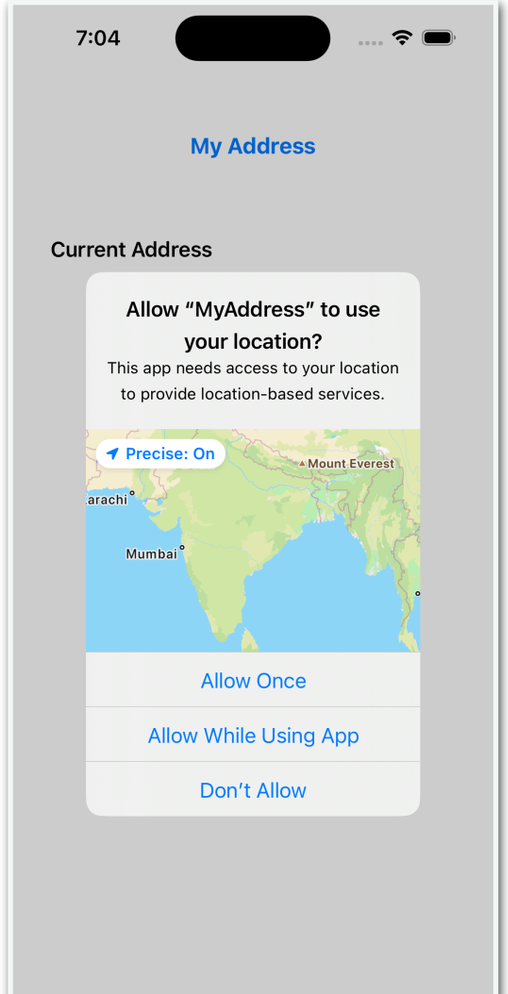
- ✅ UIKit Overview
- ✅ iOS Delegate Pattern
- ✅ Navigation Controller
- ✅ Segues
- ✅ Passing information between View Controllers
- ✅ Auto Layout
- ✅ Stock Trader App (MVC, Custom Delegate)

Day 4 Agenda

- ✅ CLLocationManagerDelegate - MyAddress App

Getting Permission

Information Property List		Dictionary	(3 items)
Application Scene Manifest	⇕	Dictionary	(2 items)
Privacy - Location Always and When In Use Usage D...	⇕	String	This app needs access to your location to provide location-based services.
Privacy - Location When In Use Usage Description	⇕	String	This app needs access to your location to provide location-based services.



Day 4 Agenda

- ✅ CLLocationManagerDelegate - MyAddress App

Request Authorisation & Start Updating Location

```
    override func viewDidLoad() {
        super.viewDidLoad()
        locationManager.delegate = self
        locationManager.desiredAccuracy = kCLLocationAccuracyBest
    }

    override func viewDidAppear(_ animated: Bool) {
        super.viewDidAppear(animated)
    //    locationManager.requestWhenInUseAuthorization()
    }

    @IBAction func getMyAddress(sender: Any) {
        locationManager.requestWhenInUseAuthorization()
        locationManager.startUpdatingLocation()
    }
```

Day 4 Agenda

✅ CLLocationManagerDelegate - MyAddress App

Delegate Methods

```
func locationManager(_ manager: CLLocationManager, didUpdateLocations locations:
    [CLLocation]) {
    guard let location = locations.last else { return }

    // Stop updating location to conserve battery
    locationManager.stopUpdatingLocation()

    // Get address from location
    getAddressFromLocation(location)
}
```

```
func locationManager(_ manager: CLLocationManager, didFailWithError error: any Error) {
    print(error.localizedDescription)
}
```

Day 4 Agenda

✅ CLLocationManagerDelegate - MyAddress App

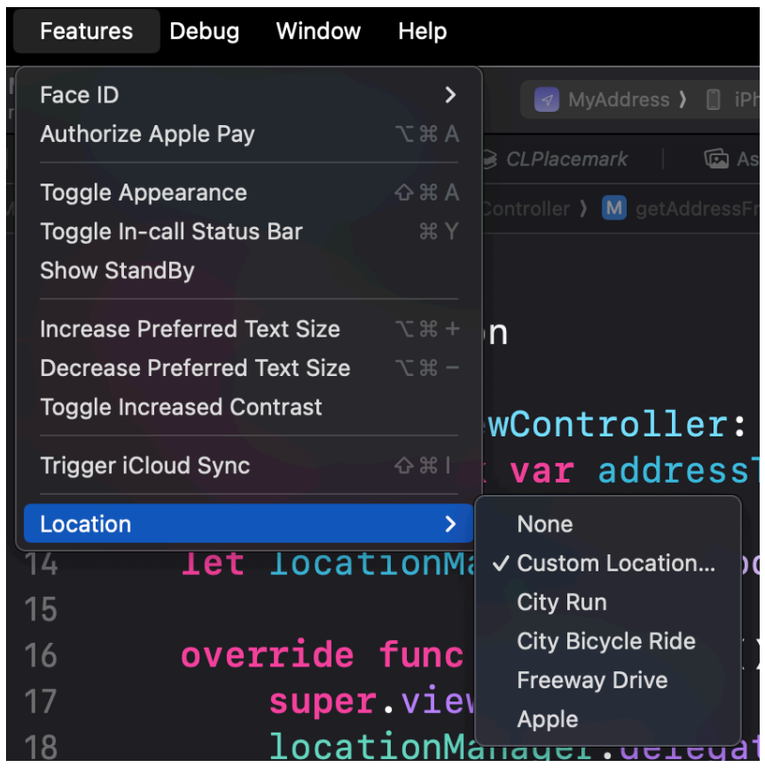
Reverse Geocode

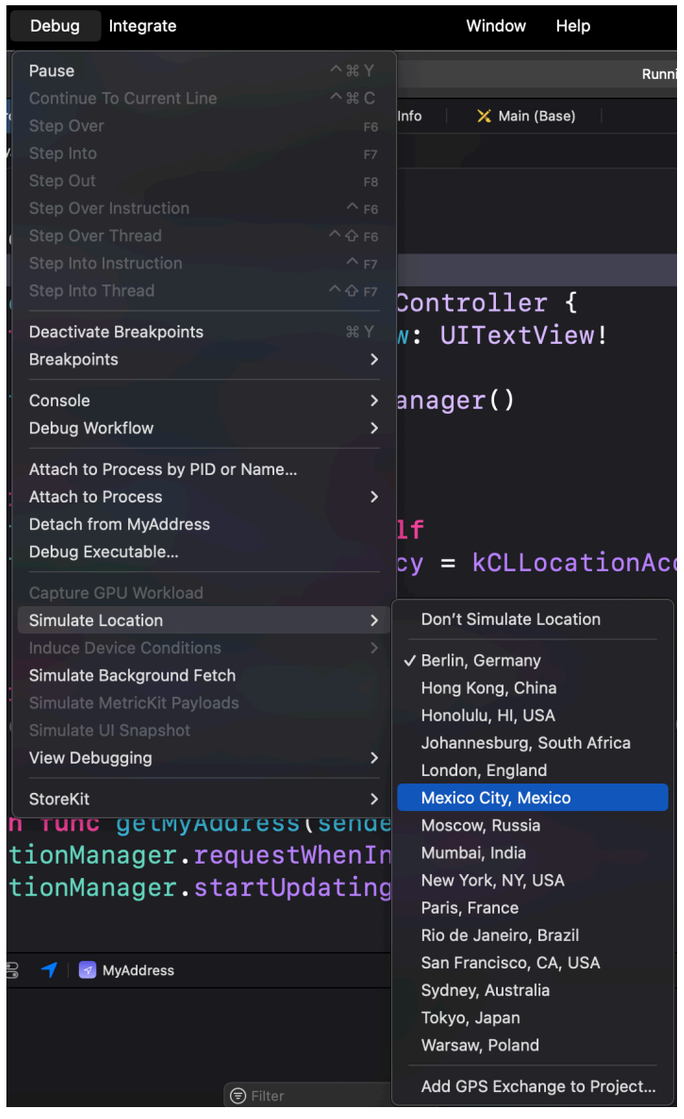
```
func getAddressFromLocation(_ location: CLLocation) {
    let geocoder = CLGeocoder()

    geocoder.reverseGeocodeLocation(location) { (placemarks, error) in
        if let error = error {
            print("Reverse geocoding error: \(error.localizedDescription)")
            return
        }

        guard let placemark = placemarks?.first else {
            print("No placemark found")
            return
        }
```

Day 4 Agenda

- ✅ CLLocationManagerDelegate - MyAddress App

Simulate Location





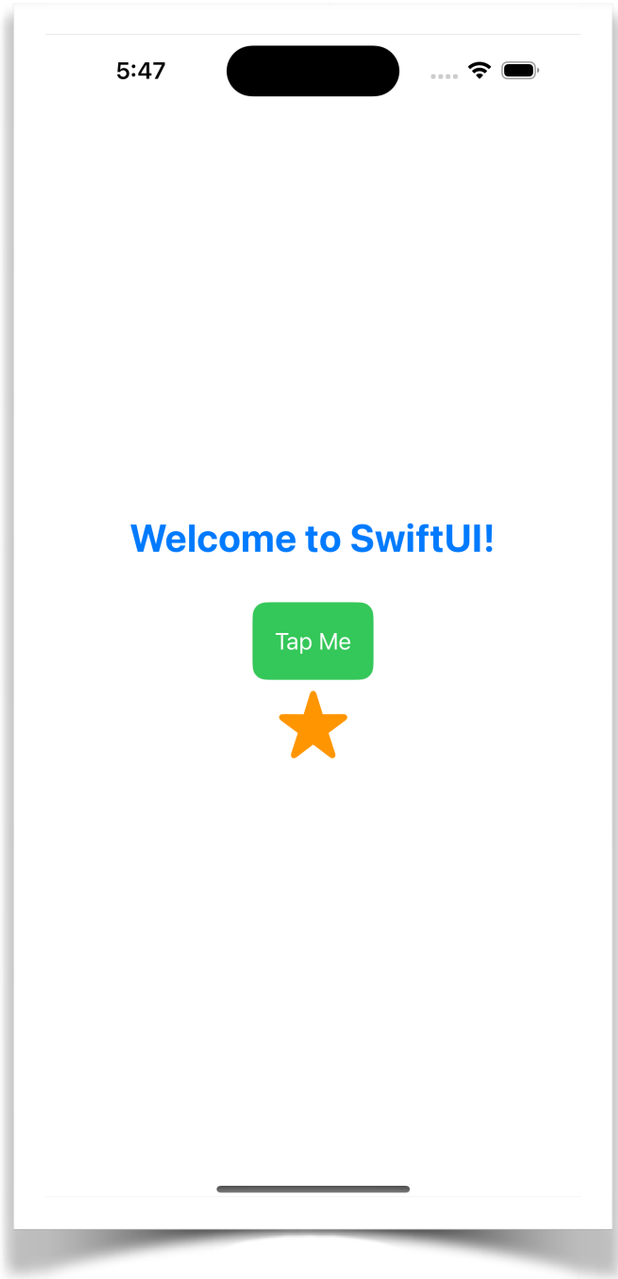
Day 5 Agenda

- ✅ SwiftUI
- ✅ Swift UI State management
- ✅ Interoperability - Objective-C & Swift
- ✅ Interoperability - UIKit & SwiftUI

Day 5 Agenda

✅ SwiftUI - My First SwiftUI App

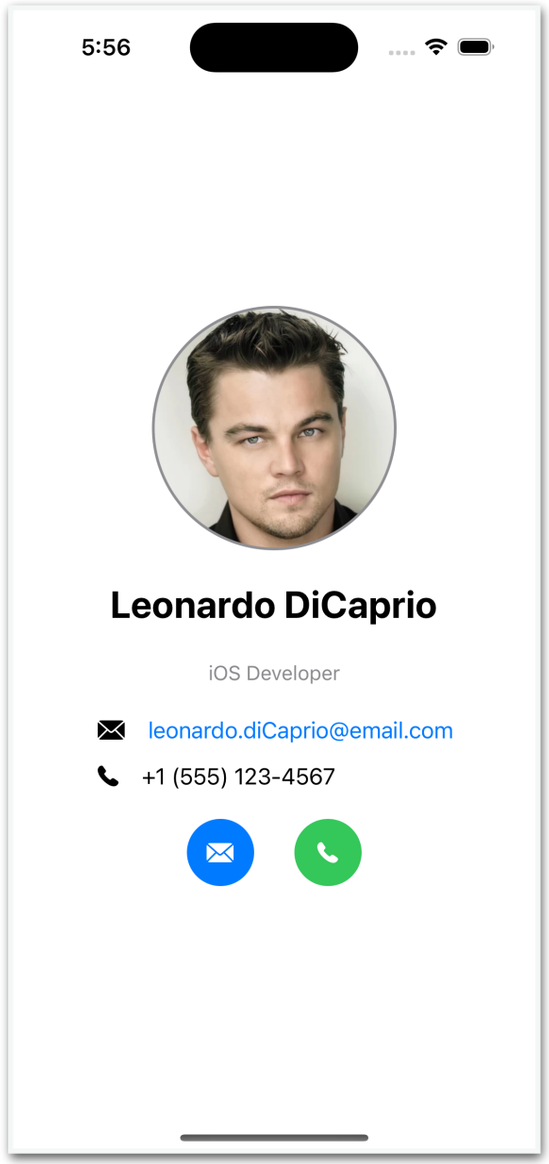
- SwiftUI Controls
  - Text
  - Button
  - Image
- Modifiers
- @State variable
- Canvas



Day 5 Agenda

✅ SwiftUI - Personal Business Card

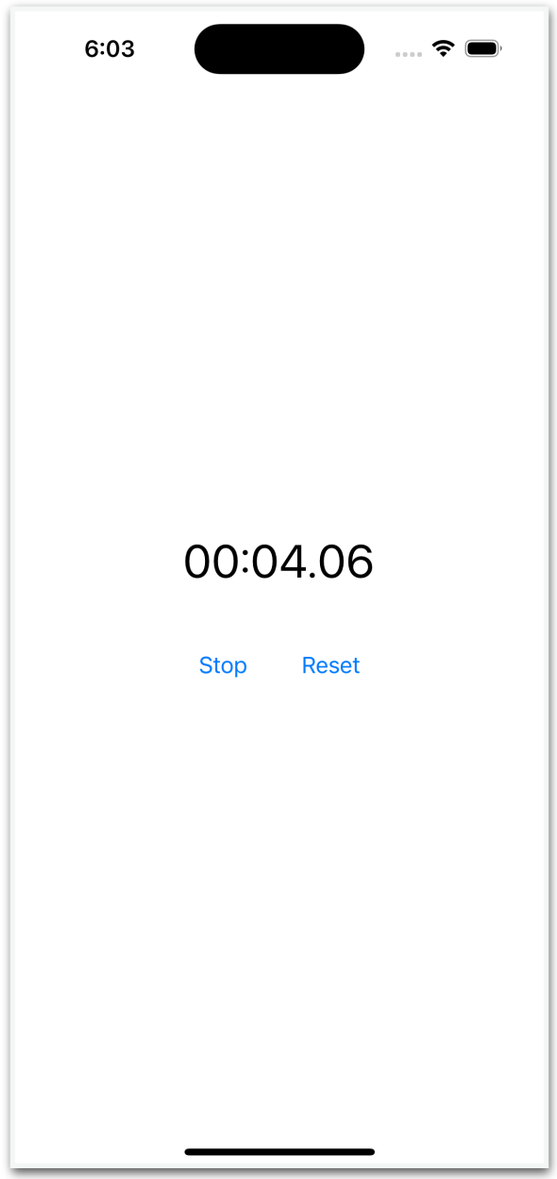
- SwiftUI Controls
  - Text
  - Button
  - Image
  - VStack
  - HStack
- Alignments, Spacing
- Preview (dark mode)



Day 5 Agenda

✅ SwiftUI

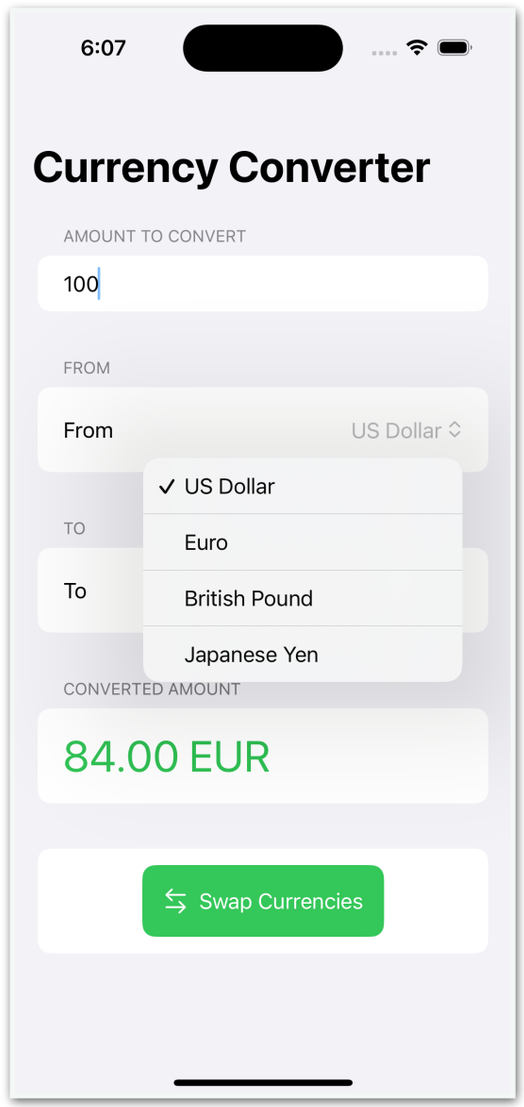
- Simple Stop Watch — Challenge



Day 5 Agenda

✅ SwiftUI - Currency Converter App

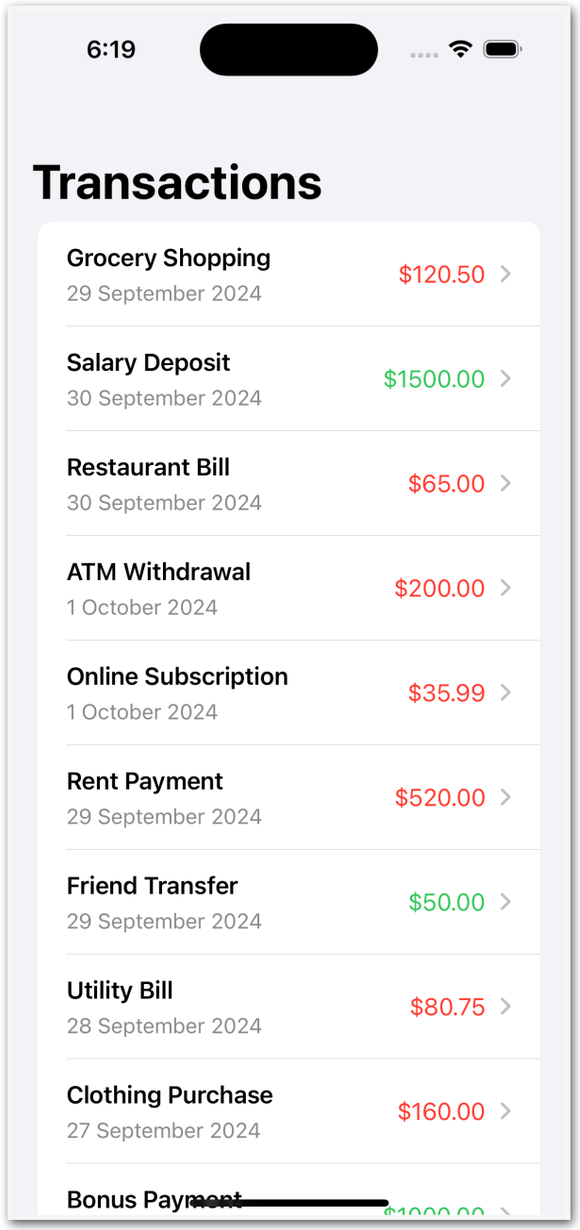
- Forms
- Picker
- Custom Button
- struct (Model)

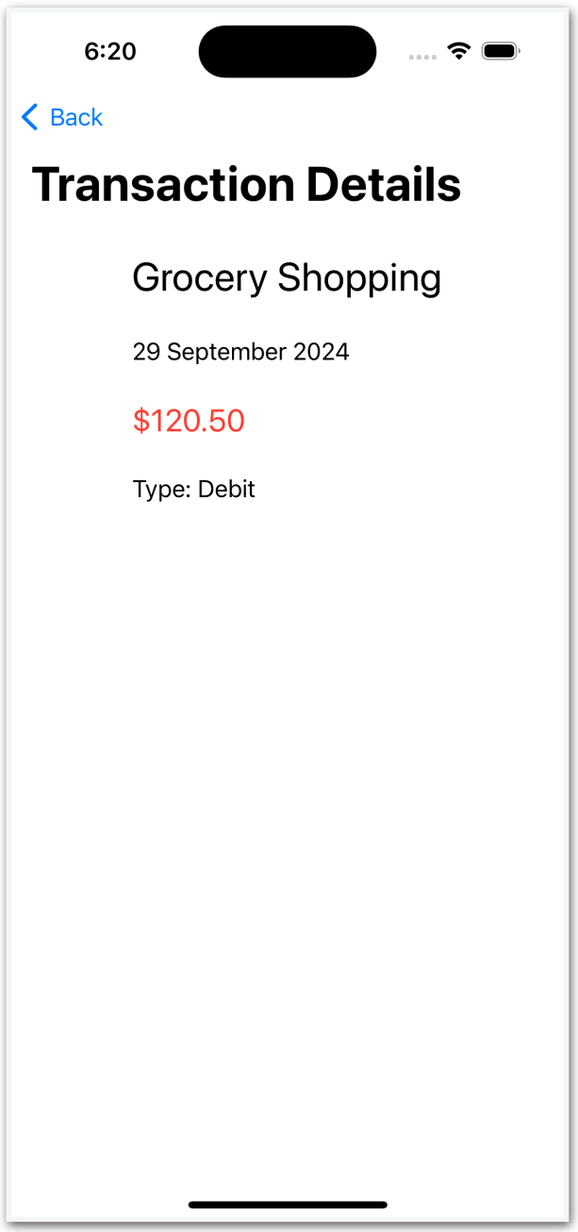


Day 5 Agenda

SwiftUI - Transactions List App

- List
- Navigation
- Custom View
- Passing Data
- struct & enum

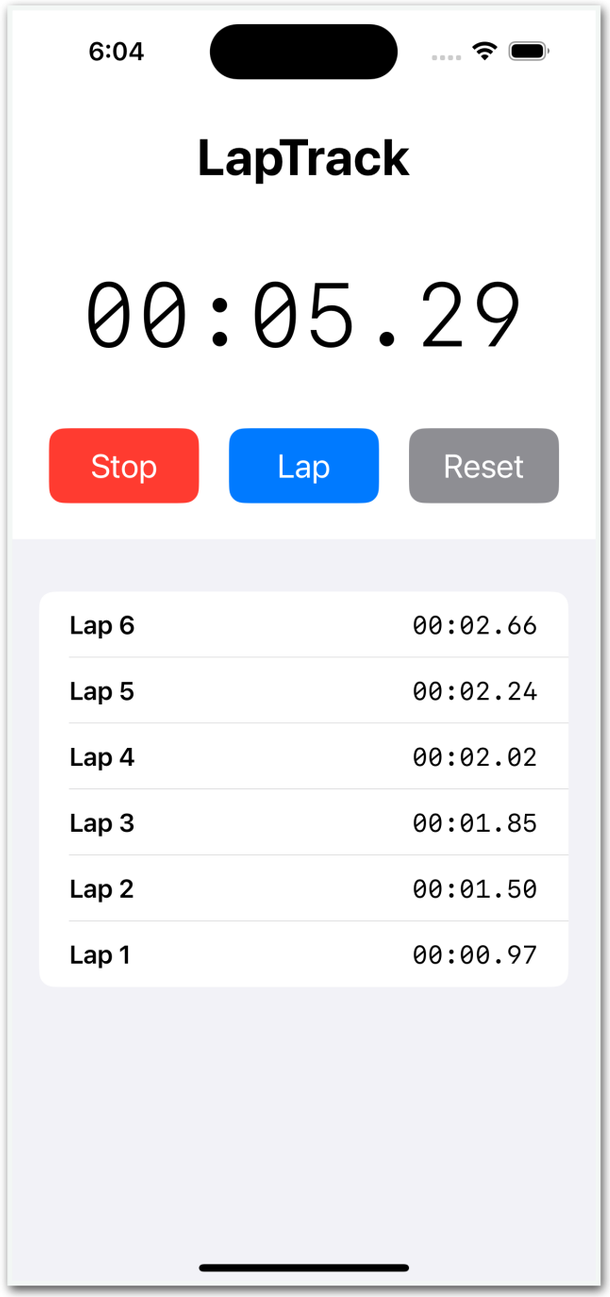




Day 5 Agenda

✅ SwiftUI

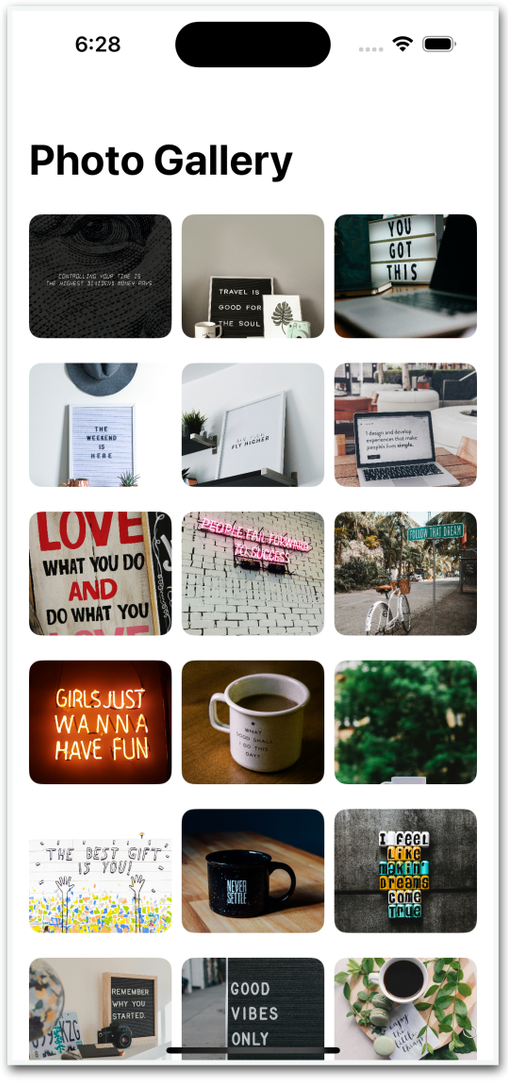
- Lap track - Challenge



Day 5 Agenda

✅ SwiftUI - Photo Gallery App

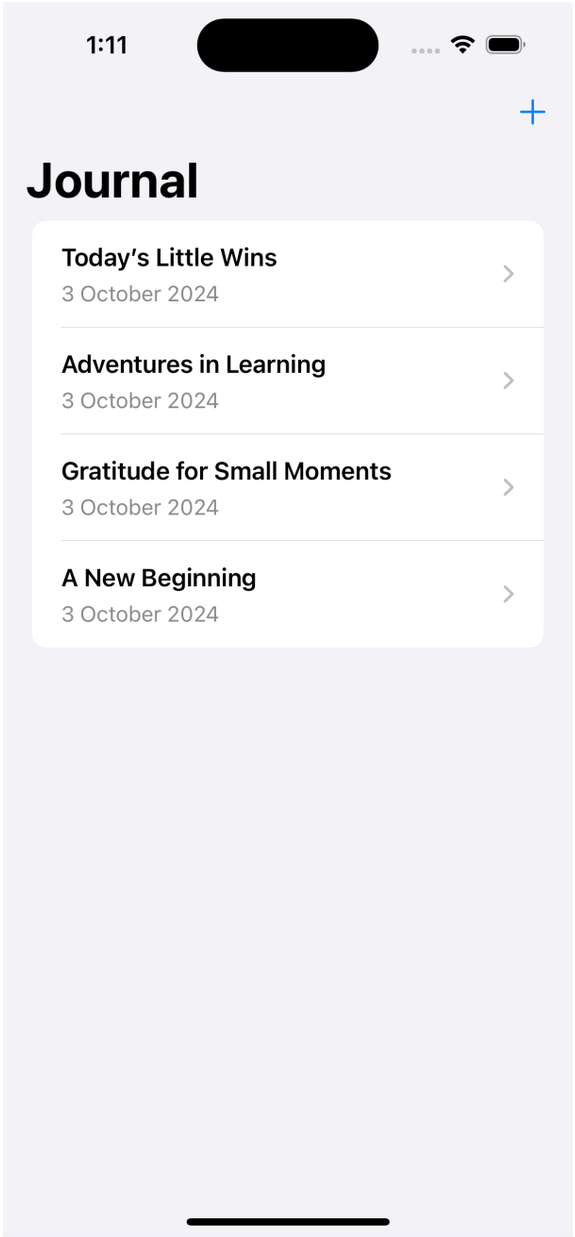
- ScrollView
- LazyVGrid
- Magnification Gesture
- Tap Gesture
- @Binding



Day 5 Agenda

SwiftUI & Core Data - Journal Entry App

- Personal Journal App with CRUD functionality using Core Data.



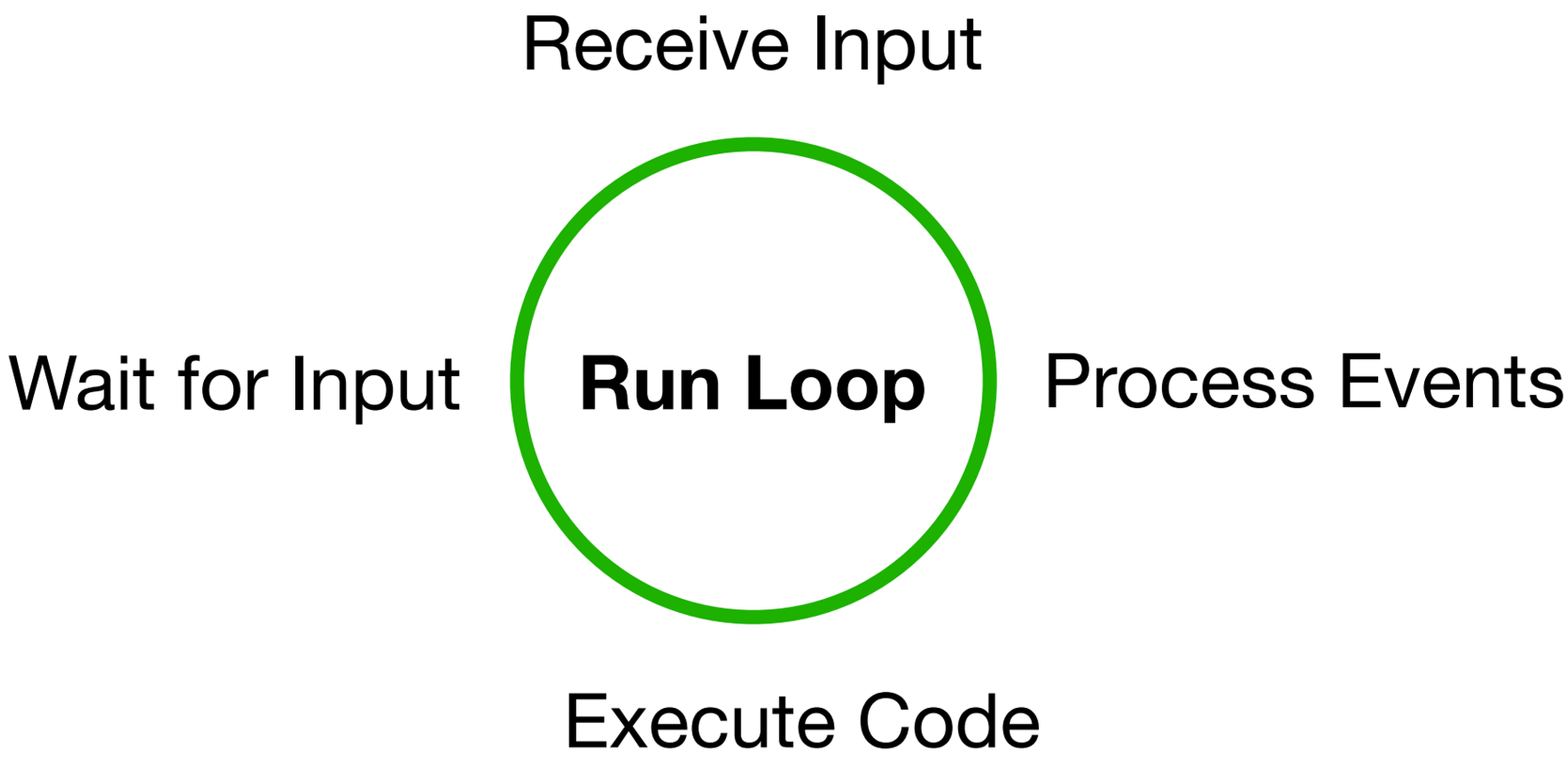
What is Current Runloop?

Creating a Timer

```
class func scheduledTimer(withTimeInterval: TimeInterval, repeats: Bool, block:
(Timer) -> Void) -> Timer
```

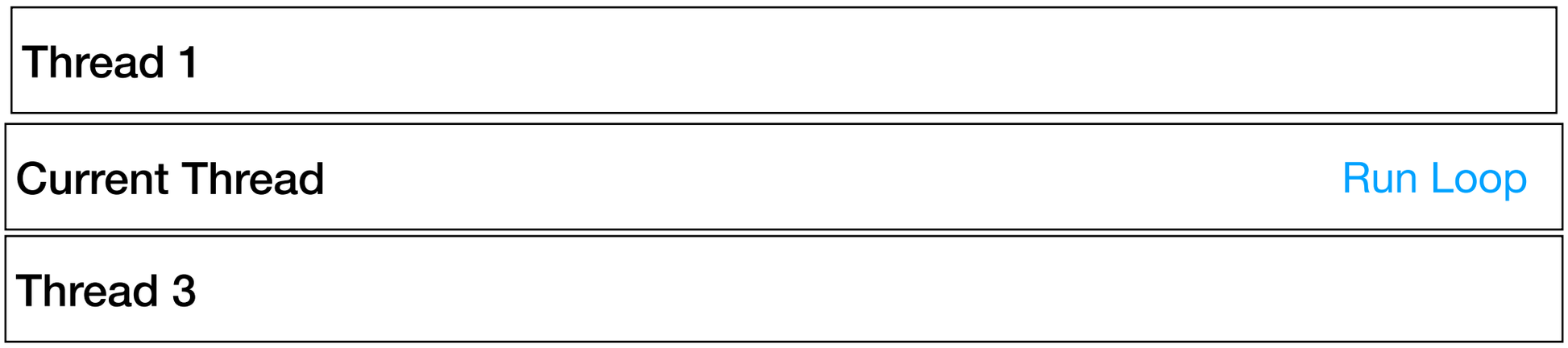
Creates a timer and schedules it on the current run loop in the default mode.

What is Current Runloop?



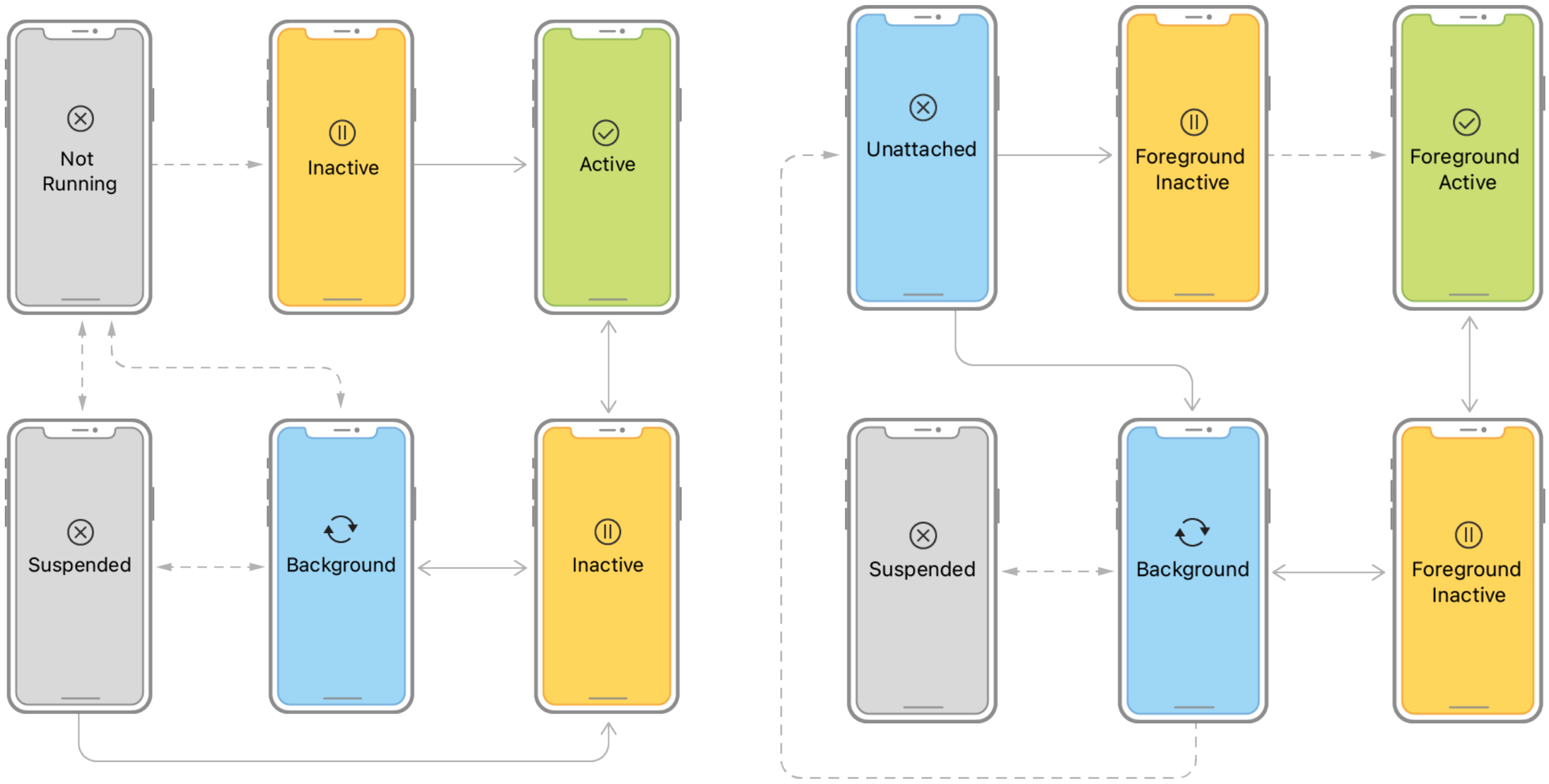
- Run loops manage and coordinate events and inputs in an app
- They run on a single thread, typically the main thread

Current Runloop



- The current run loop is associated with the thread where code is currently executing
- Usually refers to the main thread's run loop
- Accessed via RunLoop.current in Swift

App Life Cycle

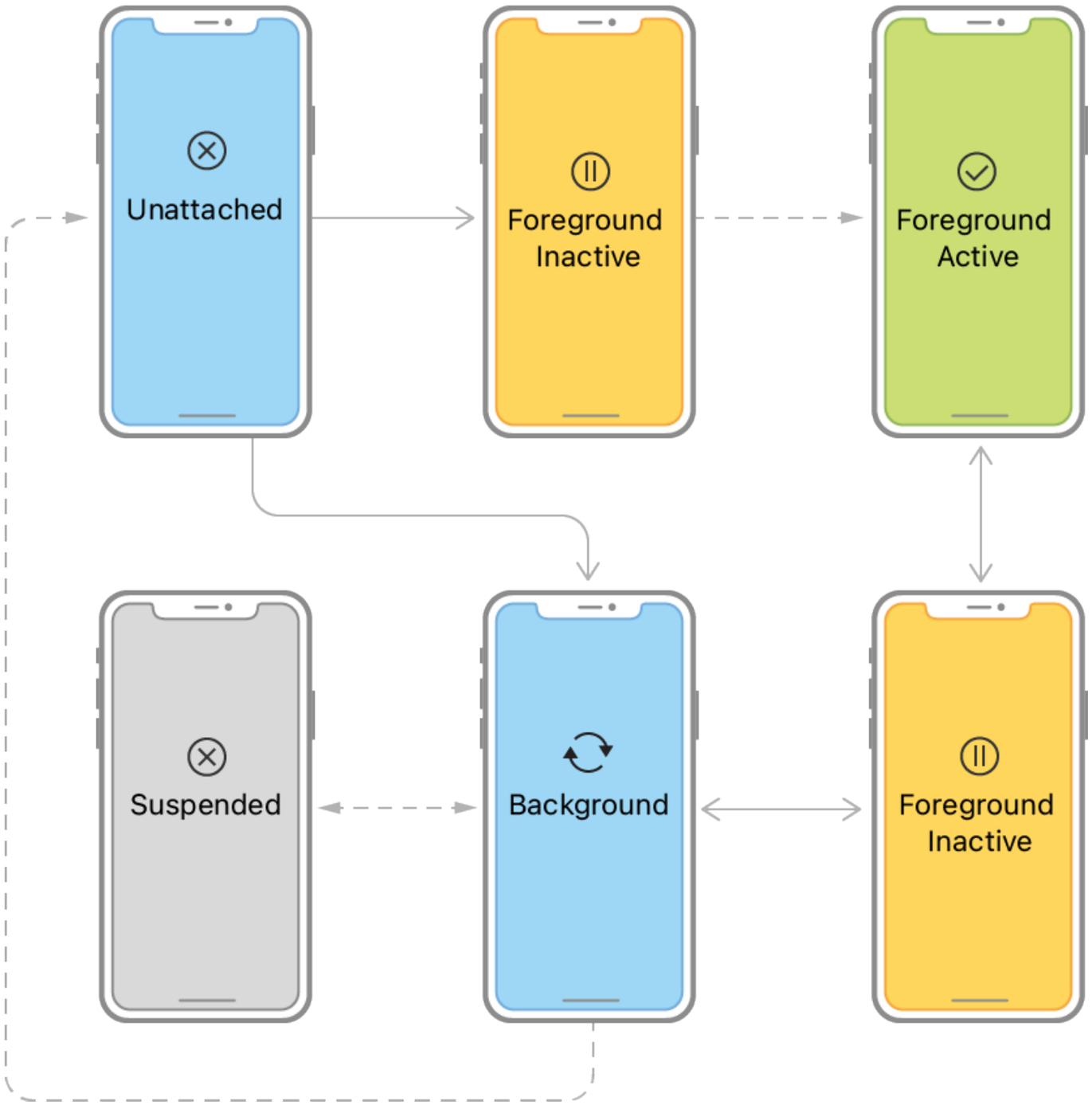


Before iOS 13

After iOS 13

- Solid Line - User initiated action
- Dashed line - User initiated action

App Life Cycle



AppDelegate vs SceneDelegate

AppDelegate Methods	SceneDelegate Methods	Description
application(_:didFinishLaunchingWithOptions:)		Called when the app launches. Used for app-wide setup
applicationWillTerminate(_:)		Called when the app is about to terminate.
	scene(_:willConnectTo:options:)	Called when a new scene session is being created. Used for scene-specific
	sceneDidDisconnect(_:)	Called when a scene is being released by the system.
applicationDidBecomeActive(_:)	sceneDidBecomeActive(_:)	Called when the app/scene becomes active.
applicationWillResignActive(_:)	sceneWillResignActive(_:)	Called when the app/scene is about to become inactive.
applicationDidEnterBackground(_:)	sceneDidEnterBackground(_:)	Called when the app/scene enters the background.
applicationWillEnterForeground(_:)	sceneWillEnterForeground(_:)	Called when the app/scene is about to enter the foreground.
application(_:handleEventsForBackgroundURLSession:completionHandler:)	scene(_:handleEventsForBackgroundURLSession:completionHandler:)	Handles events for background URL sessions
application(_:configurationForConnecting:options:)`		Configures and returns a UISceneConfiguration object.
application(_:didDiscardSceneSessions:)		Called when the user discards a scene session.

SwiftUI using AppDelegate methods

```
@main
struct AppLifeCycleSwiftUiApp: App {
    @UIApplicationDelegateAdaptor(AppDelegate.self) var appDelegate
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
    }
}

class AppDelegate: NSObject, UIApplicationDelegate {
    func applicationDidFinishLaunching(_ application: UIApplication) {

    }

    func application(_ application: UIApplication, didFinishLaunchingWithOptions
        launchOptions: [UIApplication.LaunchOptionsKey : Any]? = nil) -> Bool {
        return true
    }

    func applicationWillResignActive(_ application: UIApplication) {

    }
}
```

LifeCycle states in SwiftUI

```swift
@main
struct AppLifeCycleSwiftUiApp: App {
    @Environment(\.scenePhase) var scenePhase

    @UIApplicationDelegateAdaptor(AppDelegate.self) var appDelegate
    var body: some Scene {
        WindowGroup {
            ContentView()
        }
        .onChange(of: scenePhase) { oldPhase, newPhase in
            switch newPhase {
            case .active:
                print("App is active")
            case .inactive:
                print("App is inactive")
            case .background:
                print("App is in background")
            @unknown default:
                print("Unknown scene phase")
            }
        }
    }
}
```

Real-life use cases for handling app states (.active, .inactive, .background)

App state	Action/Behaviour
.active	Resume game, media playback or ongoing tasks
	Log when the app becomes active for usage analytics.
	Resume or initiate network requests like downloads/uploads.
	Check if data syncing is needed when the app is back.
	Restart heavy tasks or processes that were stopped.
	Unlock the app or remove screen blur (for sensitive apps).
.inactive	Save the current state or data.
	Blur sensitive information or lock the app when not in use.
	Pause media playback, games, or ongoing processes.
.background	Stop GPS tracking, processing, or machine learning tasks.
	Continue syncing or uploading tasks in the background.

What is some view?

```
struct ContentView: View {
    var body: some View {
        VStack {
            Text("Hello, world!")
        }
    }
}
```

Some refers to a opaque type - it is a way to say ‘this will be a view but don’t worry exactly what kind’. The compiler knows the specific type and uses this information to optimise performance.

Is Bitmap supported?

✓ Bitmap - image represented as grid of pixels

- Windows Bitmap - generally not used because of file size
- Other Bitmap based formats - PNG, JPEG, TIFF

✓ List of Supported image formats

- PNG (Portable Network Graphics)
- JPEG (Joint Photographic Experts Groups)
- PDF (Portable Document Format)
- SVG (Scalable Vector Graphics)
- TIFF (Tagged Image File Format)
- GIF (Graphics Interchange Format)

Common Formats - PNG and JPEG

PDF and SVG, vector graphics when you want to scale across different devices

Is Bitmap supported?

```
enum ImageFormat {
    case png
    case jpeg
}
```

```
func convertBitmapTo(format: ImageFormat, image: UIImage,
    compressionQuality: CGFloat = 1.0) -> Data? {

    switch format {
    case .png:
        return image.pngData()
    case .jpeg:
        return image.jpegData(compressionQuality:
            compressionQuality)
    }
}
```

```
// Usage example
if let image = UIImage(named: "myBitmapImage") {
    // Convert to PNG
    if let pngData = convertBitmapTo(format: .png, image:
        image) {
        // Save or use the PNG data
        try? pngData.write(to: URL(fileURLWithPath:
            "/path/to/save/image.png"))
    }

    // Convert to JPEG
    if let jpegData = convertBitmapTo(format: .jpeg,
        image: image, compressionQuality: 0.8) {
        // Save or use the JPEG data
        try? jpegData.write(to: URL(fileURLWithPath:
            "/path/to/save/image.jpg"))
    }
}
```

NavigationStack and Navigation View

- ✅ iOS 13 - NavigationView

- ✅ iOS 16 and above - NavigationStack

```
if #available(iOS 16.0, *) {
    NavigationStack {
        // Content
    }
} else {
    NavigationView {
        // Content
    }
}
```

NavigationView modifiers?

✅ Why navigationTitle and navigationBarItems within { }

```
NavigationView {
    List(transactions) { transaction in
        NavigationLink(destination:TransactionDetailView
            (transaction: transaction)) {
            TransactionRowView(transaction: transaction)
        }
    }
    .navigationTitle("Transactions")
    .navigationBarItems(trailing: Button(action: {
        isShowingAddTransaction = true
    }) {
        Image(systemName: "plus")
    })

}
```

State of isShowingAddTransaction?

```
        .navigationTitle("Transactions")
        .navigationBarItems(trailing: Button(action: {
            isShowingAddTransaction = true
        }) {
            Image(systemName: "plus")
        })
```

```
    .sheet(isPresented: $isShowingAddTransaction) {
        AddTransactionView(transactions: $transactions)
    }
    .onChange(of: isShowingAddTransaction) { newValue in
        print("isShowingAddTransaction changed to:
            \(newValue)")
        if lastTransactionState != newValue {
            lastTransactionState = newValue
            if !newValue {
                print("Sheet was dismissed!")
            }
        }
    }
```

Why .\ used for scenePhase?

```
@Environment(\.scenePhase) var scenePhase
```

\.colorScheme: Light or dark mode

\.locale: Current locale settings

\.scenePhase: Current lifecycle phase of the scene (active, inactive, background)

\.calendar: Current calendar

\.timeZone: Current time zone

Swift-Objective-C Interoperability

The @objc Attribute

- Used to expose swift elements to Objective-C
- Applied to classes, properties, methods and protocols

```swift
@objc class SwiftPerson: NSObject {
    @objc let name: String
    @objc private(set) var age: Int

    @objc init(name: String, age: Int) {
        self.name = name
        self.age = age
        super.init()
    }

    @objc func introduce() {
        print("Hi, I'm \(name), a Swift person aged \(age).")
    }
}
```

Bridging Header

Named [ProjectName] - Bridging-header.h

- Allows Swift code to use Objective-C classes
- Add Objective-C import statements here

```
4
5  #import "Person.h"
6  #import "ObjCUser.h"
7
```

Bridging Header

Generated Interface Header

- Named [ProjectName]-Swift.h
- Automatically generated by Xcode
- Import in Objective-C files to use Swift code

```objc
#import "ObjCUser.h"
#import "SwiftObjCInteropDemo-Swift.h"

@implementation ObjCUser

- (void)useSwiftPerson {
    SwiftPerson *swiftPerson = [[SwiftPerson alloc] initWithName:@"Alice" age:28];
    [swiftPerson introduce];
}

- (void)useSwiftClosure:(void (^)(NSString *))closure {
    closure(@"Hello from Objective-C!");
}

@end
```

SwiftUI and UIKit Interoperability

UIViewControllerRepresentable

- Use to wrap UIKit view controllers for use in SwiftUI

```
struct UIKitViewControllerRepresentable: UIViewControllerRepresentable {
    func makeUIViewController(context: Context) -> UIKitViewController {
        UIKitViewController()
    }

    func updateUIViewController(_ uiViewController: UIKitViewController, context:
        Context) {}
}
```

SwiftUI and UIKit Interoperability

UIHostingController

- Use to present SwiftUI views in UIKit

```
@objc private func showSwiftUIView() {
    let swiftUIView = SwiftUIView {
        self.dismiss(animated: true, completion: nil)
    }
    let hostingController = UIHostingController(rootView: swiftUIView)
    present(hostingController, animated: true, completion: nil)
}
```