State Management in SwiftUI

Understanding the tools for managing app state

What is State?

- Definition: 'State is any data that can change over time'
- Examples:

  - User input

  - API responses

  - Toggle switches

  - Animation progress

Why State Management Matters?

- Consistency between UI and data
- Performance optimization
- Code organisation and maintainability
- Predictable app behaviour

SwiftUI State Management Tools vs UIKit

SwiftUI:

- @State
- @Binding
- @ObservedObject
- @StateObject
- @EnvironmentObject
- @Environment

UIKit:

- Properties
- Delegation
- Key-Value Observing (KVO)
- NotificationCenter

@State and UIKit Properties

SwiftUI: @State

```
@State private var count = 0
```

UIKit:

```
var count = 0 {
  didSet {
     updateUI()
      }
  }
```

@Binding and UIKit Delegation

SwiftUI: @Binding

```
struct ToggleView: View {
    @Binding var isOn: Bool
     var body: some View {
     Toggle("Switch", isOn: $isOn)
      }
}
```

UIKit: Delegation

```
protocol ToggleViewDelegate: AnyObject {
func toggleView(_ toggleView: ToggleView, didChangeState isOn: Bool)
}

class ToggleView: UIView {
weak var delegate: ToggleViewDelegate?
// Implementation...
}
```

@ObservedObject and KVO

SwiftUI: @ObservedObject

```
class UserSettings: ObservableObject {
@Published var username = ""
}

struct ProfileView: View {
@ObservedObject var settings: UserSettings
// View body...
}
```

UIKit: Key-Value Observing (KVO)

```
class UserSettings: NSObject {
   @objc dynamic var username = ""
}

class ProfileViewController: UIViewController {
var observation: NSKeyValueObservation?

override func viewDidLoad() {
   super.viewDidLoad()
    observation = settings.observe(\.username, options: [.new]) { [weak self] _, change in
    self?.updateUI()
          }
     }
}
```

@StateObject

SwiftUI: @StateObject

```
struct ContentView: View {

@StateObject private var settings = UserSettings()

// View body...

}
```

UIKit: No direct equivalent

```
class ContentViewController: UIViewController {

private let settings = UserSettings()

// Implementation...

}
```

@EnvironmentObject and Dependency Injection

SwiftUI: @EnvironmentObject

```
@main
struct MyApp: App {
    var body: some Scene {
    WindowGroup {
    ContentView()
        .environmentObject(UserSettings())
                        }
            }
}

struct ProfileView: View {
@EnvironmentObject var settings: UserSettings
// View body...
}
```

@Environment and UIKit's UITraitCollection

SwiftUI: @Environment

```
struct ContentView: View {
@Environment(\.colorScheme) var colorScheme

var body: some View {
   Text("Current mode: \(colorScheme == .dark ? "Dark" : "Light")")
     }
}
```