


Build iOS apps with SwiftUI


Present by Wing Chan




Introduction

- SwiftUI is a modern and intuitive framework for building iOS apps.
 - It provides a declarative approach to app development, making it easier and faster to create stunning user interfaces.
 - In this presentation, we will explore the key features and benefits of SwiftUI and demonstrate how to build iOS apps using SwiftUI.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.


History

- Swift was first announced by Apple at the Apple Worldwide Developers Conference (WWDC) in June 2014.
 - Swift was designed to address the limitations of Objective-C, which was the primary programming language for Apple's platforms at the time. It aimed to provide a more efficient, expressive, and beginner-friendly language.
 - SwiftUI was introduced by Apple at WWDC 2019 as a modern framework for building user interfaces across Apple platforms.
 - SwiftUI is designed to be cross-platform, supporting iOS, macOS, watchOS, tvOS and visionOS with a shared codebase.
 - It is compatible with iOS 13 and later.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Key features

- Declarative Syntax: SwiftUI allows developers to describe the user interface and behavior of their apps using a simple and intuitive declarative syntax.
 - Swift Integration: SwiftUI seamlessly integrates with Swift, Apple's powerful and easy-to-use programming language.
 - Live Previews: With SwiftUI, developers can see real-time previews of their UI changes in Xcode, making it easier to iterate and refine their designs.
 - Native Performance: SwiftUI leverages the full power of the underlying platform to provide smooth and performant user experiences.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Elements

- Views: In SwiftUI, views are the building blocks of the user interface. Each view represents a specific part of the UI, such as a button, label, or image. Views can also be combined to create complex and nested interfaces.
 - Modifiers: Modifiers are used to customize the appearance and behavior of views. They allow you to apply various changes to a view, such as setting the font, color, padding, or adding animations.
 - Layouts: SwiftUI offers flexible layout options for arranging views on the screen. You can use layout views like VStack (vertical stack) and HStack (horizontal stack) to stack views vertically or horizontally, respectively. Additionally, you can use ZStack to overlay views, and Grid for grid-based layouts.
 - State and Data Flow: SwiftUI leverages a concept called "State" to handle changes and updates within the app. By using the @State property wrapper, you can create mutable state properties within a view that trigger UI updates whenever the state changes. SwiftUI manages the state and automatically updates the view accordingly. Additionally, SwiftUI provides mechanisms like @Binding and @ObservableObject to propagate data changes across different views and components.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Elements

- Gestures: SwiftUI simplifies gesture handling by providing a range of gesture modifiers that can be attached to views. These modifiers enable interaction with the user interface through tap gestures, swipe gestures, long-press gestures, and more. SwiftUI also supports gesture recognition and coordination for complex interactions.
- Animation: SwiftUI provides a powerful animation framework that allows you to create smooth and engaging transitions and effects. With simple modifiers like `.animation()`, you can add animations to views and even chain multiple animations together. SwiftUI's animation system is built to be easy to use and offers options for customizing the duration, timing, and easing of animations.
- Integration with UIKit: SwiftUI seamlessly integrates with UIKit, Apple's original framework for building iOS apps. You can embed SwiftUI views within existing UIKit-based projects, enabling a gradual adoption of SwiftUI while leveraging the existing codebase and components. This integration enables developers to take advantage of SwiftUI's modern features while still utilizing UIKit's extensive ecosystem and APIs.

Code sample

```
import SwiftUI

struct ContentView: View {
    @State private var rotation: Double = 0.0

    var body: some View {
        VStack {
            Text("Welcome to SwiftUI")
                .font(.largeTitle)
                .foregroundColor(.blue)
                .padding()

            Rectangle()
                .fill(Color.red)
                .frame(width: 200, height: 200)
                .rotationEffect(.degrees(rotation))
                .animation(.linear(duration: 2))
                .gesture(
                    RotationGesture()
                        .onChanged { angle in
                            rotation =
                                angle.degrees
                        }
                )
                .padding()
        }
    }
}
```

Demo

