Introduction to SwiftUI

Michael
10/17/2019

Outline

- Views and Modifiers
- Building custom views
- State and Binding
- Data Flow
- Lifecycle of SwiftUI View

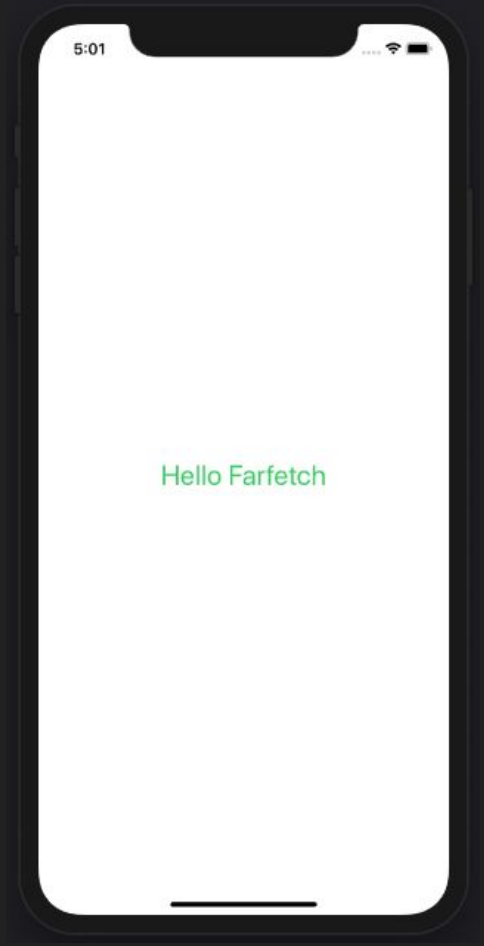
What is SwiftUI?

SwiftUI is an innovative, exceptionally simple way to build user interfaces across all Apple platforms with a declarative Swift syntax.

Views

A view defines a piece of UI

```
import SwiftUI

struct HelloView: View {

    var body: some View {
        VStack() {
            Text("Hello Farfetch")
                .font(.title)
                .foregroundColor(.green)
        }
    }
}
```

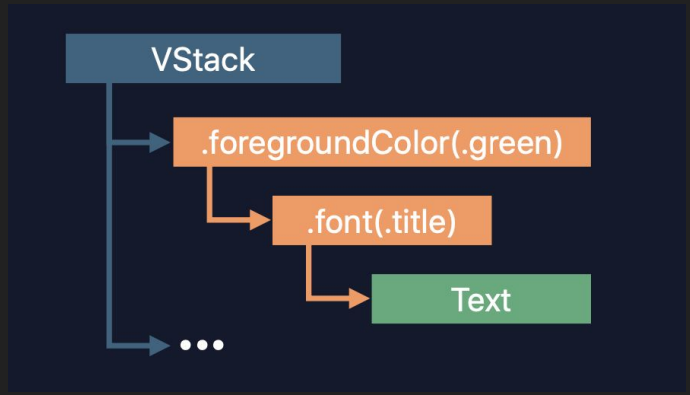


View protocol

```swift
/// A piece of user interface.
///
/// You create custom views by declaring types that conform to the `View`
/// protocol. Implement the required `body` property to provide the content
/// and behavior for your custom view.
@available(iOS 13.0, OSX 10.15, tvOS 13.0, watchOS 6.0, *)
public protocol View {

    /// The type of view representing the body of this view.
    ///
    /// When you create a custom view, Swift infers this type from your
    /// implementation of the required `body` property.
    associatedtype Body : View

    /// Declares the content and behavior of this view.
    var body: Self.Body { get }
}
```

Views Using Modifiers

```
VStack() {
    Text("Hello Farfetch")
        .font(.title)
        .foregroundColor(.green)
}
```




```
extension Text {

    /// Sets the color of this text.
    ///
    /// - Parameter color: The color to use when displaying this text.
    /// - Returns: Text that uses the color value you supply.
    public func foregroundColor(_ color: Color?) -> Text

    /// Sets the font to use when displaying this text.
    ///
    /// - Parameter font: The font to use when displaying this text.
    /// - Returns: Text that uses the font you specify.
    public func font(_ font: Font?) -> Text
```

View Container

- VStack
- HStack
- ZStack

```
VStack(alignment: .leading) {
    Image("farfetch.logo")
    Text("Hello")
    Text("Farfetch")
}
```

```
public struct VStack<Content : View> : View {
    public init(
        alignment: HorizontalAlignment = .center,
        spacing: Length? = nil,
        @ViewBuilder content: () -> Content
    )
}
```

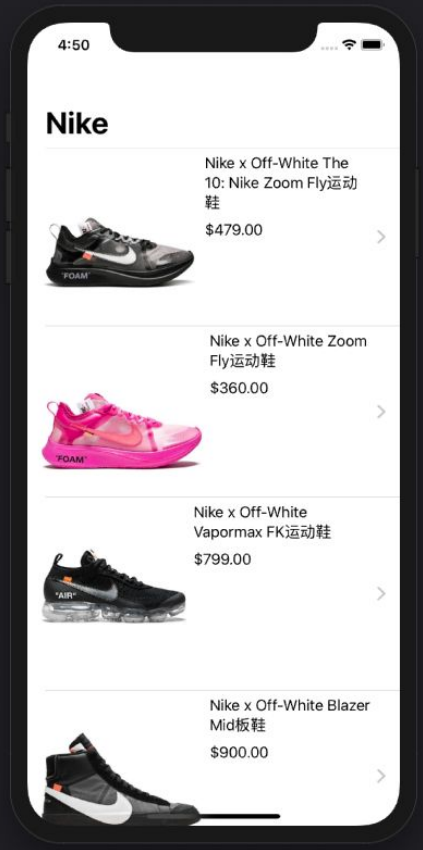
View Container

```
struct ProductDetailView: View {

    var body: some View {
        VStack(alignment: .center, spacing: 10) {
            Image("13678429")
                .resizable()
                .aspectRatio(contentMode: .fit)

            Text("Nike Zoom Fly运动鞋")
            Text("$479")

        }.padding()
    }
}
```

Building custom views

- Prefer smaller, single-purpose views
- Build larger views using composition

Demo


4:50
Nike
Nike x Off-White The 10: Nike Zoom Fly运动鞋
$479.00
Nike x Off-White Zoom Fly运动鞋
$360.00
Nike x Off-White Vapormax FK运动鞋
$799.00
Nike x Off-White Blazer Mid板鞋
$900.00

Building custom views

```
struct ProductRowView: View {
    var product: Product

    var body: some View {
        HStack(alignment: .top) {
            Image(product.imageName)
                .resizable()
                .aspectRatio(contentMode: .fit)

            VStack(alignment: .leading, spacing: 10) {
                Text(product.name)
                Text(String(format: "$%.2f", product.price))

                Spacer()
            }
            Spacer()
            Image(systemName: "star")
                .padding()
        }
    }
}
```

State and Binding

Connect views to your app’s underlying data model.

- @State
- @Binding

@State

persisted between view refresh

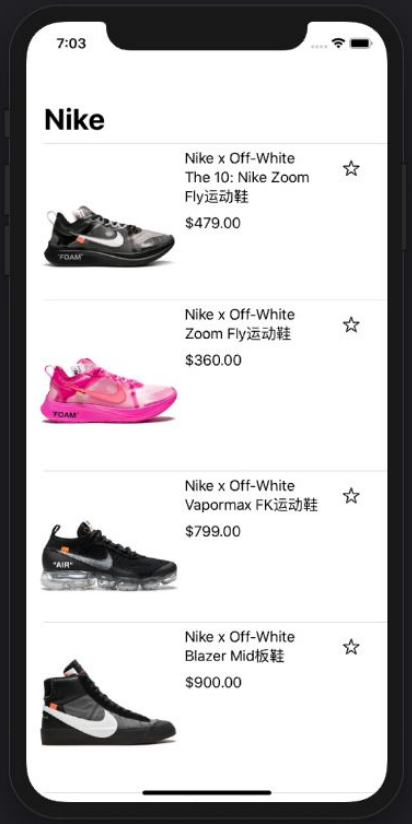
```
struct HelloView: View {
    @State private var name = ""

    var body: some View {
        VStack() {
            TextField("enter name here", text: $name)
                .font(.title)
                .padding()

            Text("Hello \(name)")
                .font(.title)
                .foregroundColor(.green)
        }
    }
}
```

@Binding

Create a two-way connection to a value managed by something else.

Most probably it is a @State from a parent view

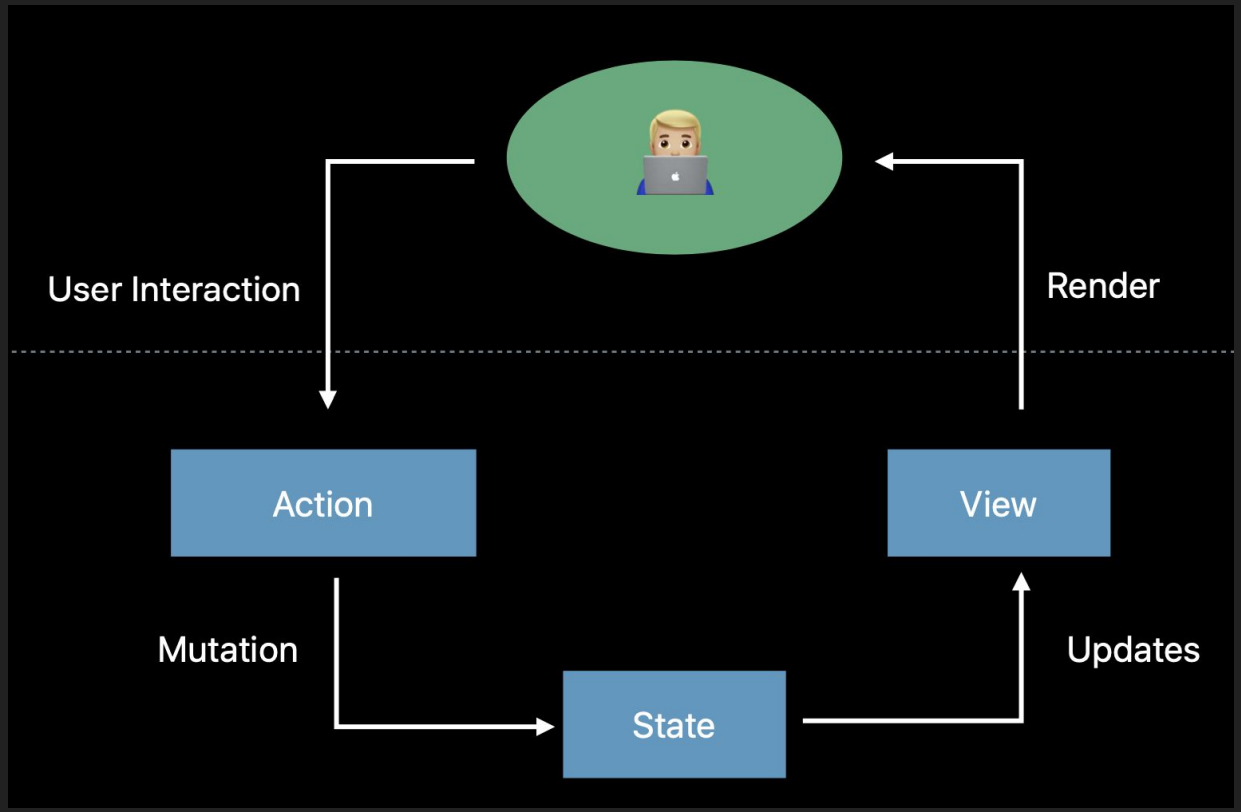
Demo


7:03
Nike
Nike x Off-White The 10: Nike Zoom Fly运动鞋
$479.00
Nike x Off-White Zoom Fly运动鞋
$360.00
Nike x Off-White Vapormax FK运动鞋
$799.00
Nike x Off-White Blazer Mid板鞋
$900.00

Difference between @State and @Binding

- @State used for local/private changes inside a View
- @Binding used in subviews/reusable components when the value lives outside the current view domain.

Data flow in SwiftUI



Lifecycle of SwiftUI View

“Your SwiftUI code will be maybe 10–20% of what your UIKit code was — almost all of it disappears because we no longer repeat ourselves, no longer need to handle so many lifecycle methods, and more.” — Paul Hudson

- View Initialization
- onAppear
- onDisappear
- State and Data Flows

Resources

- SwiftUI Videos
- SwiftUI Doc
- SwiftUI on github

Better apps. Less code.

Q & A