LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Felix Wohnhaas

SwiftUI – An Introduction

Praktikum iOS WS 19/20

SwiftUI – A UI Framework

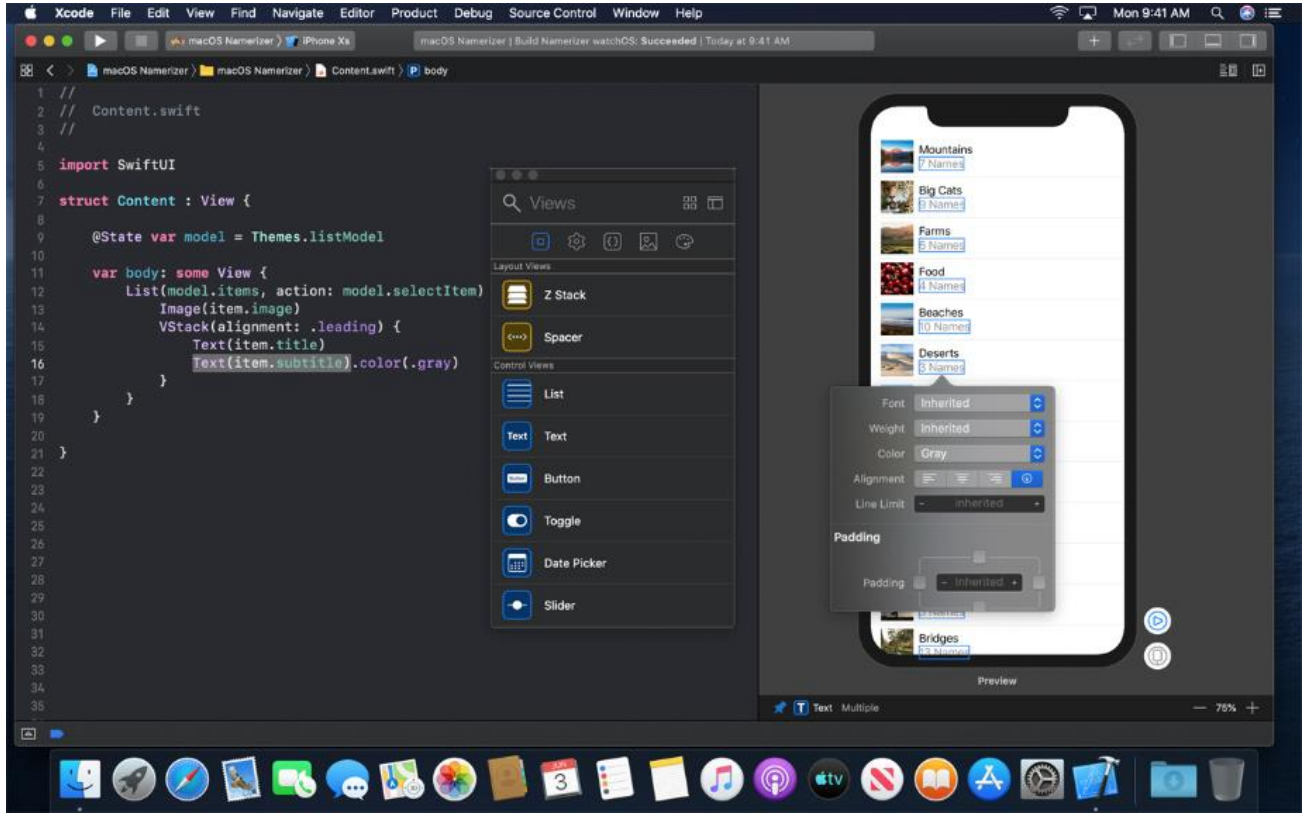
- A User Interface Toolkit built on top of Swift
- Native on all Apple Platforms
- New Design Tools inside Xcode, improving development
- Declarative Syntax, with focus on simplicity

Native on all Apple Platforms

- Uses the same API across all apple platforms
- Removes the barrier between iOS Development, Watch, Mac and TV Applications
- Uses the same declarative controls to achieve platform-specific behaviour

New Design Tools

- Intuitive new Design Tools which allow simple Dragging and Dropping to compose User interfaces
- Preview in the Design Canvas always in sync with the code you write
- Instant recompiling of changes to see a live version of your app at all times



Declarative API vs Imperative API

- Declarative Syntax allows the developer to…
  - State what their interface should do, instead of describing how to do it
  - Describe their dynamic UI as a transition of state
  - Easily compose their code into seperate components

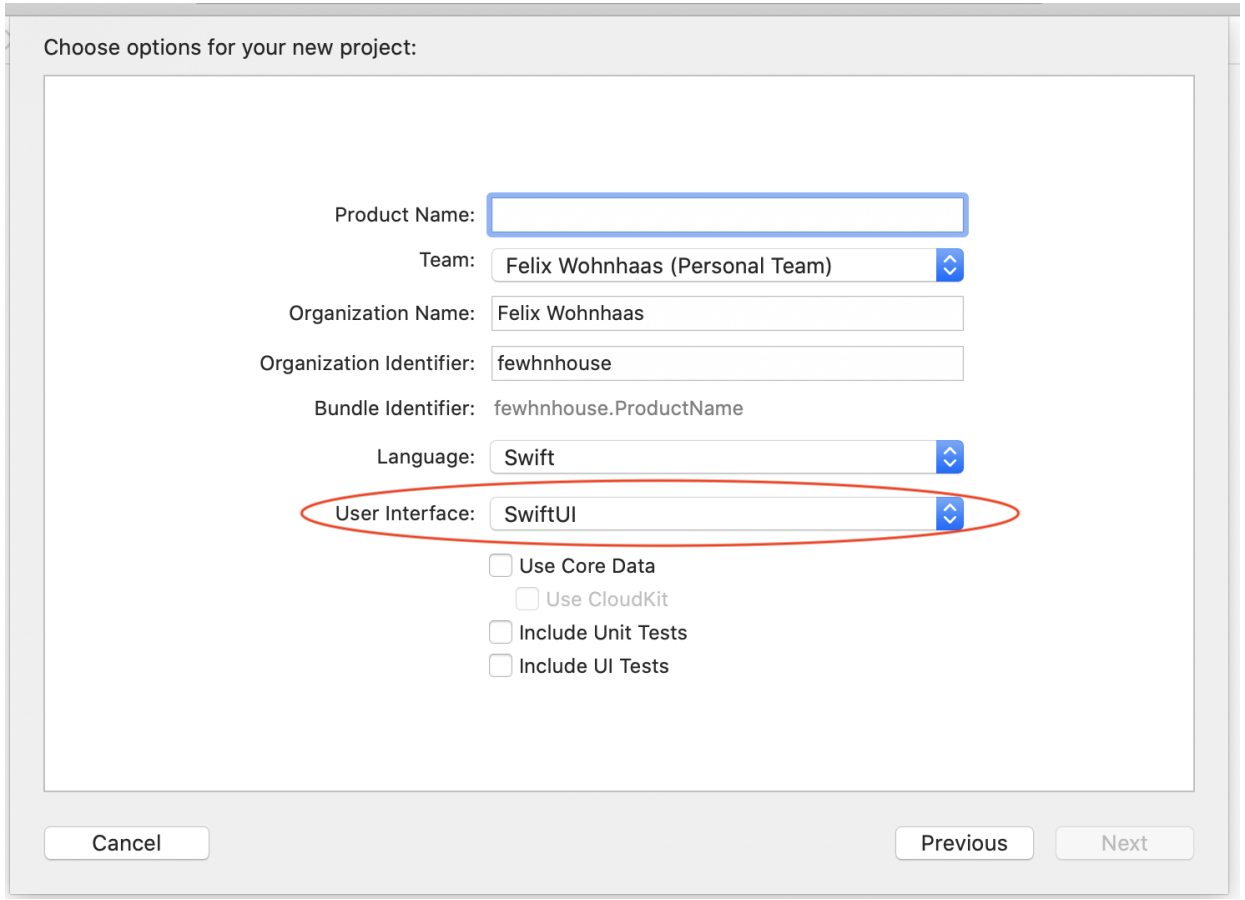
- Imperative Syntax allows the developer to…
  - Describe the logic of the UI via Control Flow
  - Use statements to change your UI state

SwiftUI – What does it provide?

- Views, controls, and layout structures
- A variety of modifiers to customize these views and controls
- Event handlers for delivering taps, gestures, and other types of input
- Out-of-the-box Animation Support
- Tools to manage the flow of data from the models down to the views and controls
- State Management and Transitioning

Create a SwiftUI Project

- Use Xcode 11
- Steps similar to normal project creation
- Make sure to select „SwiftUI“ in the User Interface Setting

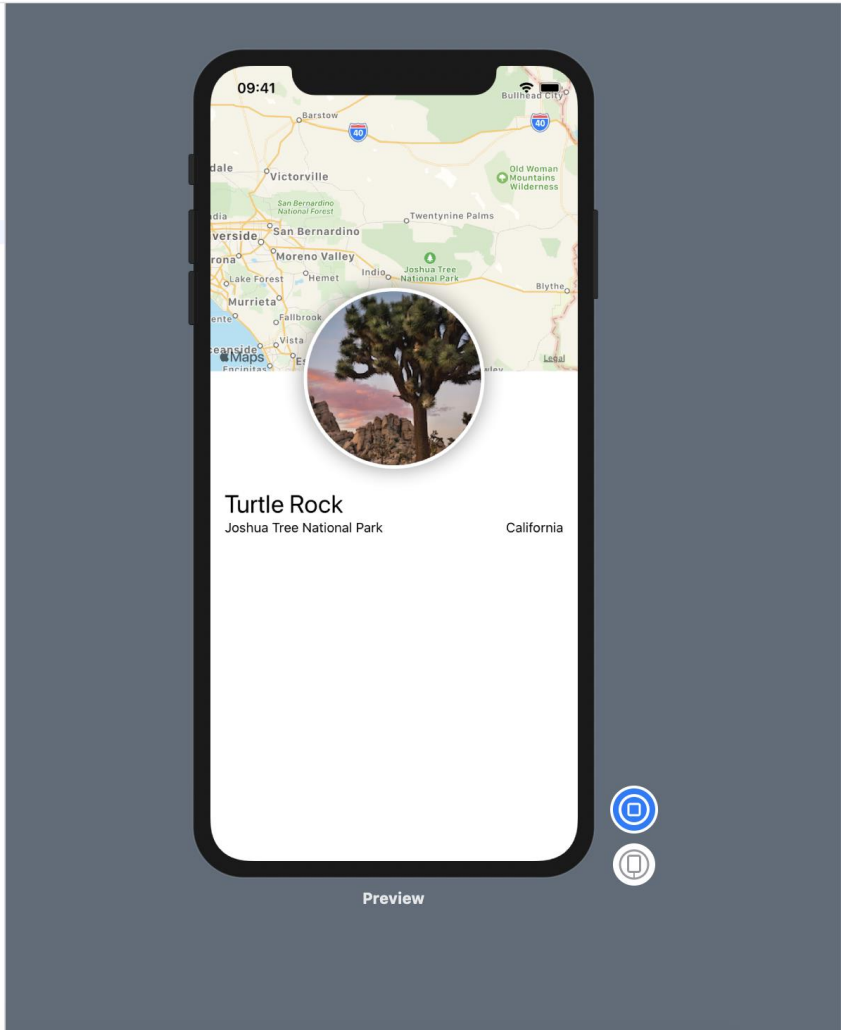


SwiftUI – The Code and Preview

- Xcode allows you to see the code and the preview in sync with each other
- Changes in either of the two instantly become visible in the other
- Previews can be run to test functionality

Landmarks 〉 Landmarks 〉 ContentView.swift 〉 No Selection

```swift
//
//  ContentView.swift
//  Landmarks
//
//  Created by Felix Wohnhaas on 29.09.19.
//  Copyright © 2019 Felix Wohnhaas. All rights reserved.
//

import SwiftUI

struct ContentView: View {
    var body: some View {
        VStack {
            MapView()
                .edgesIgnoringSafeArea(.top)
                .frame(height: 300)
            CircleImage()
                .offset(y: -100)
                .padding(.bottom, -100)
            VStack(alignment: .leading) {
                Text("Turtle Rock")
                    .font(.title)
                HStack {
                    Text("Joshua Tree National Park")
                        .font(.subheadline)
                    Spacer()
                    Text("California")
                        .font(.subheadline)
                }
            }.padding()
            Spacer()
        }

    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}

```

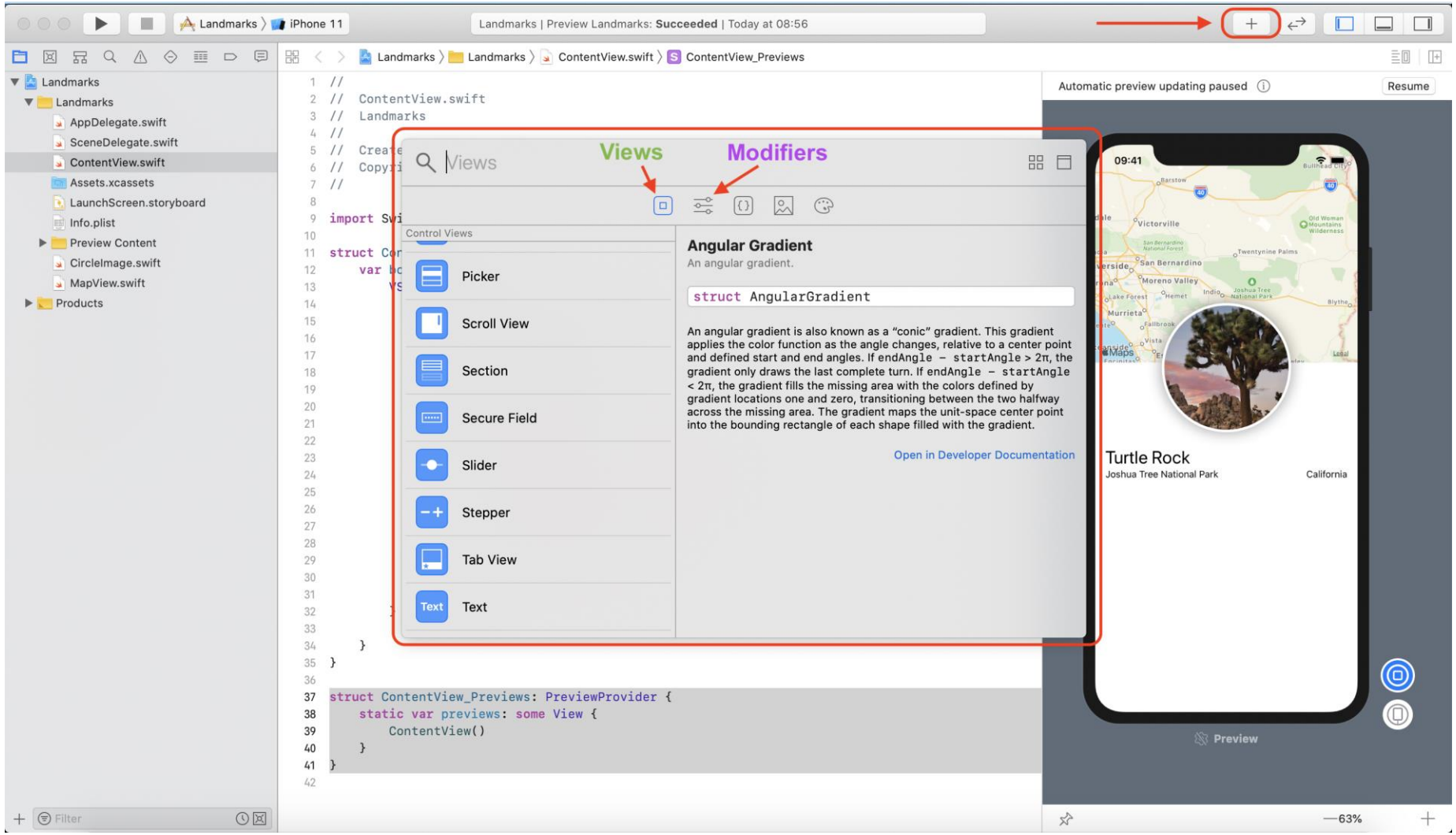


SwiftUI – A component

- A SwiftUI Component is composed of many views
- Each view can have specific arguments and predefined modifiers to customize them
- The Component offers a PreviewProvider to customize the behaviour in Preview

```
9  import SwiftUI
10
11 struct ContentView: View {
12     var body: some View {
13         VStack {                                   View
14             MapView()
15                 .edgesIgnoringSafeArea(.top)
16                 .frame(height: 300)
17             CircleImage()                          Modifiers
18                 .offset(y: -100)
19                 .padding(.bottom, -100)
20             VStack(alignment: .leading) {
21                 Text("Turtle Rock")                View with
22                     .font(.title)                  arguments
23                 HStack {
24                     Text("Joshua Tree National Park")
25                         .font(.subheadline)
26                     Spacer()
27                     Text("California")
28                         .font(.subheadline)
29                 }
30             }.padding()
31             Spacer()
32         }                                          Body
33
34     }                                              Component
35 }
36
37 struct ContentView_Previews: PreviewProvider {
38     static var previews: some View {
39         ContentView()
40     }                                              Preview
41 }
42
```

SwiftUI – Add views and modifiers



SwiftUI – State Management

- Declare a state variable to manage state transitions
- Use bindings to change state variables with pre-defined components
- Use the state value to describe change in your UI

```
1  import SwiftUI
2
3  struct LandmarkList: View {
4      @State var showFavoritesOnly = true
5
6      var body: some View {
7          NavigationView {
8              List {
9                  Toggle(isOn: $showFavoritesOnly) {
10                     Text("Favorites only")
11                 }
12
13                 ForEach(landmarkData) { landmark in
14                     if !self.showFavoritesOnly || landmark.isFavorite {
15                         NavigationLink(destination: LandmarkDetail(landmark: landmark)) {
16                             LandmarkRow(landmark: landmark)
17                         }
18                     }
19                 }
20             }
21             .navigationBarTitle(Text("Landmarks"))
22         }
23     }
24 }
```

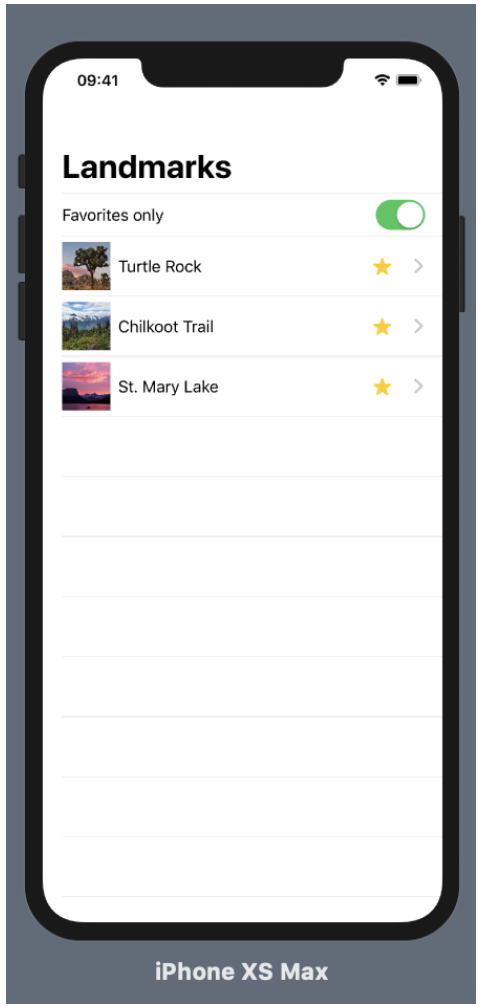
State Variable

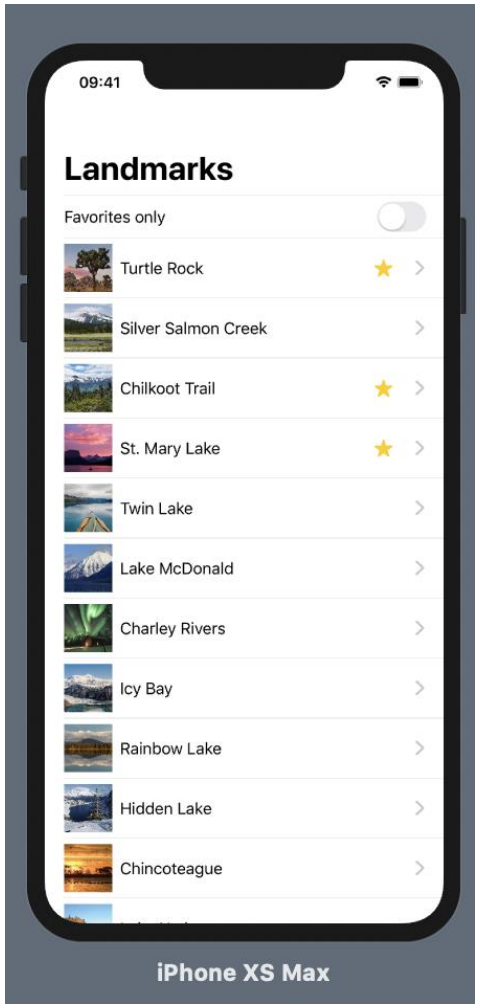
Binding

Usage of state value

What does this component render?

SwiftUI – State Management





SwiftUI – Observable Objects

- SwiftUI tracks any changes occuring to Observable Objects and publishes them to Subscribers

- Make your data object conform to the ObservableObject Protocol

- Mark the containing variables as published if you want their changes to be published to subscribers (components)

```
8  import Combine
9  import SwiftUI
10
11 final class UserData: ObservableObject {
12     @Published var showFavoritesOnly = false
13     @Published var landmarks = landmarkData
14 }
15
```

SwiftUI – Environment Objects

- Add the @EnvironmentObject modifier to subscribe to the Observable.
- Environment Objects also offer access to bindings ($), just like State
- SwiftUI automatically updates any UI which depends on the observable values.

```swift
import SwiftUI

struct LandmarkList: View {
    @EnvironmentObject var userData: UserData

    var body: some View {
        NavigationView {
            List {
                Toggle(isOn: $userData.showFavoritesOnly) {
                    Text("Favorites only")
                }

                ForEach(userData.landmarks) { landmark in
                    if !self.userData.showFavoritesOnly || landmark.isFavorite {
                        NavigationLink(destination: LandmarkDetail(landmark: landmark)) {
                            LandmarkRow(landmark: landmark)
                        }
                    }
                }
            }
            .navigationBarTitle(Text("Landmarks"))
        }
    }
}
```

Environment Object declaration (Subscription)

Binding

Usage