移动平台开发技术

iOS应用的数据模式与SwiftUI和UIKit交互

章国锋 2026春夏学期

浙江大學
ZHEJIANG UNIVERSITY

SwiftUI中的属性包装器

SwiftUI中的数据模式

SwiftUI与UIKit交互

浙江大学
ZHEJIANG UNIVERSITY

SwiftUI中的属性包装器

SwiftUI中的数据模式

SwiftUI与UIKit交互

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

交互响应属性包装器

Property wrappers for interaction

数据持久化属性包装器

Property wrappers for data persistent

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

```
@State & @Binding

@StateObject & @ObservedObject & @EnvironmentObject

@Environment
```

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

@State & @Binding

- 当需要因视图内的数据变化而触发视图更新时，**@State** 是理想的选择。
- 它常用于简单的 UI 组件状态管理，如开关状态、文本输入等。
- 如果数据不需要复杂的跨视图共享，使用 **@State**可以简化状态管理。
- **@Binding** 不直接持有数据，而是提供了对其他数据源的读写访问的包装
- 它允许 UI 元素直接修改数据，并反映这些数据的变化

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

`@StateObject` & `@ObservedObject` & `@EnvironmentObject`

- `@StateObject` 专门用于管理符合 `ObservableObject` 协议的实例。
- 标注的对象实例在视图的整个生命周期中保持唯一，即使视图更新，对象实例也不会重新创建。
- 相较 `@State` 而言，`@StateObject` 更适合管理复杂的数据模型及其执行逻辑
- `@ObservedObject` 不持有被观察的实例，不保证其生存期
- 使用 `@EnvironmentObject` 前，必须确保已在视图层级的上游提供了相应的实例（通过 `.environmentObject` 修饰器），否则将导致运行时错误。
- 它对视图的更新触发条件与 `@StateObject` 和 `@ObservedObject` 一样。

SwiftUI中的属性包装器

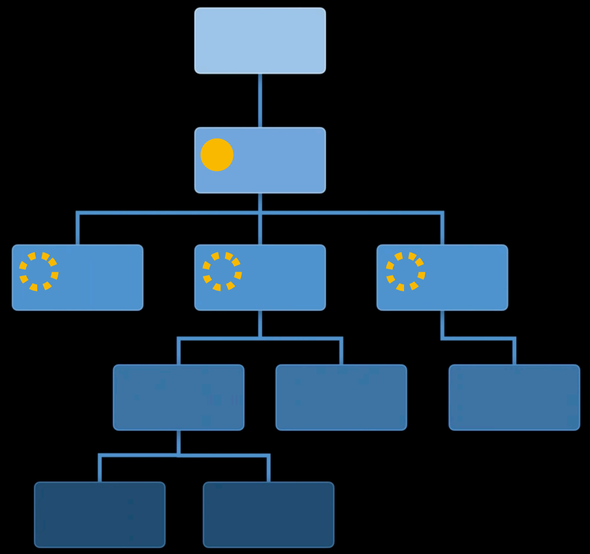
Property wrappers in SwiftUI

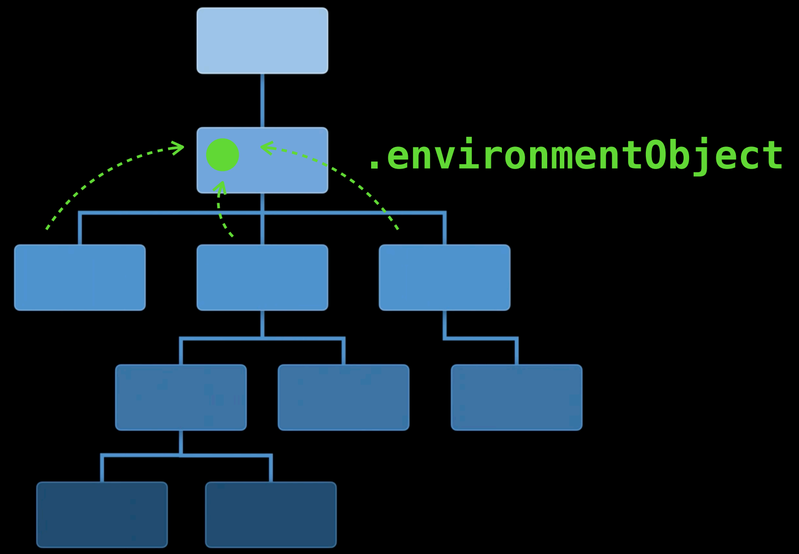
浙江大学 ZHEJIANG UNIVERSITY

数据流属性包装器

Property wrappers for data flow

@StateObject & @ObservedObject & @EnvironmentObject





SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

- 当需要访问和响应如界面样式（暗模式/亮模式）、设备方向、字体大小等由系统或上层视图提供的环境值时（通常对应值类型）
- 当需要访问和调用 SwiftData 的 ModelContext 时（对应引用类型）
- 当需要使用系统提供的一些方法时，比如 **`dismiss`**、**`openURL`**（通过 `struct` 的 **`callAsFunction`** 封装的方法）

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

- 在 Observation 框架的背景下，**`@State`** 和 **`@Environment`** 成为了最主要的属性包装器

`@Environment`

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

`@State` & `@Binding`

`@StateObject` & `@ObservedObject` & `@EnvironmentObject`

`@Environment`

`@StateObject`、`@ObservedObject` 和 `@EnvironmentObject` 专用于关联符合 `ObservableObject` 协议的实例。

`@StateObject` 通常用于创建和维护实例，而 `@ObservedObject` 用于引入和响应已存在的实例，`@EnvironmentObject` 适用于在多视图中数据传递

在 iOS 17+ 的环境中，如果应用主要依赖于 Observation 和 SwiftData 框架，那么这三个属性包装器的使用频率可能会相对较低

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

交互响应属性包装器

Property wrappers for interaction

数据持久化属性包装器

Property wrappers for data persistent

SwiftUI中的属性包装器

Property wrappers in SwiftUI

交互响应属性包装器

Property wrappers for interaction

```
@FocusState

@GestureState

@ScaledMetric

@Namespace
```

SwiftUI中的属性包装器

Property wrappers in SwiftUI

交互响应属性包装器

Property wrappers for interaction

`@FocusState`

- 主要用于管理和追踪用户界面中的焦点状态
- 目前，只有 `TextField` 和 `TextEdit` 支持通过代码修改 `@FocusState` 的值来获得或失去焦点

SwiftUI中的属性包装器

Property wrappers in SwiftUI

交互响应属性包装器

Property wrappers for interaction

@GestureState

- 临时存储与手势相关的状态
- 让手势处理代码更加简洁，且易于维护

SwiftUI中的属性包装器

Property wrappers in SwiftUI

交互响应属性包装器

Property wrappers for interaction

`@ScaledMetric`

- 主要用于根据用户的辅助功能设置（如更大的文本大小）自动调整数值
- 它能够确保应用界面在不同用户偏好下保持可用性和舒适性
- 可以用于调整任何需要根据系统字体大小比例变化的尺寸，如图标大小、布局间距等。
- 使用 `.dynamicTypeSize` 限制视图的动态类型尺寸变化范围，防止布局异常

SwiftUI中的属性包装器

Property wrappers in SwiftUI

交互响应属性包装器

Property wrappers for interaction

- 通常应用于传递到其他视图中进行animation使用
- @Namespace 不仅限于与 matchedGeometryEffect 搭配使用，还可以用于其他元素或视图修饰器，如 accessibilityRotorEntry、AccessibilityRotorEntry、 accessibilityLinkedGroup、prefersDefaultFocus 和 defaultFocus

```
@Namespace
```

SwiftUI中的属性包装器

Property wrappers in SwiftUI

交互响应属性包装器

Property wrappers for interaction

```
@FocusState

@GestureState

@ScaledMetric

@Namespace
```

@FocusState 简化了焦点管理；@GestureState 自动化手势状态的生命周期；
@ScaledMetric 实现了尺寸的自动缩放；@Namespace 充当标识符的角色；

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

交互响应属性包装器

Property wrappers for interaction

数据持久化属性包装器

Property wrappers for data persistent

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据持久化属性包装器

Property wrappers for data persistent

`@AppStorage` & `@SceneStorage`

`@FetchRequest` & `@SectionedFetchRequest` & `@Query`

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据持久化属性包装器

Property wrappers for data persistent

@AppStorage & @SceneStorage

- 通过 UserDefaults，**@AppStorage**实现了数据的持久存储，确保即便应用关闭后，数据依然得以保存
- UserDefaults 的持久化不是原子级别的，存在数据丢失的风险
- 不建议用来保存关键或敏感数据
- 默认仅支持有限的数据类型。常见的数据类型如日期和数组等默认不被支持

- **@SceneStorage** 是一个专属于 SwiftUI 的独特概念，它并不对应任何已知的底层数据结构。
  因此，**@SceneStorage** 应仅在视图内部使用，不应在视图外部或视图模型中使用它

```
@SceneStorage("selectedTab") private var selectedTab: Int = 0
@AppStorage("isLogin") var isLogin: Bool = false
```

SwiftUI中的属性包装器

Property wrappers in SwiftUI

数据流属性包装器

Property wrappers for data flow

交互响应属性包装器

Property wrappers for interaction

数据持久化属性包装器

Property wrappers for data persistent

浙江大学
ZHEJIANG UNIVERSITY

SwiftUI中的属性包装器

SwiftUI中的数据模式

SwiftUI与UIKit交互

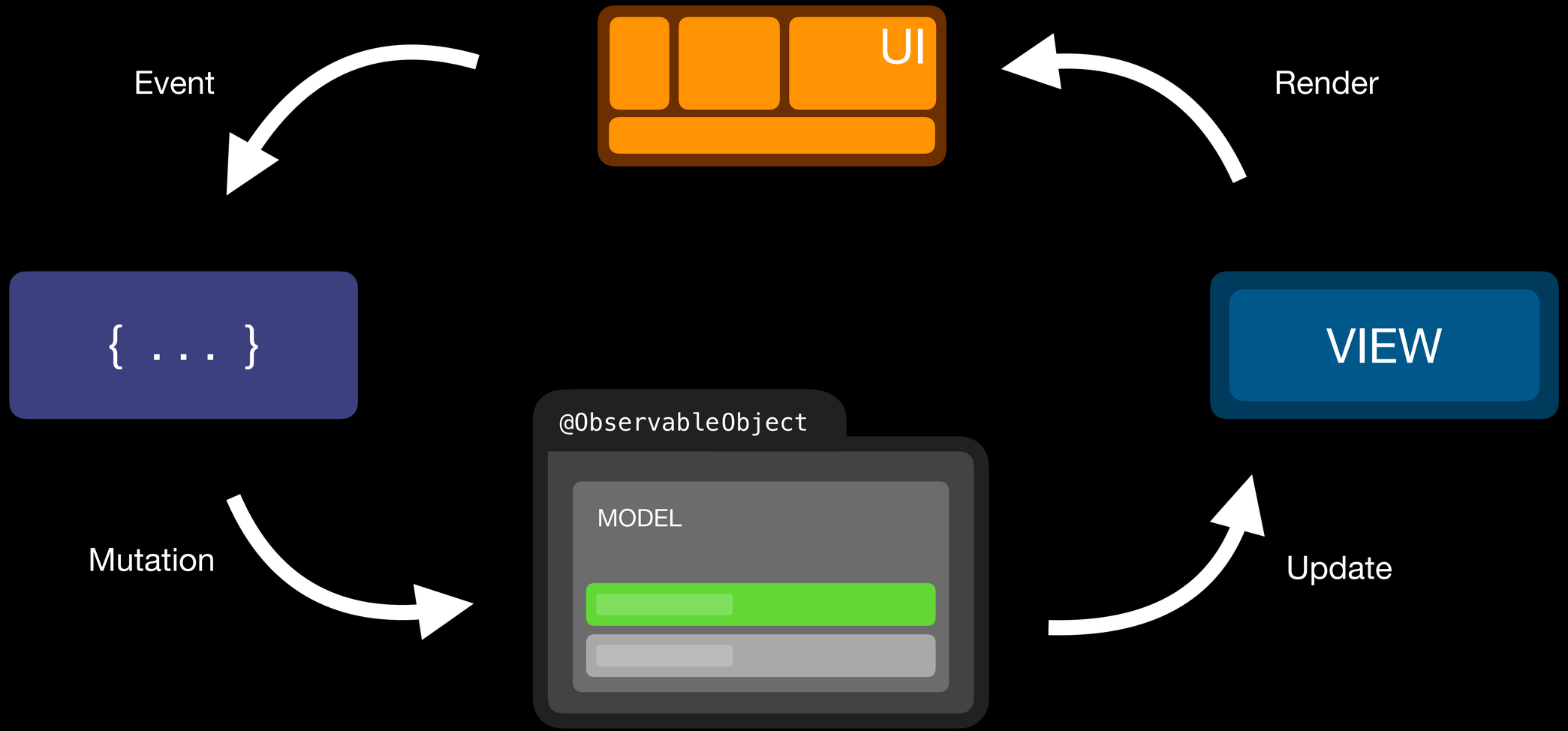
SwiftUI中的属性包装器

SwiftUI中的数据模式

SwiftUI与UIKit交互

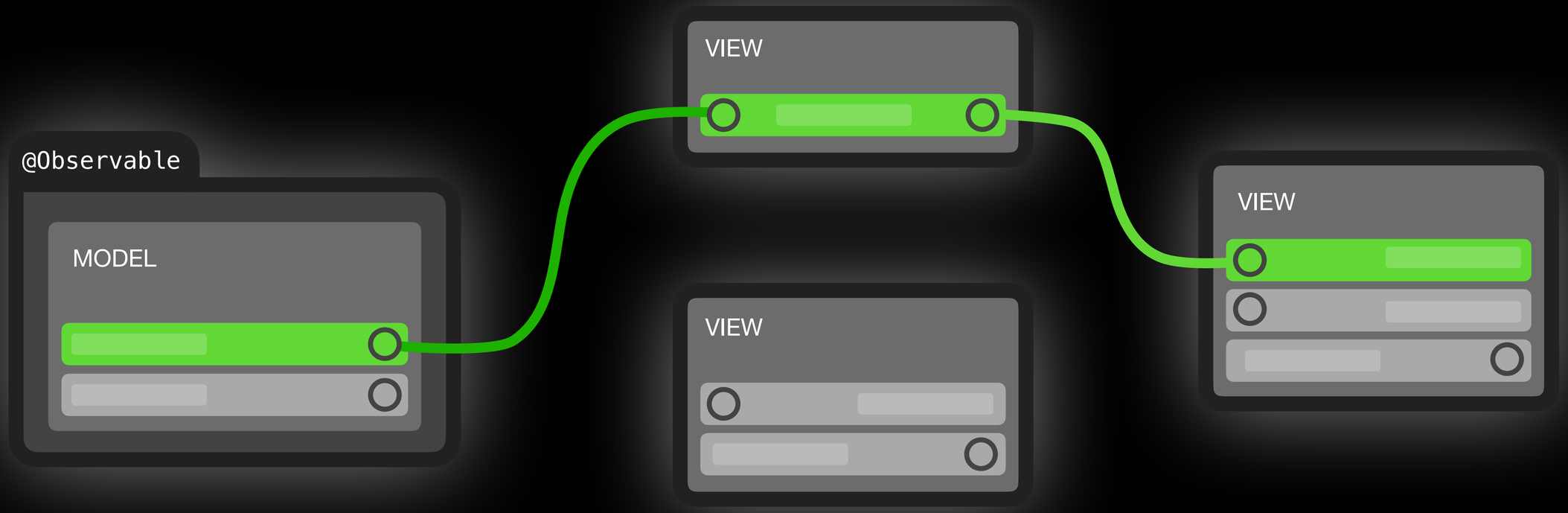
SwiftUI中的数据模式

Model-View-ViewModel architecture in SwiftUI



SwiftUI中的数据模式

Model-View-ViewModel architecture in SwiftUI



SwiftUI中的属性包装器

SwiftUI中的数据模式

SwiftUI与UIKit交互

浙江大学
ZHEJIANG UNIVERSITY

SwiftUI中的属性包装器

SwiftUI中的数据模式

SwiftUI与UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

SwiftUI与UIKit交互

Interfacing with UIKit

SwiftUI与UIKit交互

Interfacing with UIKit

UIKit

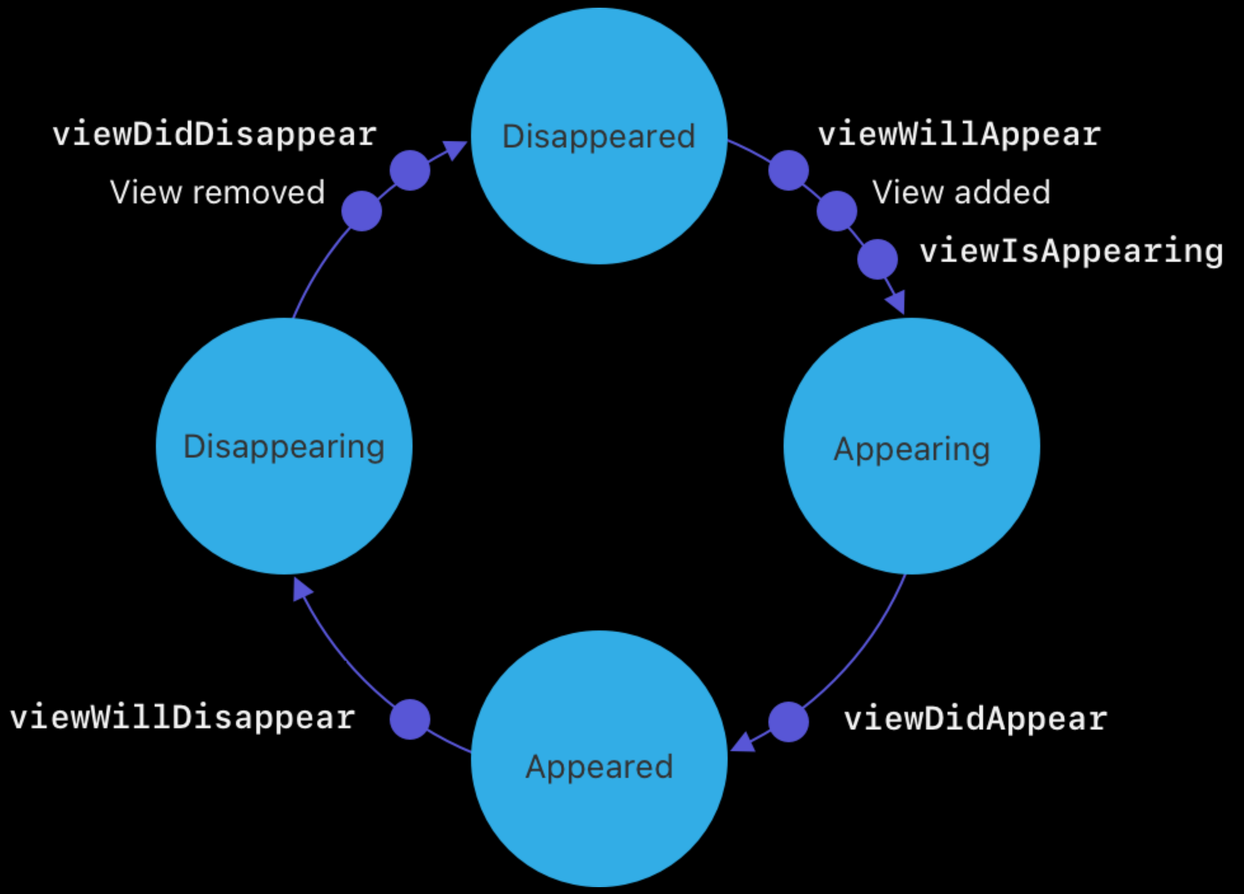
- View controllers
- Interface Builder and storyboards
- Delegates and data sources

SwiftUI与UIKit交互

Interfacing with UIKit

UIKit

View controllers



SwiftUI与UIKit交互

Interfacing with UIKit

UIKit

View controllers

Interface Builder and storyboards

Delegates and data sources

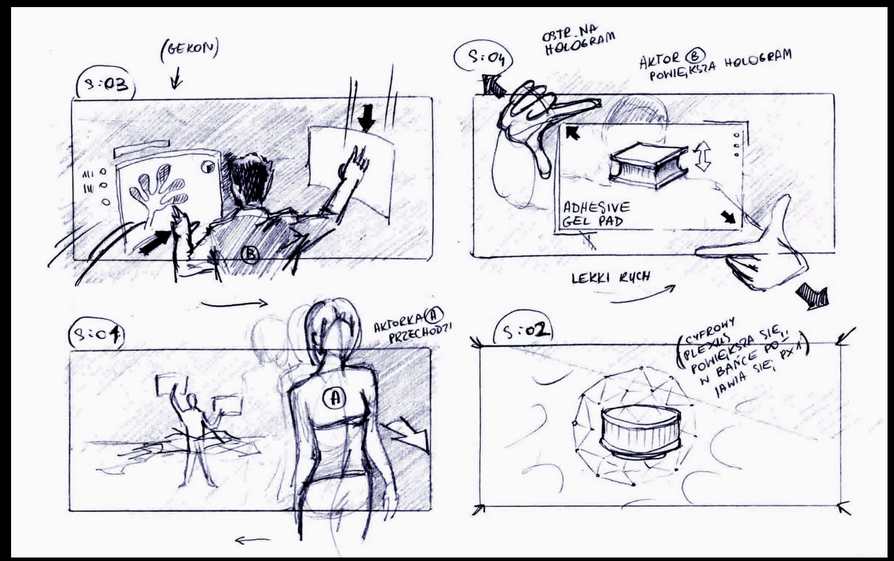
SwiftUI与UIKit交互

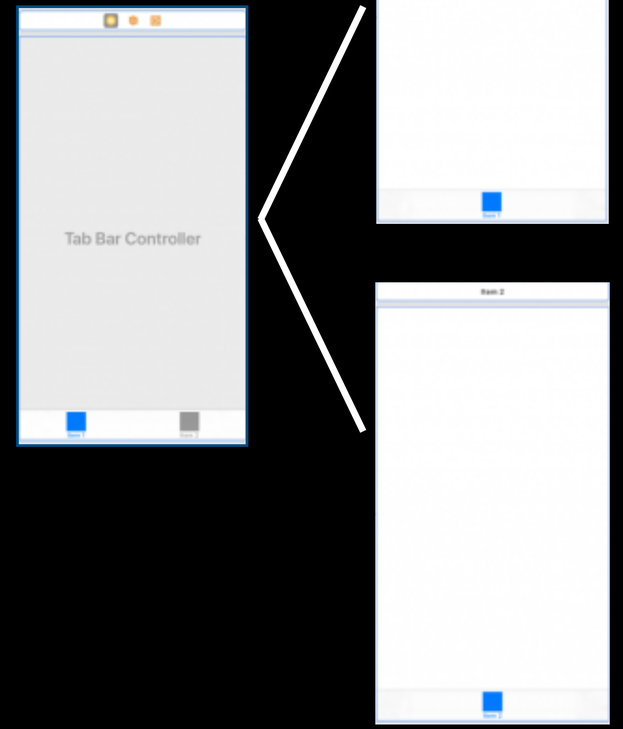
Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY

UIKit

Interface Builder and storyboards





SwiftUI与UIKit交互

Interfacing with UIKit

UIKit

View controllers

Interface Builder and storyboards

Delegates and data sources

SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY

UIKit

Delegates and data sources

SwiftUI与UIKit交互

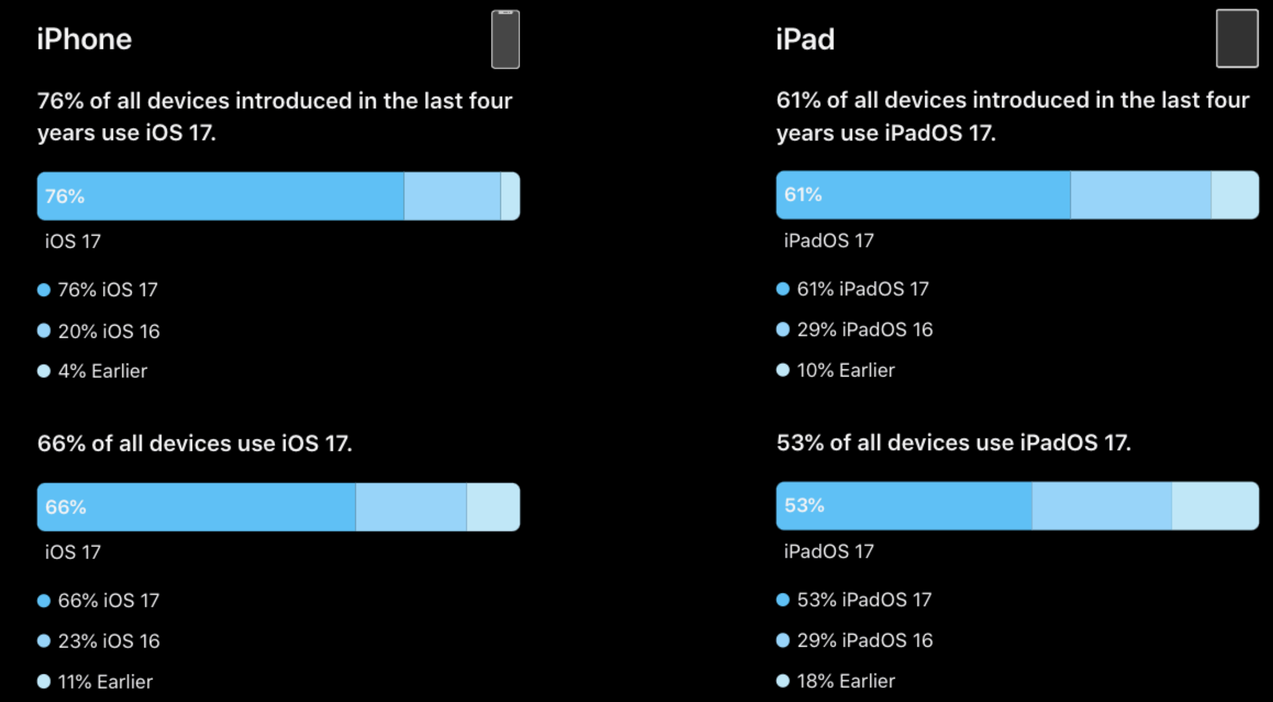
Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY

1. 考虑 iOS 版本的兼容性

Target iOS version and device compatibility

UIKit 支持 iOS 9 及之后的所有版本，
SwiftUI只支持 iOS 13 及之后的版本



SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学
ZHEJIANG UNIVERSITY

1. 考虑 iOS 版本的兼容性

Target iOS version and device compatibility

UIKit 支持 iOS 9 及之后的所有版本，SwiftUI只支持 iOS 13 及之后的版本

2. 项目复杂性和团队经验

Project complexity and team experience

UIKit 是成熟的框架，但是学习难度较大，SwiftUI易上手，但是还在更新迭代

SwiftUI与UIKit交互

Interfacing with UIKit

浙江大學
ZHEJIANG UNIVERSITY

1. 考虑 iOS 版本的兼容性

Target iOS version and device compatibility

UIKit 支持 iOS 9 及之后的所有版本，SwiftUI只支持 iOS 13 及之后的版本

2. 项目复杂性和团队经验

Project complexity and team experience

UIKit 是成熟的框架，但是学习难度较大，SwiftUI易上手，但是还在更新迭代

3. 项目所需的UI 组件和设计模式

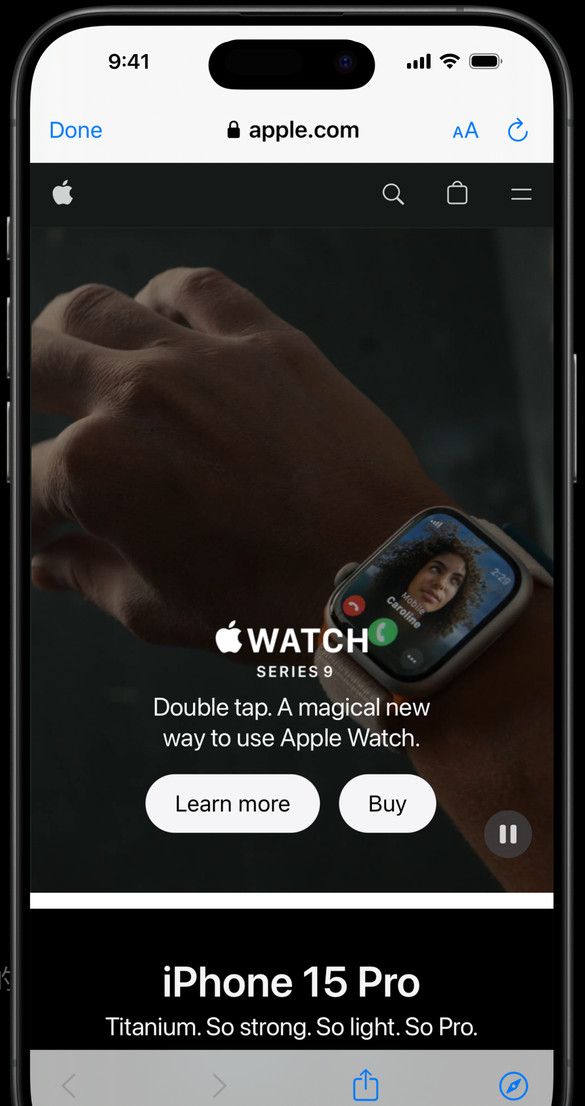
Required UI components and design patterns

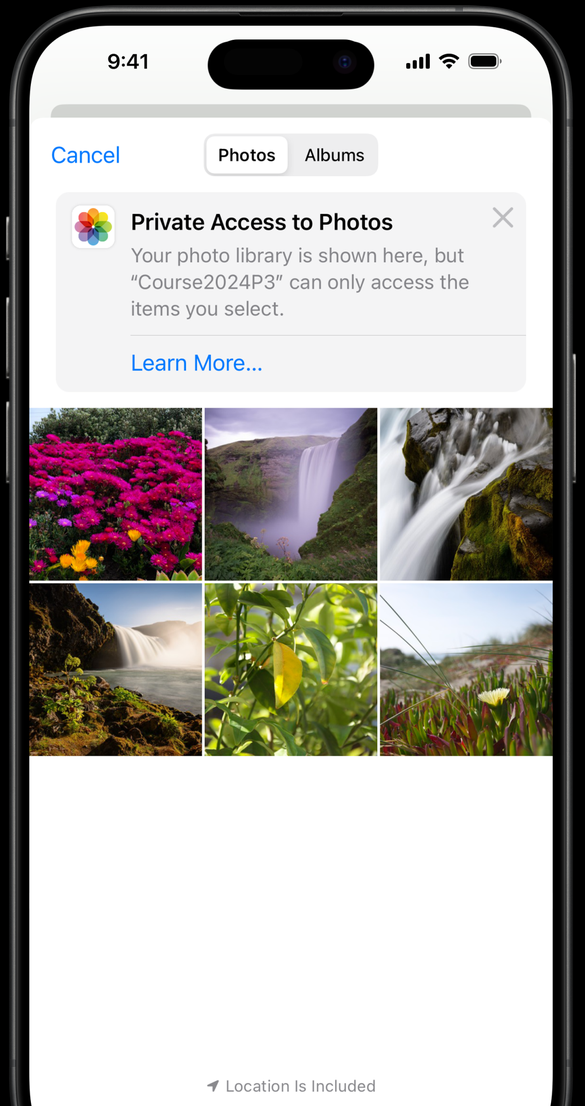
UIKit 具有大量 UI 组件（包括部分在SwiftUI上没有对应的组件），但在 SwiftUI 中可以使用 UIkit 的组件

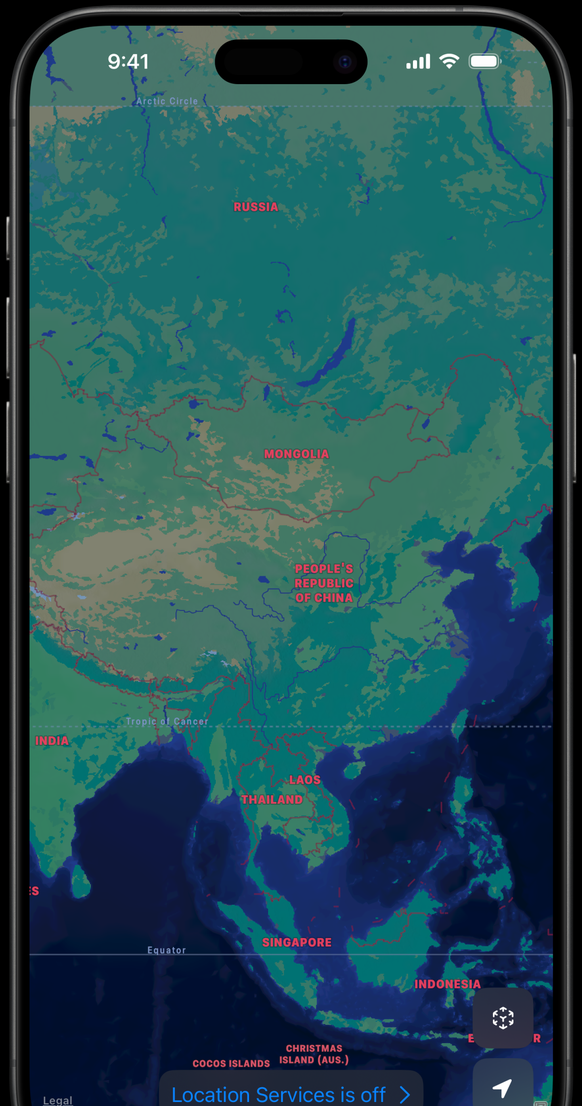
SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY

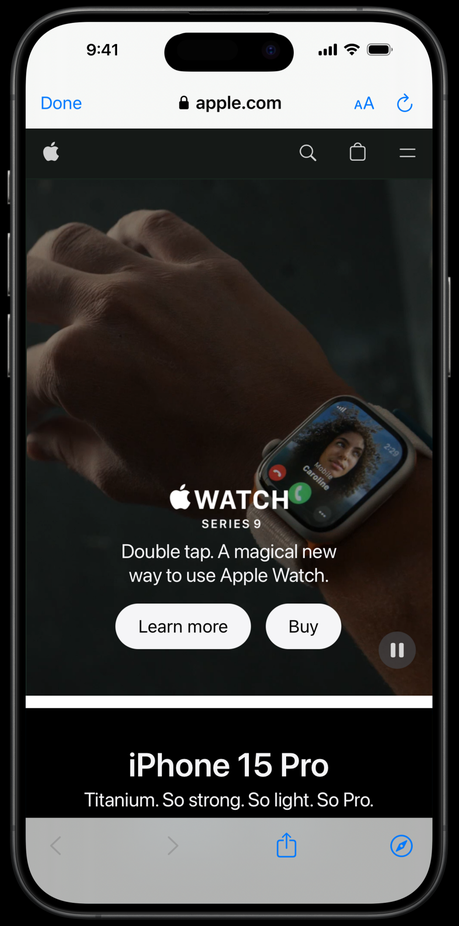






SwiftUI与UIKit交互

Interfacing with UIKit



iOS应用的数据模式与SwiftUI和UIKit交互

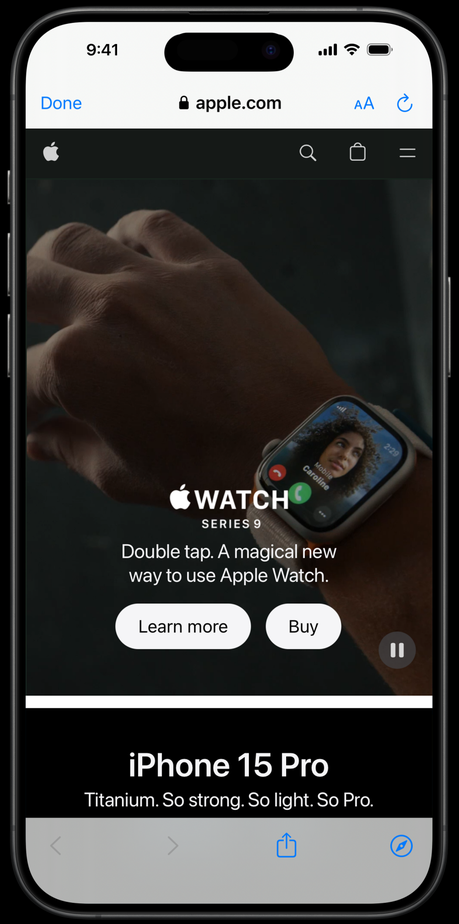
SFSafariViewController

iOS 9.0+ | iPadOS 9.0+ | Mac Catalyst 13.1+ | visionOS 1.0+

浙江大学 ZHEJIANG UNIVERSITY

SwiftUI与UIKit交互

Interfacing with UIKit



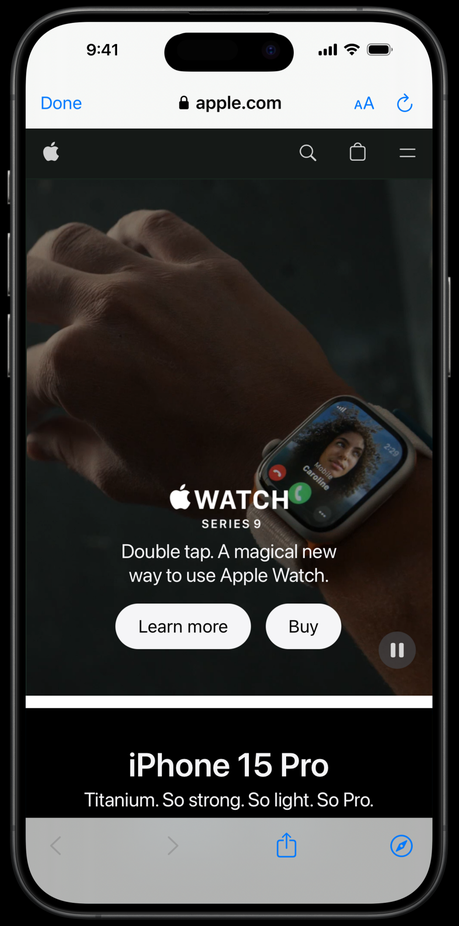
SFSafariViewController

Representable

iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit



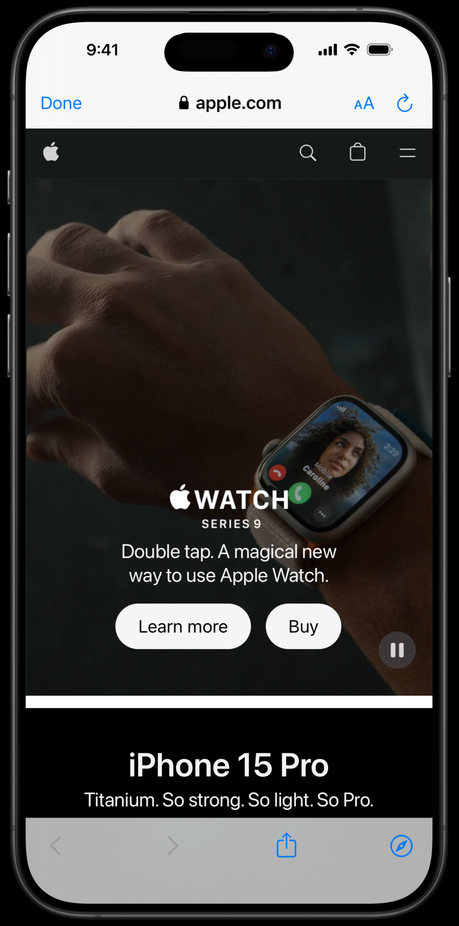

```
struct SFSafariViewControllerWrapper: UIViewControllerRepresentable {

}
```

Type ‘SFSafariViewControllerWrapper' does not conform to protocol ‘UIViewControllerRepresentable’

Do you want to add protocol stubs? Fix it

SwiftUI与UIKit交互

Interfacing with UIKit

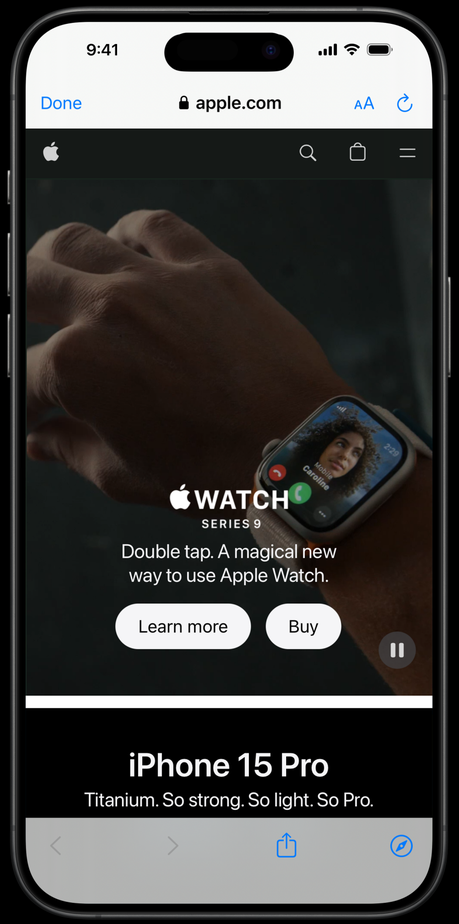


iOS应用的数据模式与SwiftUI和UIKit交互

```
struct SFSafariViewControllerWrapper: UIViewControllerRepresentable {

    typealias UIViewControllerType = type


}
```

SwiftUI与UIKit交互

Interfacing with UIKit




```
struct SFSafariViewControllerWrapper: UIViewControllerRepresentable {

    typealias UIViewControllerType = SFSafariViewController


}
```

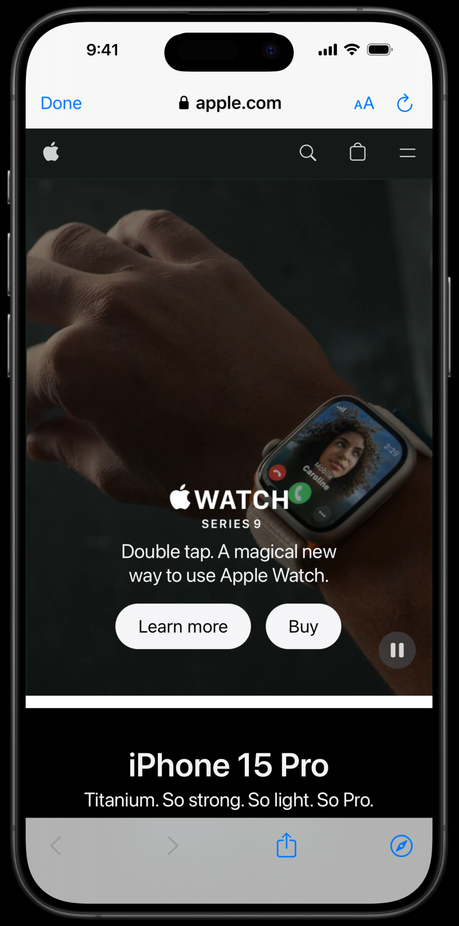
2 ✕

Type ‘SFSafariViewControllerWrapper' does not conform to protocol ‘UIViewControllerRepresentable’

Do you want to add protocol stubs? Fix it

SwiftUI与UIKit交互

Interfacing with UIKit

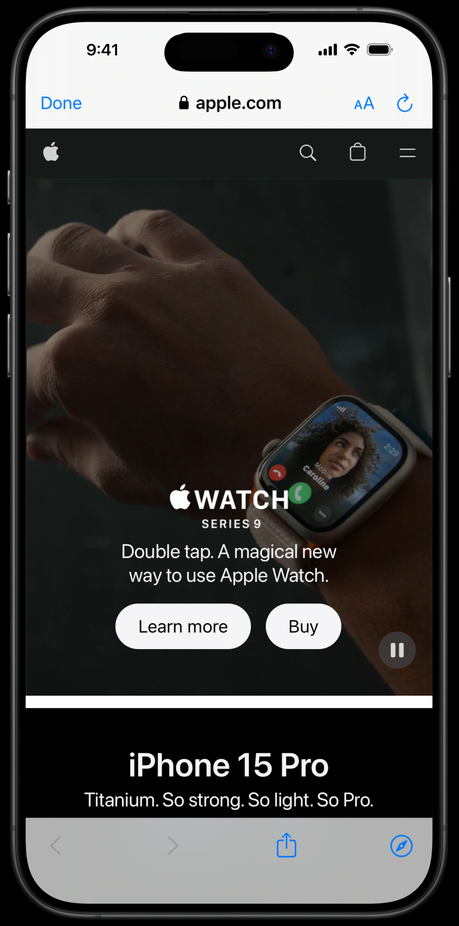



```
struct SFSafariViewControllerWrapper: UIViewControllerRepresentable {

    typealias UIViewControllerType = SFSafariViewController

    func makeUIViewController(context: Context)
                                        -> SFSafariViewController {
    }

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


}
```

iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

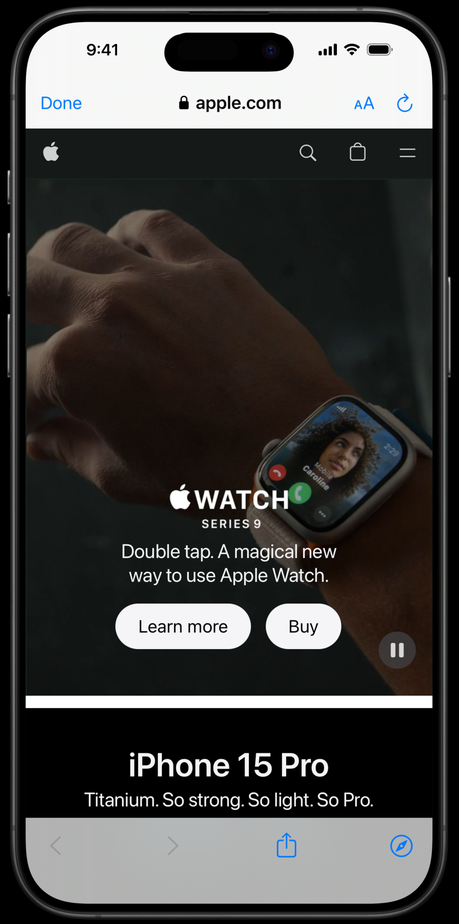



```
struct SFSafariViewControllerWrapper: UIViewControllerRepresentable {

    func makeUIViewController(context: Context)
                                        -> SFSafariViewController {
    }

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

}
```

iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit



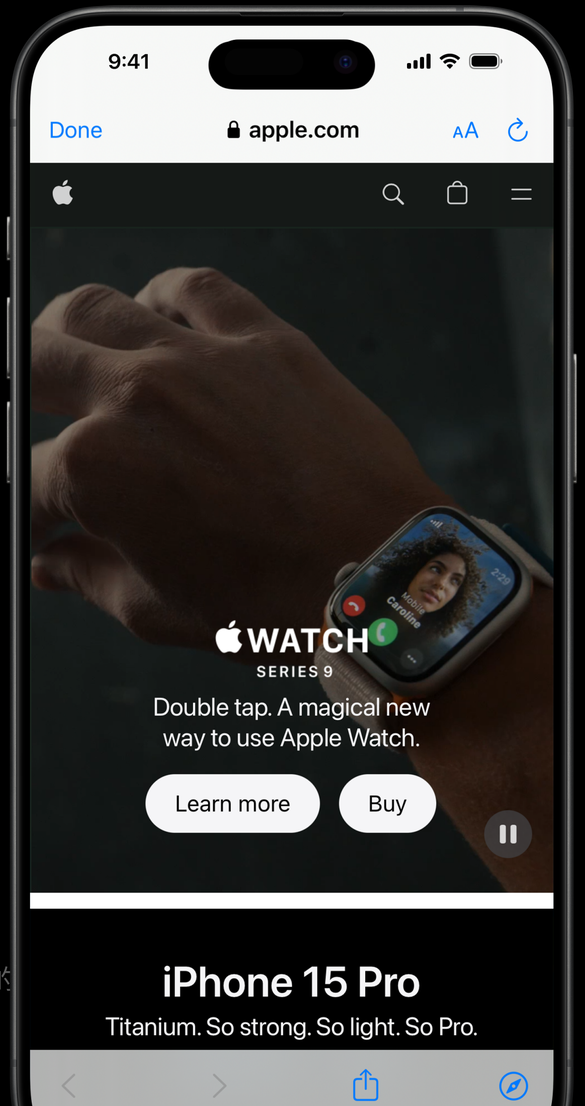

```
let url: URL
```

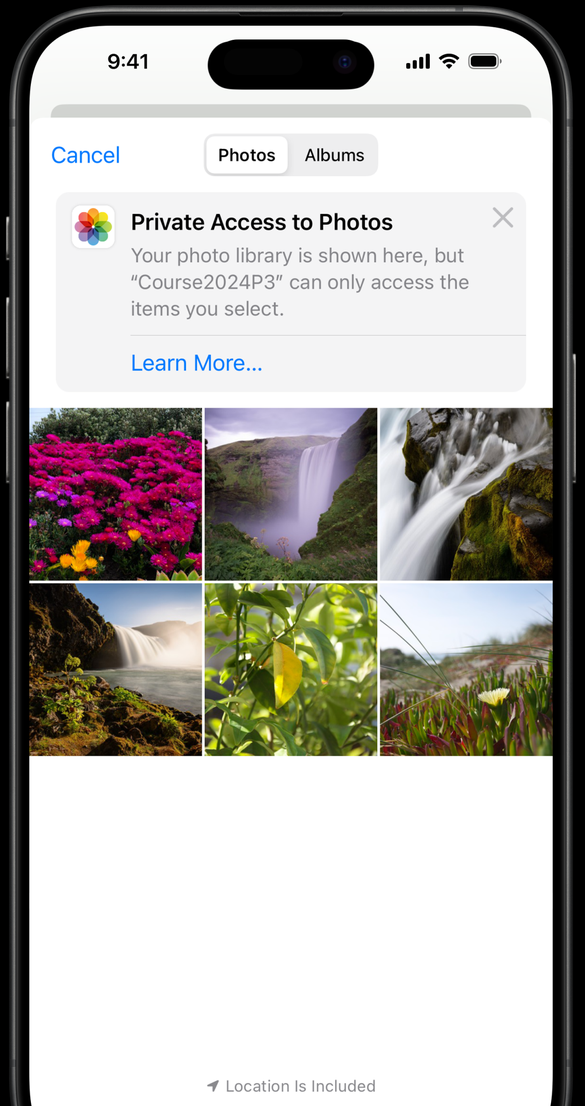
```
return SFSafariViewController(url: url)
```

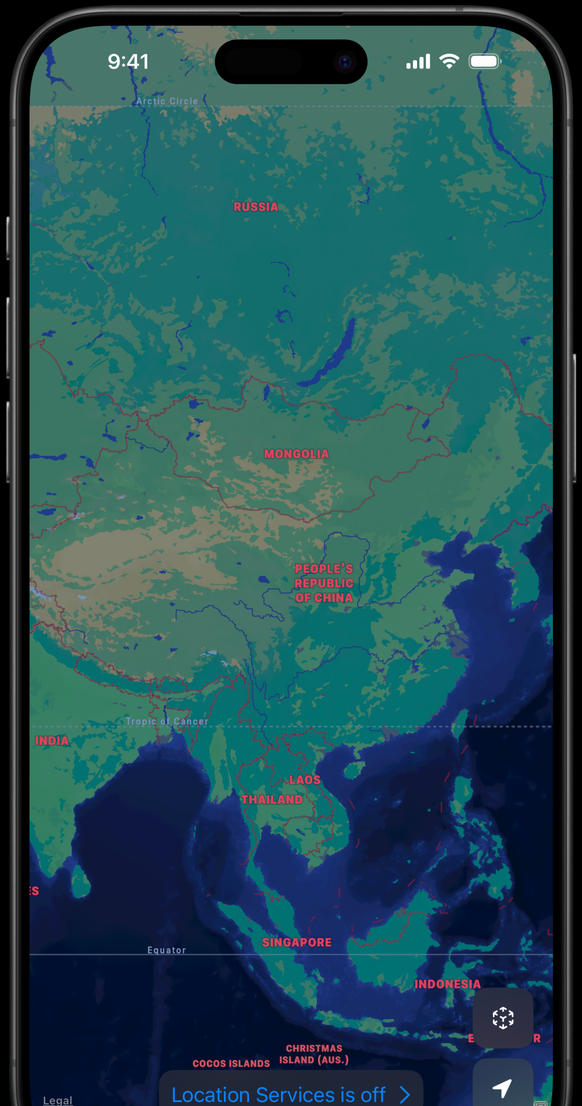
SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY

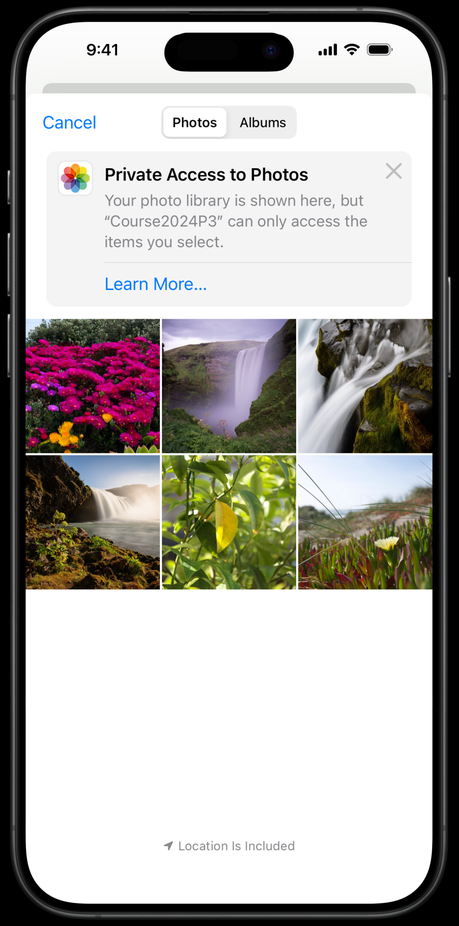






SwiftUI与UIKit交互

Interfacing with UIKit



iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

Joint **P**hotographic **E**xperts **G**roup

- standard image format
- containing lossy and compressed image data
- compressed but maintain reasonable image quality still

SwiftUI与UIKit交互

Interfacing with UIKit

Portable **N**etwork **G**raphic

- raster image file.
- can handle graphics with transparent backgrounds.

SwiftUI与UIKit交互

Interfacing with UIKit

High **E**fficiency **I**mage **F**ormat

- a modern photo file type
- similar quality to a JPEG but take less space.

SwiftUI与UIKit交互

Interfacing with UIKit

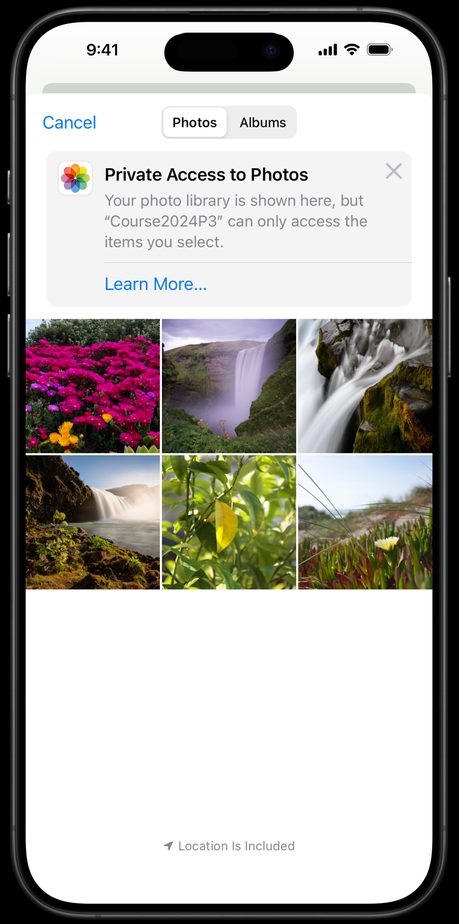
High **E**fficiency **I**mage **C**ontainer

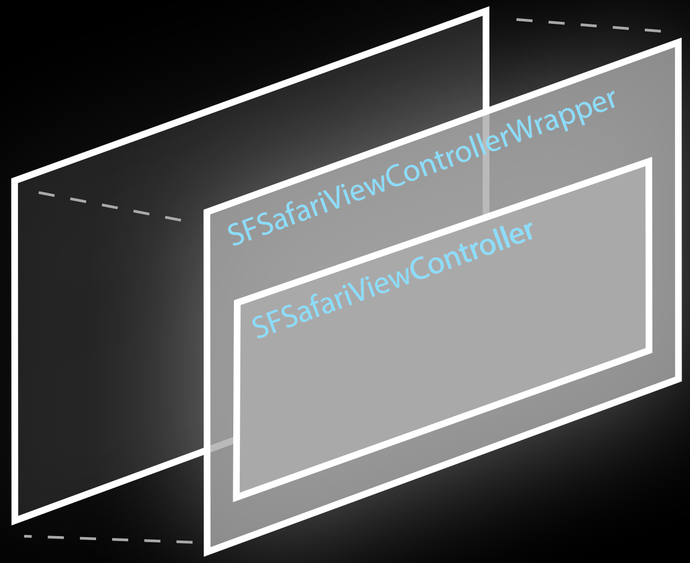
- It is Apple’s proprietary version of the HEIF
- smaller while retaining high quality.
- store the HEIF photo as well as additional data

  like sounds or motion when recording in a live photo.

SwiftUI与UIKit交互

Interfacing with UIKit

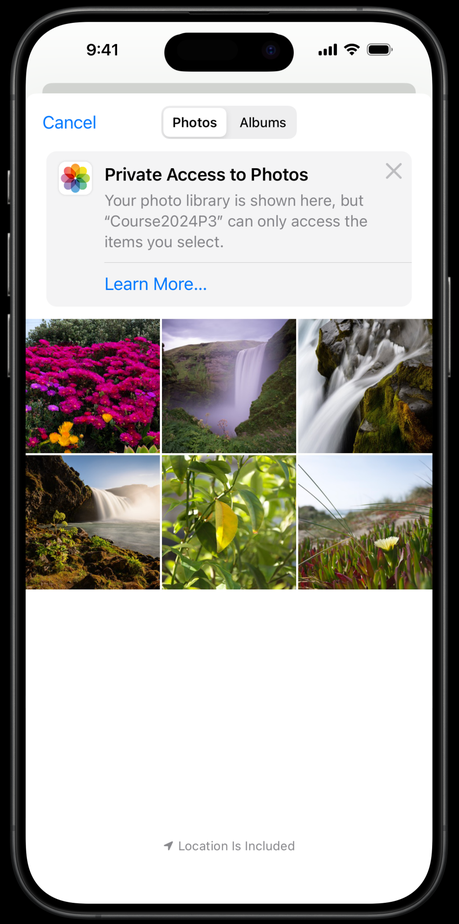


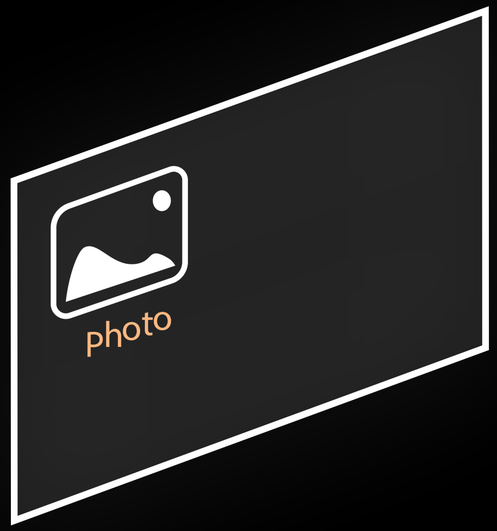


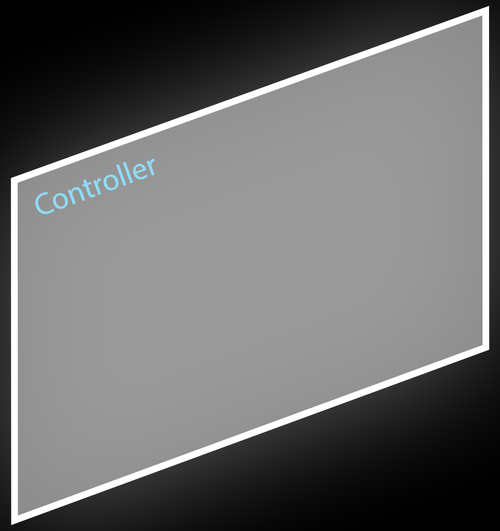
iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit



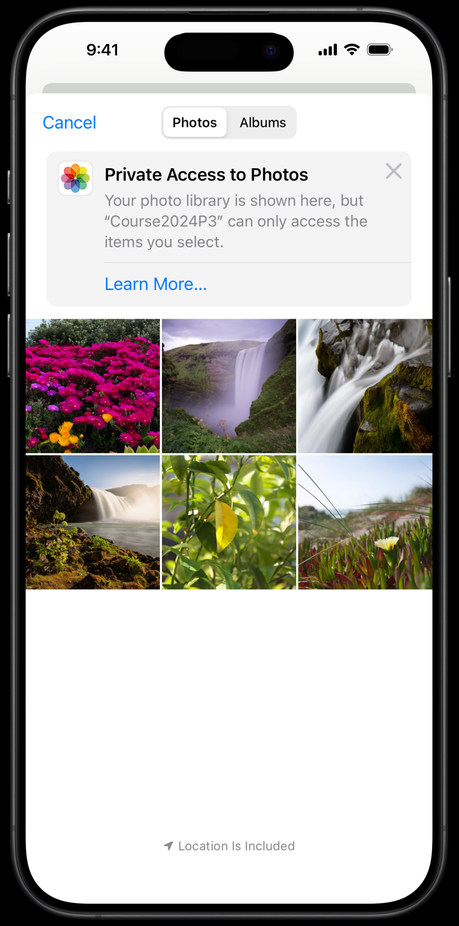


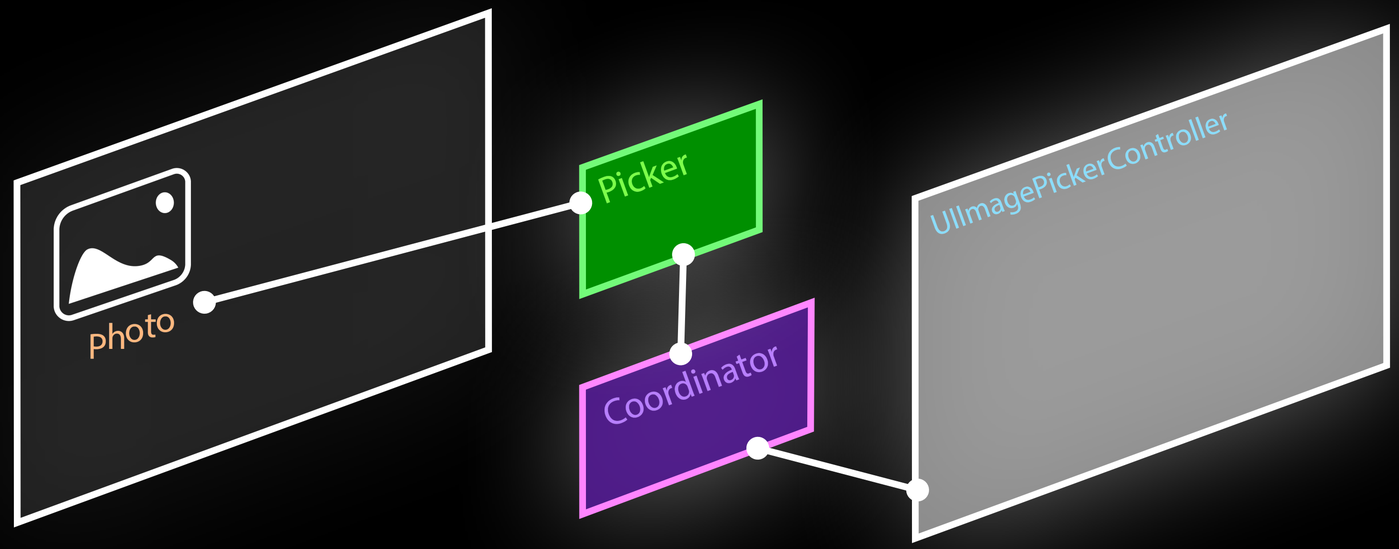


iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

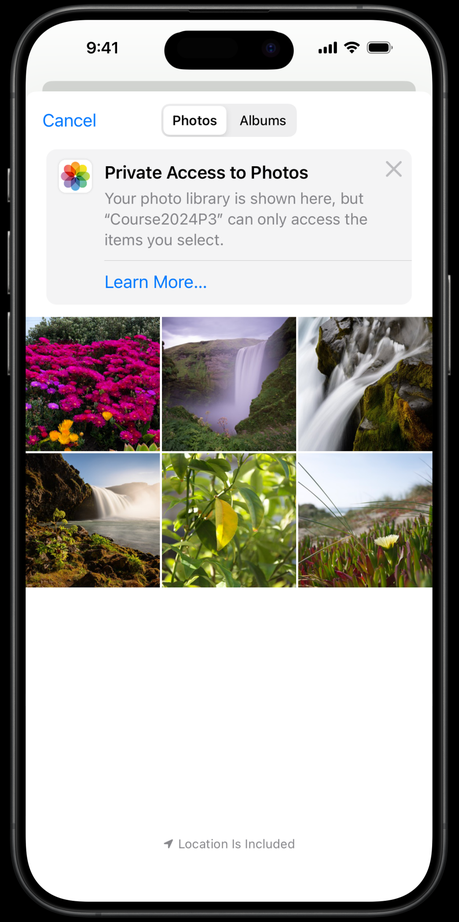




iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit



iOS应用的数据模式与SwiftUI和UIKit交互

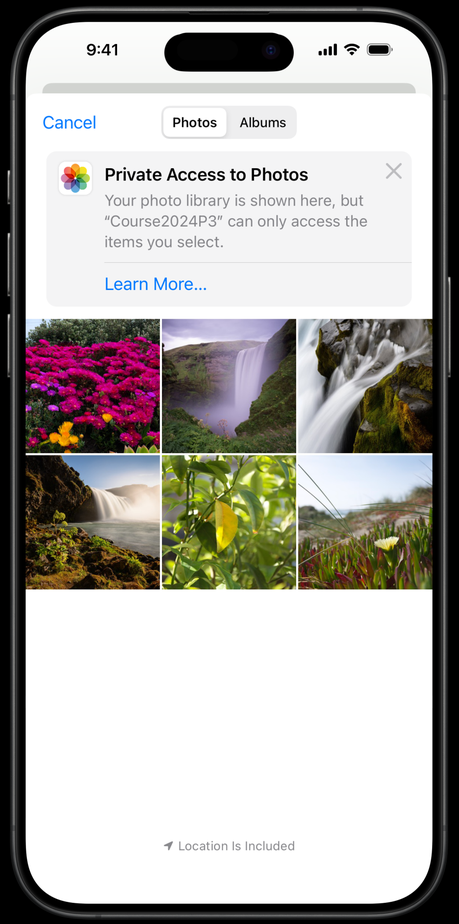
UIImagePickerViewController

iOS 2.0+ | iPadOS 2.0+ | Mac Catalyst 13.1+ | visionOS 1.0+

```
struct UIImagePickerControllerWrapper: UIViewControllerRepresentable{

    func makeUIViewController(context: Context) -> UIImagePickerController {

        return UIImagePickerController()

    }

    func updateUIViewController(_ uiViewController: UIImagePickerController,
                                context: Context) {

    }

}
```

SwiftUI与UIKit交互

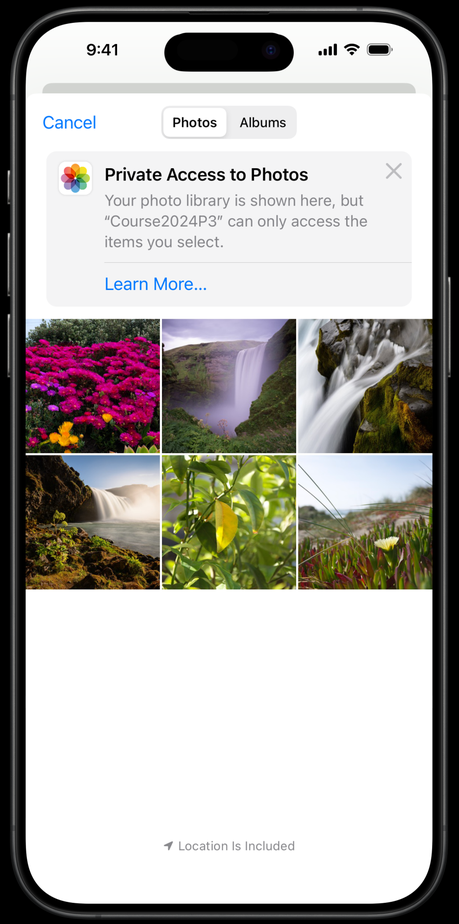
Interfacing with UIKit




```
func makeCoordinator() -> Coordinator {
    Coordinator()
}

class Coordinator{

}
```

SwiftUI与UIKit交互

Interfacing with UIKit



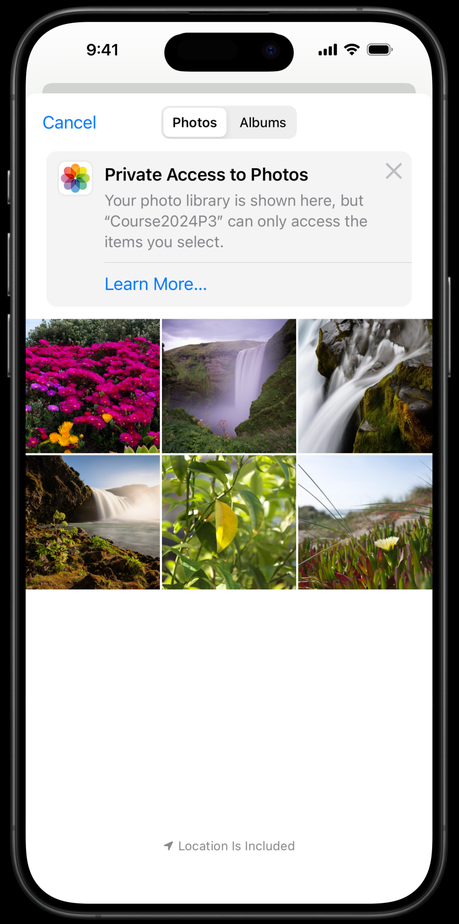
2

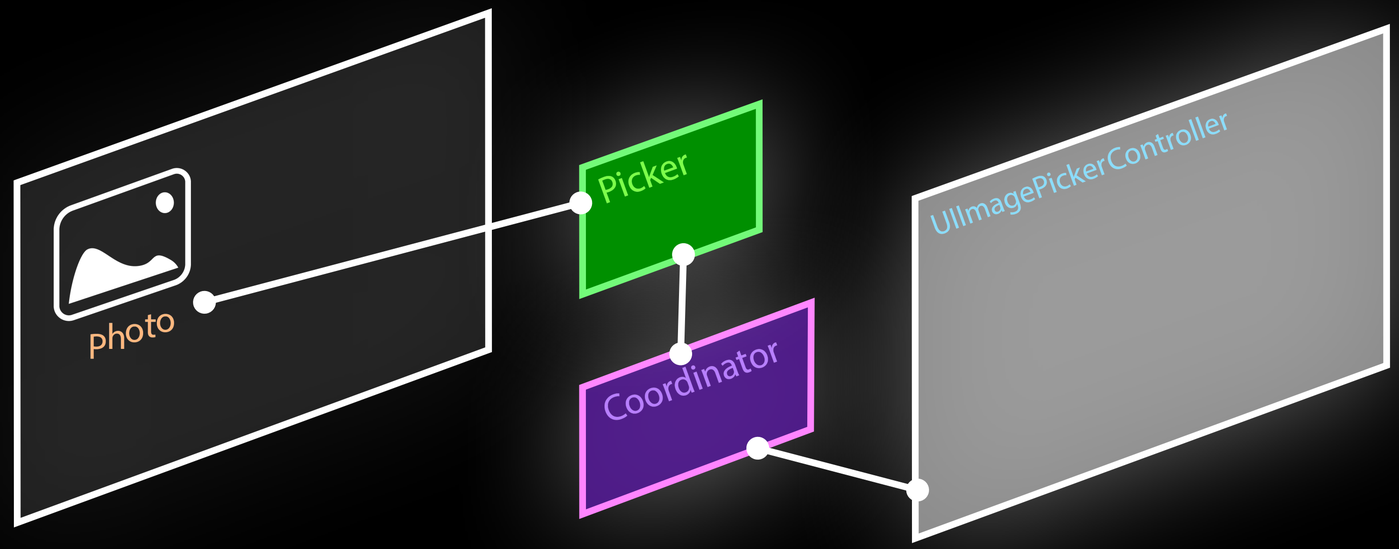
```swift
func makeCoordinator() -> Coordinator {
    Coordinator()
}

class Coordinator{

}
```

1

SwiftUI与UIKit交互

Interfacing with UIKit

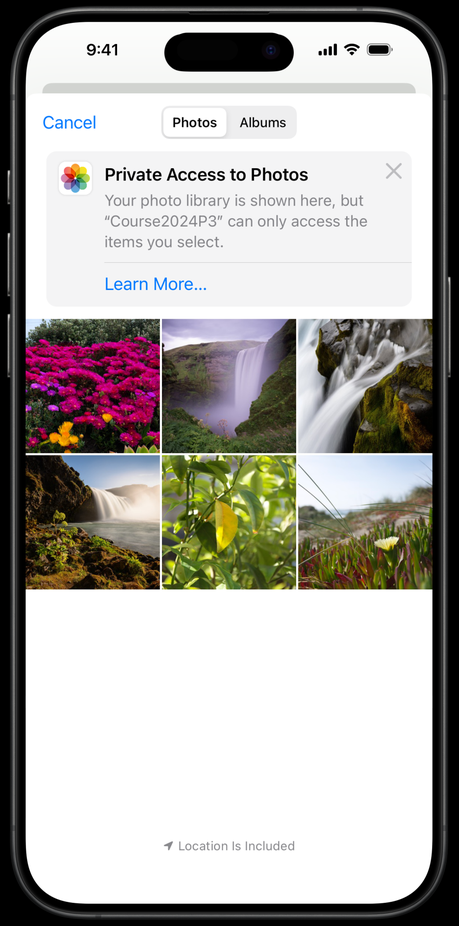




iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit




```
let picker = UIImagePickerController()
picker.delegate = context.coordinator
return picker
```

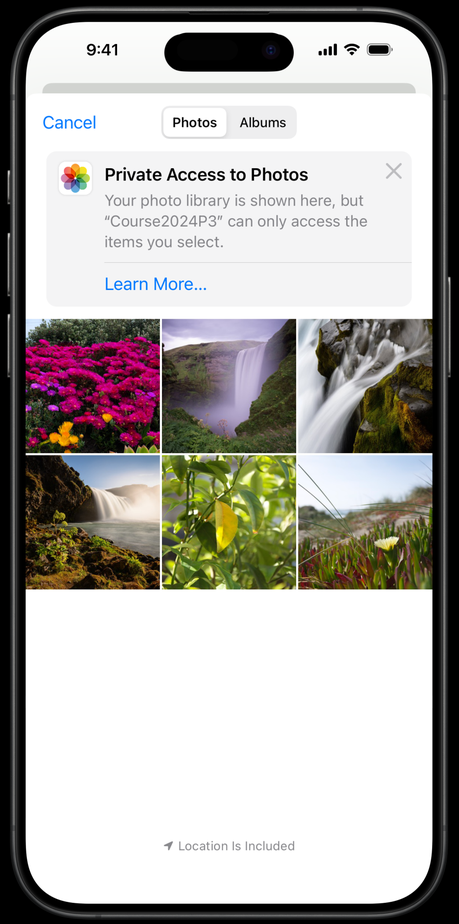
Cannot assign value of type 'ImagePicker.Coordinator' to type '(UIImagePickerControllerDelegate & UINavigationControllerDelegate)?'

Do you want to add protocol stubs?

Fix it

SwiftUI与UIKit交互

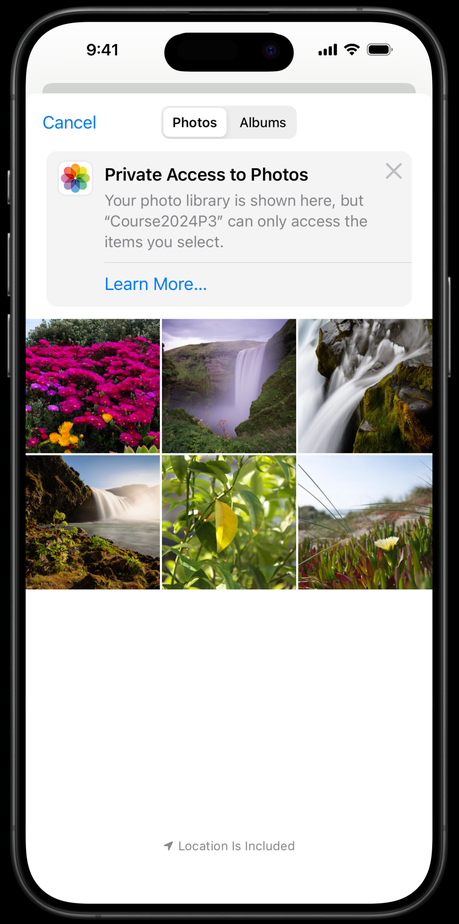
Interfacing with UIKit




```
class Coordinator: NSObject, UIImagePickerControllerDelegate,
                   UINavigationControllerDelegate {
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

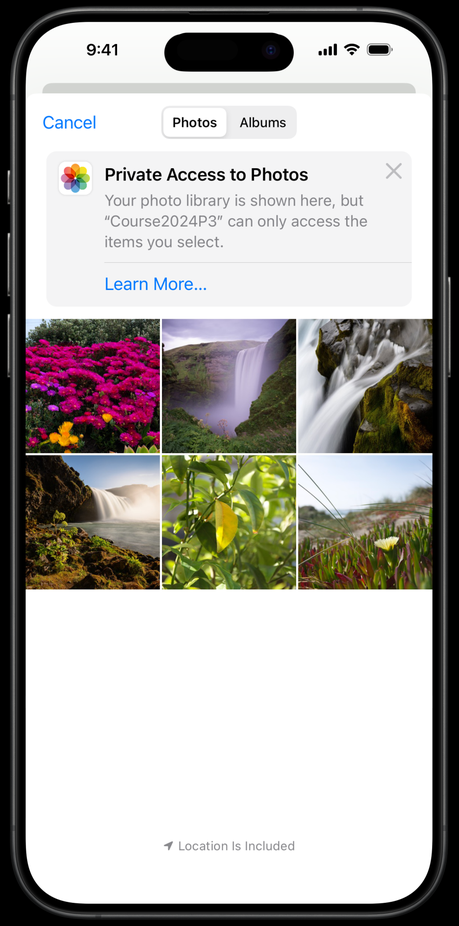


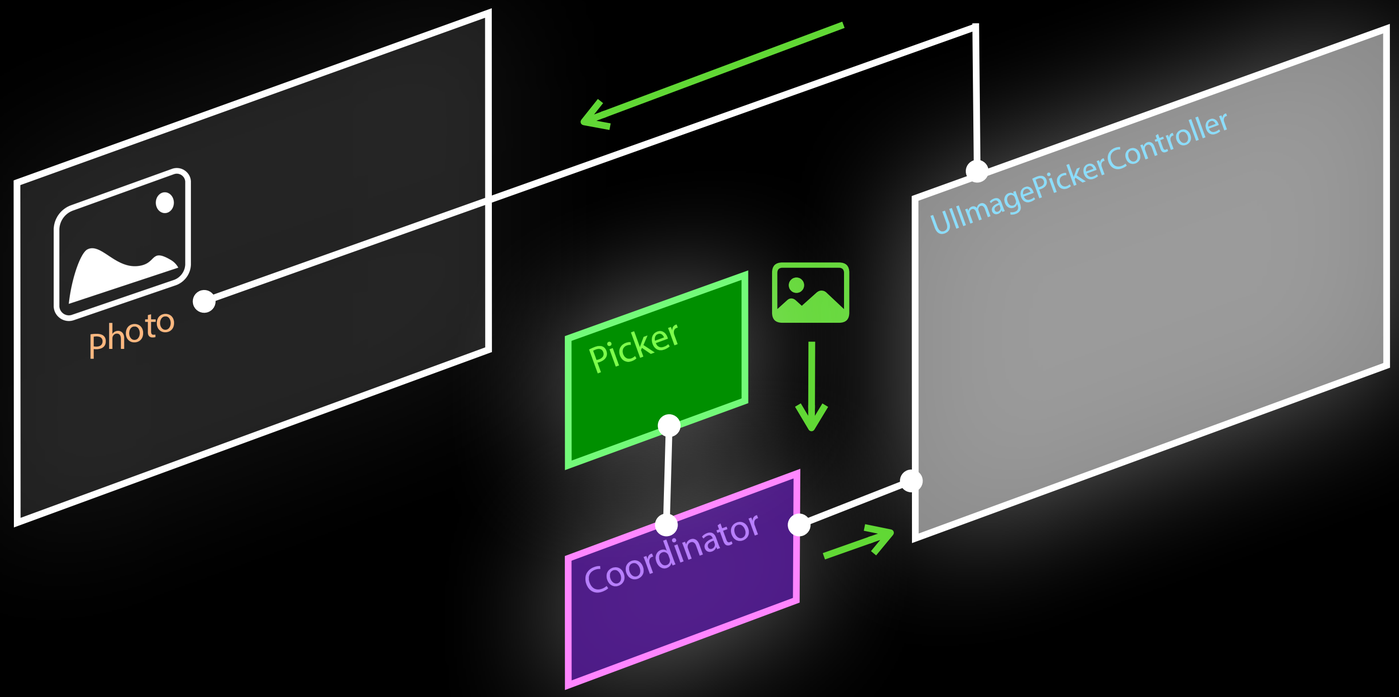

```
class Coordinator : NSObject, UIImagePickerControllerDelegate,
                    UINavigationControllerDelegate {

  func imagePickerController(_ picker: UIImagePickerController,
                             didFinishPickingMediaWithInfo info:
                             [UIImagePickerController.InfoKey : Any]) {
      if let image = info[UIImagePickerController.InfoKey.originalImage]
                                                    as? UIImage {

      }
  }

}
```

iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

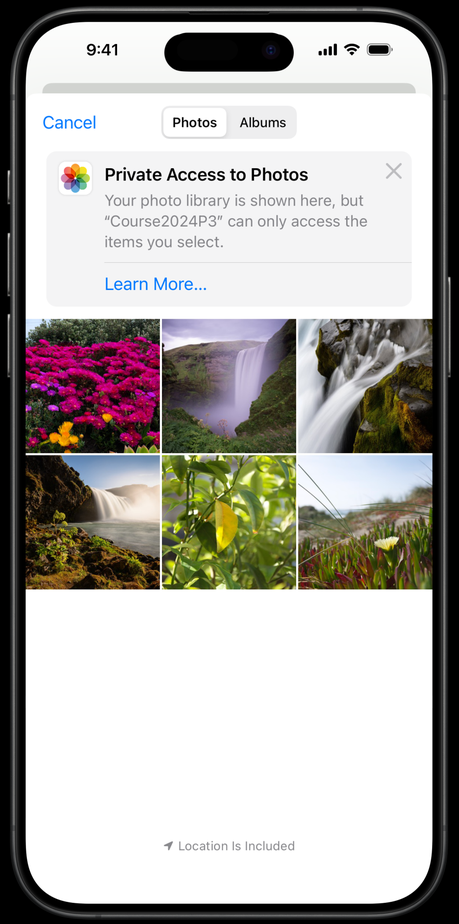




iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit



iOS应用的数据模式与SwiftUI和UIKit交互

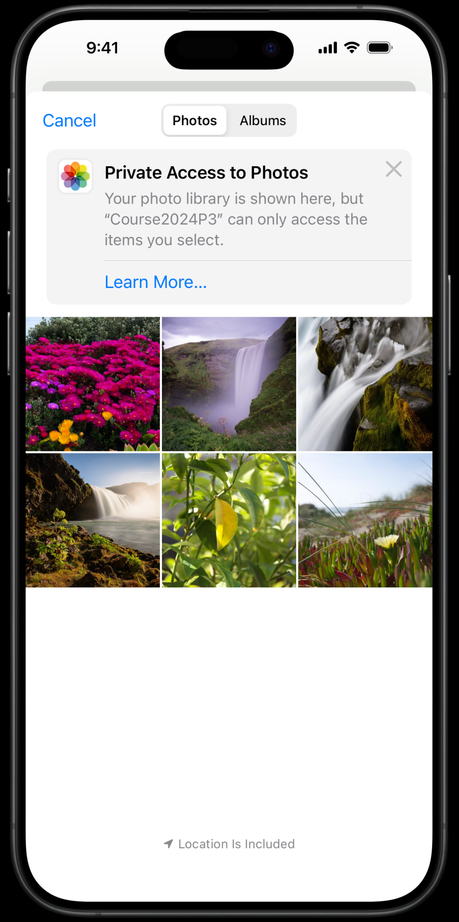
```
@Binding var selectedImage: UIImage?
```

```
self
```

```
var parent: UIImagePickerControllerWrapper

init(_ parent: UIImagePickerControllerWrapper) {
    self.parent = parent
}
    ...
}
```

SwiftUI与UIKit交互

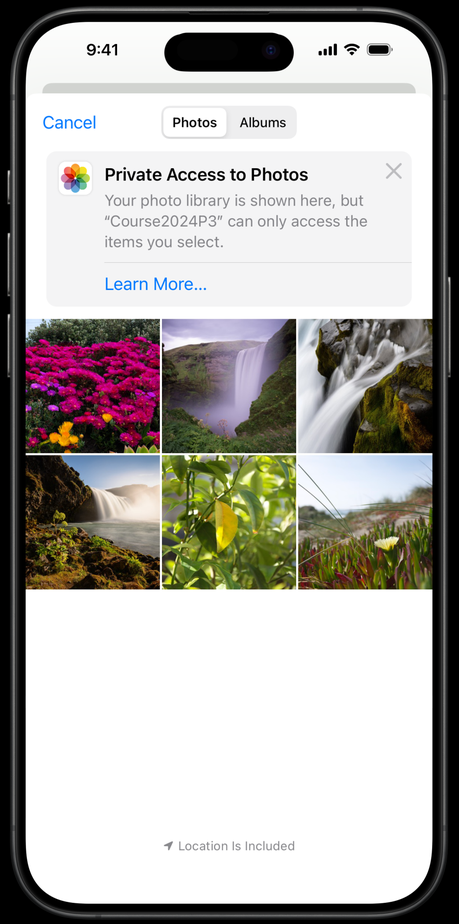
Interfacing with UIKit




```
struct PhotoPickerView: View {
    @State private var image: Image?
    @State private var showingImagePicker = false
    @State private var pickedImage: UIImage?

    var body: some View {
        VStack {
            image?
                .resizable()
                .scaledToFit()

            Button("Select Image") {
                showingImagePicker = true
            }
        }
        .sheet(isPresented: $showingImagePicker) {
            UIImagePickerControllerWrapper(selectedImage: $pickedImage)
                .ignoresSafeArea()
        }
    }

}
```

SwiftUI与UIKit交互

Interfacing with UIKit

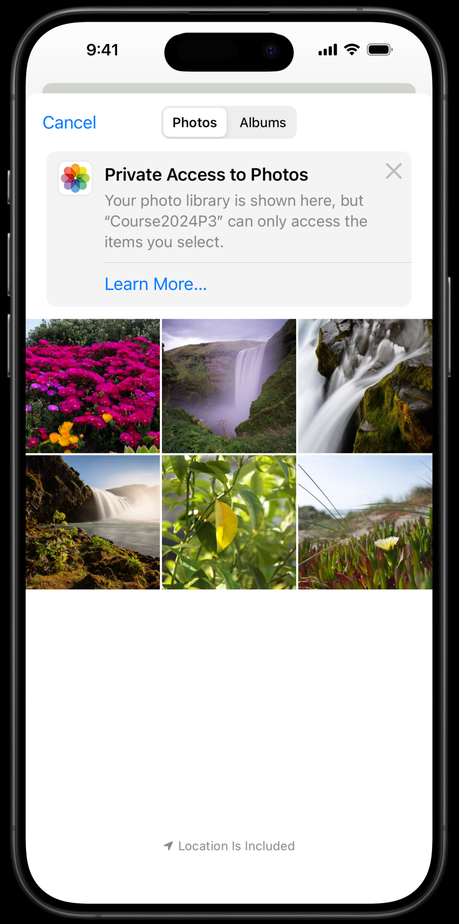



```
@State private var pickedImage: UIImage?
```

iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit



iOS应用的数据模式与SwiftUI和UIKit交互

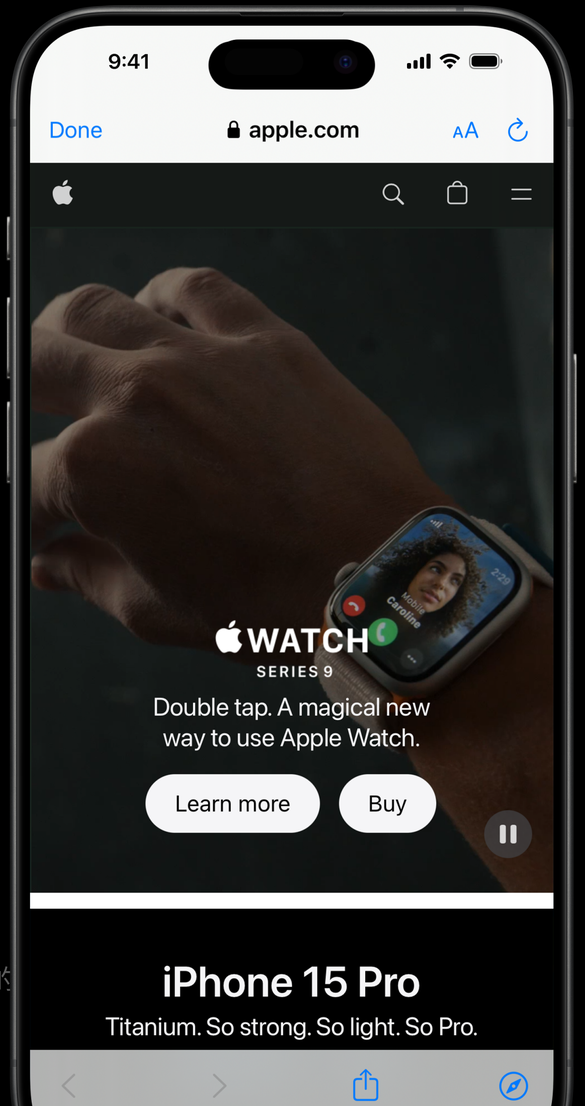
```
.onChange(of: pickedImage, { loadImage() })
```

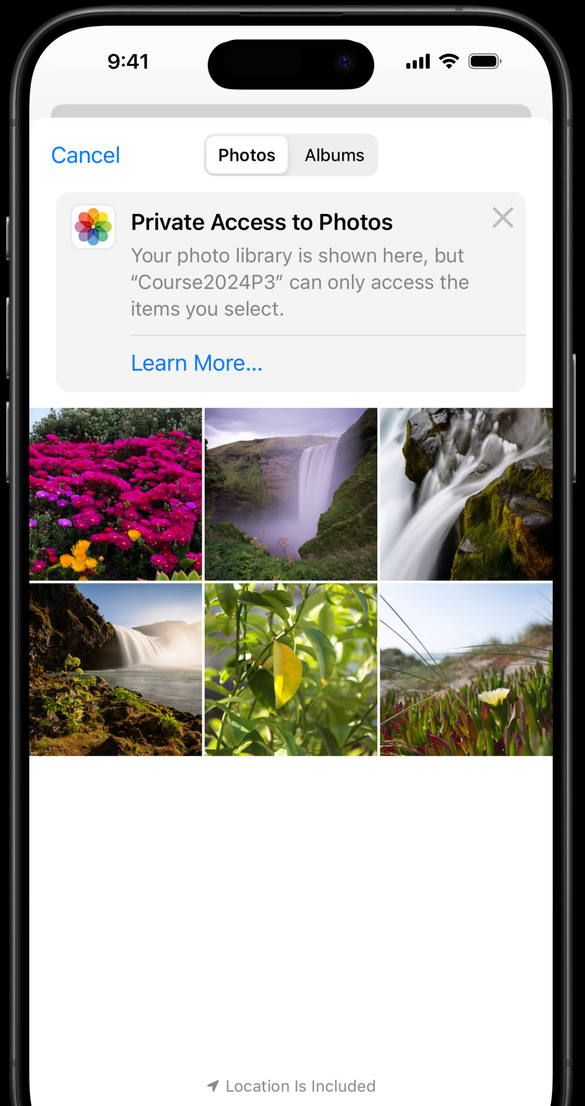
```
func loadImage(){
    guard let theImage = pickedImage else {return}
    image = Image(uiImage: theImage)
}
```

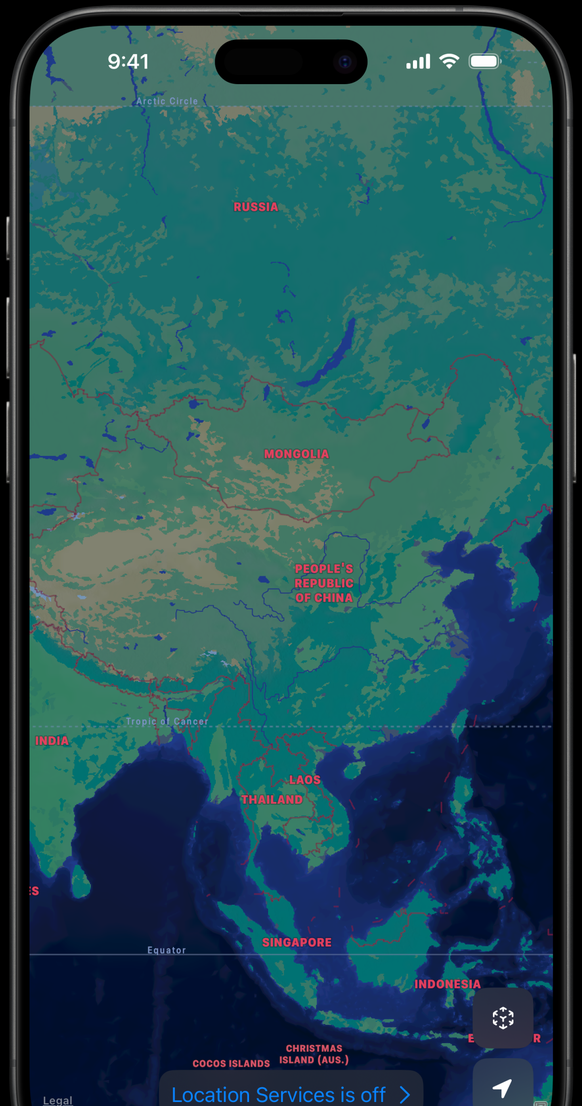
SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY







SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```swift
import MapKit

struct MKMapViewWrapper: UIViewRepresentable {

    func makeUIView(context: Context) -> MKMapView {
        let view = MKMapView()
        view.showsUserLocation = true

        return view
    }

    func updateUIView(_ uiView: MKMapView, context: Context) {

    }
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

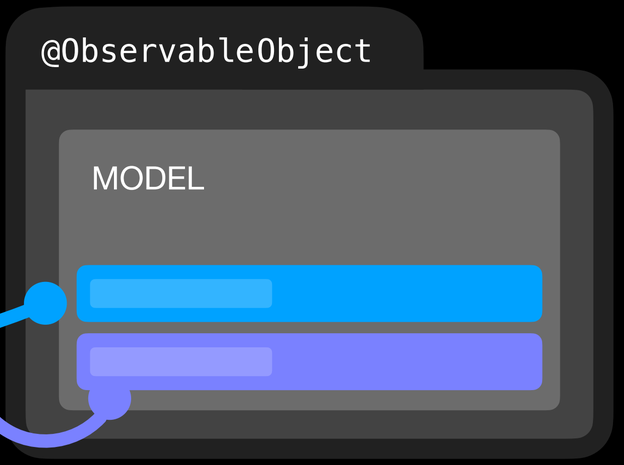
```swift
import MapKit

struct MKMapViewWrapper: UIViewRepresentable {

    func makeUIView(context: Context) -> MKMapView {
        let view = MKMapView()
        let locationManager = CLLocationManager()
        view.showsUserLocation = true

        return view
    }

    func updateUIView(_ uiView: MKMapView, context: Context) {

    }
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```swift
import MapKit

struct MKMapViewWrapper: UIViewRepresentable {

    func makeUIView(context: Context) -> MKMapView {
        let view = MKMapView()
        let locationManager = CLLocationManager()
        view.showsUserLocation = true

        return view
    }

    func updateUIView(_ uiView: MKMapView, context: Context) {

    }
}
```



SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel: ObservableObject {
    @Published var mapView = MKMapView()

    override init(){
        super.init()
    }

}
```

SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel:ObservableObject {
    @Published var mapView = MKMapView()

    override init(){
        super.init()
    }

}

struct MKMapViewWrapper: UIViewRepresentable {
    @EnvironmentObject var mapData: MapViewModel

    func makeUIView(context: Context) -> MKMapView {

        return mapData.mapView
    }

    func updateUIView(_ uiView: MKMapView, context: Context) {

    }
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

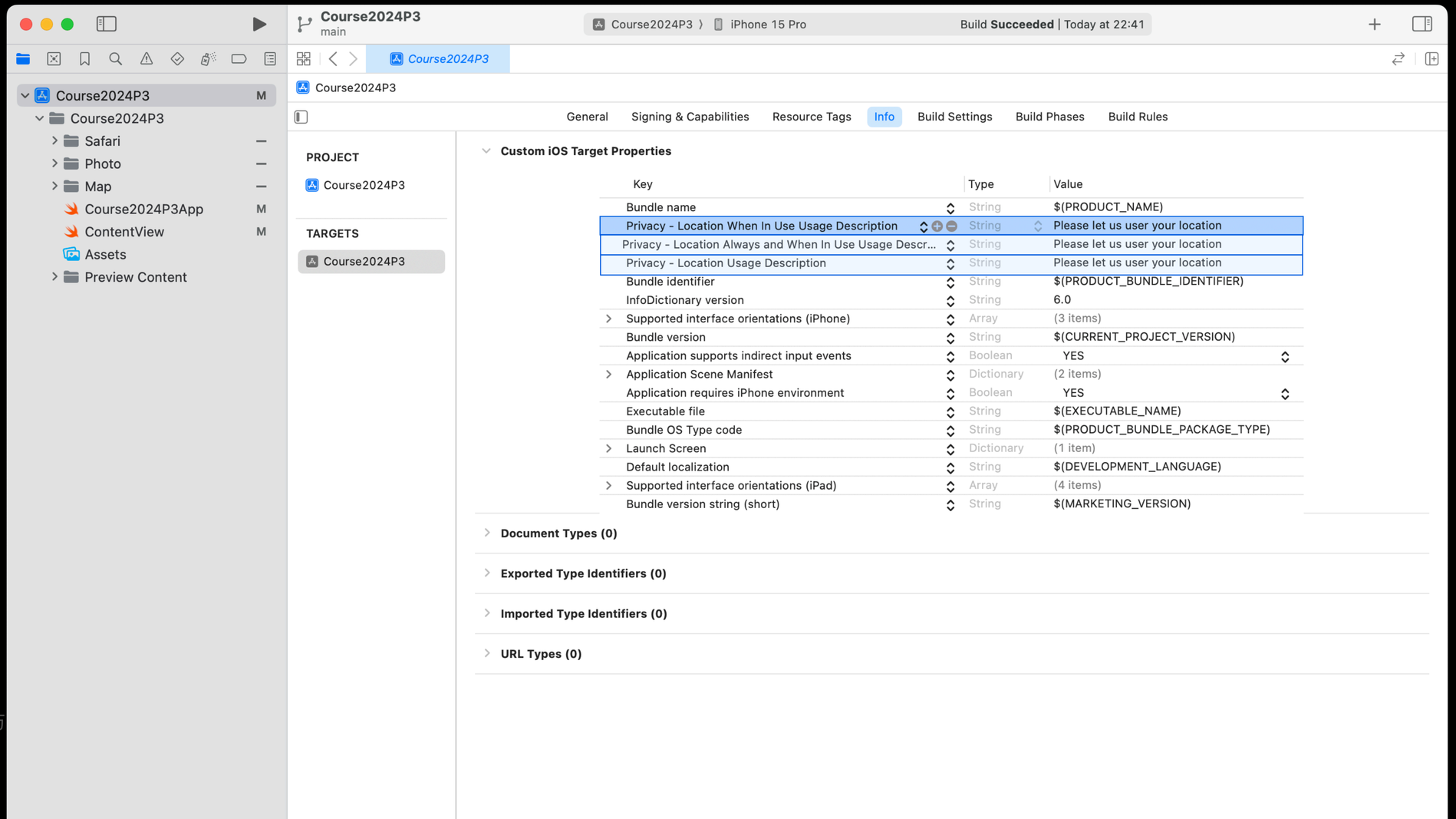
iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel:NSObject,ObservableObject,CLLocationManagerDelegate{
    @Published var mapView = MKMapView()

    @Published var locationManager = CLLocationManager()


    override init(){
        super.init()
        locationManager.delegate = self
    }

}
```

SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY



SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel: NSObject, ObservableObject, CLLocationManagerDelegate{
    @Published var mapView = MKMapView()
    @Published var locationManager = CLLocationManager()

    override init(){
        super.init()
        locationManager.delegate = self
    }

}
```

SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel: NSObject, ObservableObject, CLLocationManagerDelegate{
    @Published var mapView = MKMapView()

    @Published var locationManager = CLLocationManager()

    @Published var permissionDenied = true

    override init(){
        super.init()
        locationManager.delegate = self
    }

    func locationManagerDidChangeAuthorization(_ manager: CLLocationManager) {
         switch manager.authorizationStatus {
            case .notDetermined:
               manager.requestWhenInUseAuthorization()
            case .authorizedWhenInUse:
               manager.requestLocation()
            case .authorizedAlways:
                permissionDenied = false
            @unknown default:
                // print("Unknown error")
    }
}
```

SwiftUI与UIKit交互

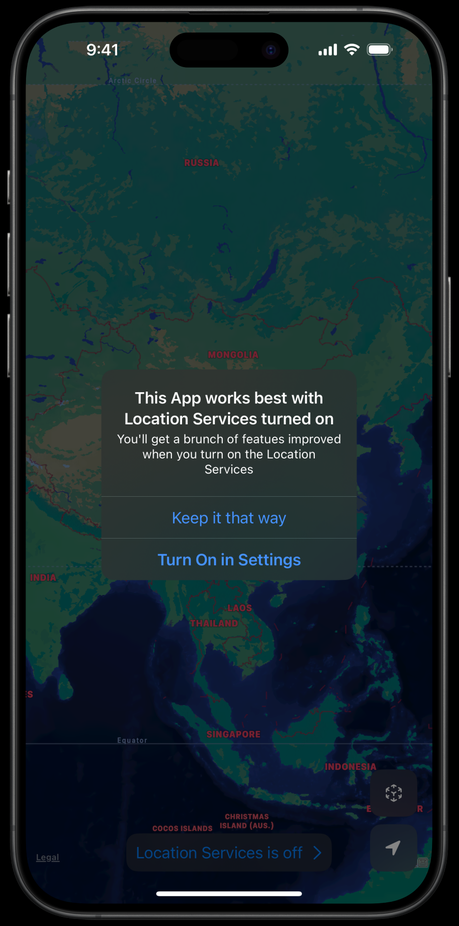
Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel: NSObject, ObservableObject, CLLocationManagerDelegate{
    @Published var mapView = MKMapView()

    @Published var locationManager = CLLocationManager()

    @Published var permissionDenied = true

    override init(){
        super.init()
        locationManager.delegate = self
    }

    func locationManagerDidChangeAuthorization(_ manager: CLLocationManager) {
         switch manager.authorizationStatus {
            case .notDetermined:
               manager.requestWhenInUseAuthorization()
            case .authorizedWhenInUse:
               manager.requestLocation()
            case .authorizedAlways:
                permissionDenied = false
            @unknown default:
                // print("Unknown error")
    }
}
```

SwiftUI与UIKit交互

Interfacing with UIKit



iOS应用的数据模式与SwiftUI和UIKit交互

```
Button···

 .alert("This App works best with Location Services turned on", isPresented: $isShowAlert){
      Button(role: .none){
      }label:{
          Text("Keep it that way")
      }
      Button(role: .cancel){
          UIApplication.shared.open(URL(string: UIApplication.openSettingsURLString)!)
      }label:{
           Text("Turn On in Settings")
      }
 } message: {
        Text("You'll get a brunch of featues improved when you turn on the Location Services")
 }
```

SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
class MapViewModel: NSObject, ObservableObject, CLLocationManagerDelegate{
    @Published var mapView = MKMapView()

    @Published var locationManager = CLLocationManager()

    @Published var permissionDenied = true

    override init(){
        super.init()
        locationManager.delegate = self
    }

    func locationManagerDidChangeAuthorization(_ manager: CLLocationManager) {
         switch manager.authorizationStatus {
            case .notDetermined:
               manager.requestWhenInUseAuthorization()
            case .authorizedWhenInUse:
               manager.requestLocation()
            case .authorizedAlways:
                permissionDenied = false
            @unknown default:
                // print("Unknown error")
    }
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```
@Published var region: MKCoordinateRegion!



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

func locationManager(_ manager: CLLocationManager,
                     didUpdateLocations locations: [CLLocation]){
     guard let location = locations.last else { return }
     self.region = MKCoordinateRegion(center: location2D,
                                      span: MKCoordinateSpan(latitudeDelta: 0.2,
                                                             longitudeDelta: 0.2))
     self.mapView.setRegion(self.region, animated: true)
     self.mapView.setVisibleMapRect(self.mapView.visibleMapRect, animated: true)
 }
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

Search

Annotations

Direction

Geocoding

Snapshots

Selectable

iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY

Annotations

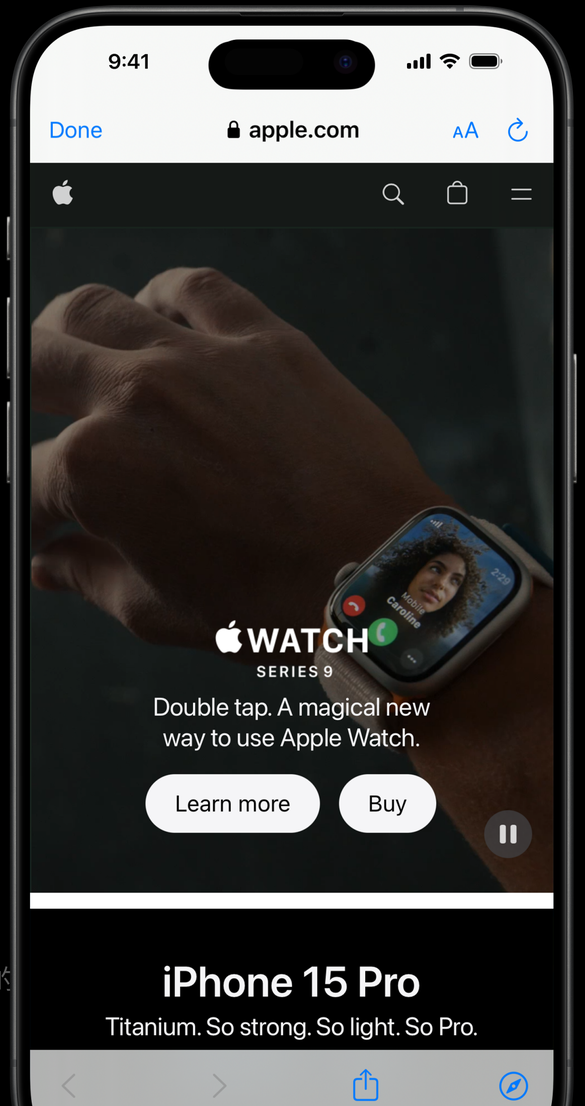
Snapshots

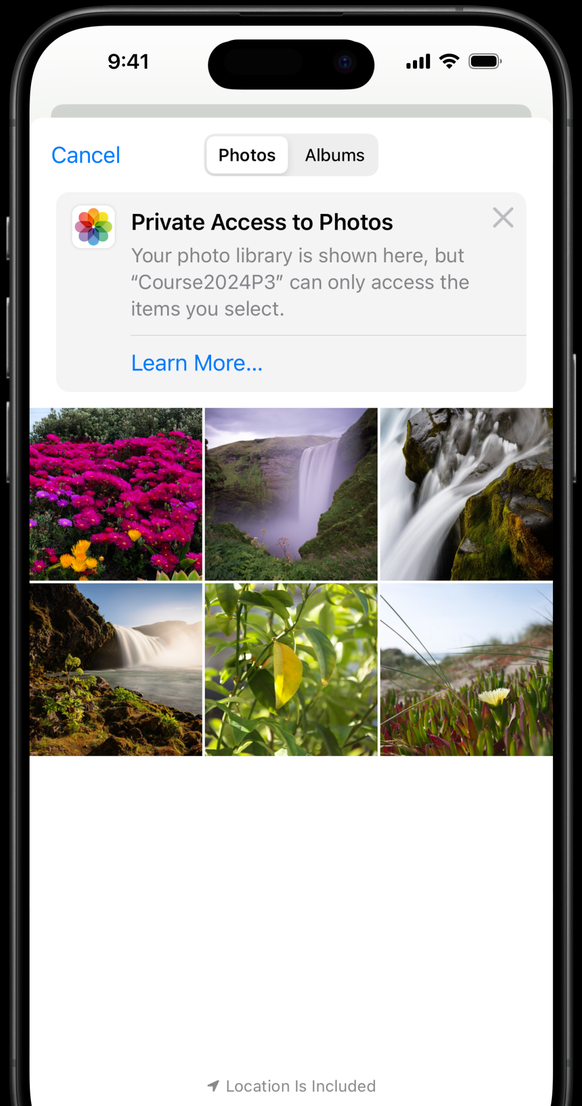
iOS应用的数据模式与SwiftUI和UIKit交互

SwiftUI与UIKit交互

Interfacing with UIKit

浙江大学 ZHEJIANG UNIVERSITY





SwiftUI与UIKit交互

Interfacing with UIKit

iOS应用的数据模式与SwiftUI和UIKit交互

```swift
struct NewMapView: View {
    @State var position: MapCameraPosition =
                                .camera(MapCamera(centerCoordinate: .meetingPoint,
                                                  distance: 300,
                                                  pitch: 30))

    var body: some View {
        Map(position: $position){
            Marker("Metting Point", coordinate: .meetingPoint)
        }
        .mapStyle(.standard(elevation: .realistic))
    }

}

extension CLLocationCoordinate2D{
    static let meetingPoint = CLLocationCoordinate2D(latitude: 42.3528,
                                                     longitude: -71.068369)
}
```

SwiftUI与UIKit交互

Interfacing with UIKit

WWDC 23: https://developer.apple.com/videos/play/wwdc2023/10043/

浙江大学
ZHEJIANG UNIVERSITY

Any Questions ?