Protocol

App

A type that represents the structure and behavior of an app.

iOS 14.0+ | iPadOS 14.0+ | Mac Catalyst 14.0+ | macOS 11.0+ | tvOS 14.0+ | visionOS 1.0+ | watchOS 7.0+

```
@MainActor @preconcurrency
protocol App
```

Mentioned in

Migrating to the SwiftUI life cycle

Overview

Create an app by declaring a structure that conforms to the App protocol. Implement the required body computed property to define the app's content:

```
@main
struct MyApp: App {
    var body: some Scene {
        WindowGroup {
            Text("Hello, world!")
        }
    }
}
```

Precede the structure's declaration with the @main attribute to indicate that your custom App protocol conformer provides the entry point into your app. The protocol provides a default

implementation of the `main()` method that the system calls to launch your app. You can have exactly one entry point among all of your app's files.

Compose the app's body from instances that conform to the `Scene` protocol. Each scene contains the root view of a view hierarchy and has a life cycle managed by the system. SwiftUI provides some concrete scene types to handle common scenarios, like for displaying documents or settings. You can also create custom scenes.

```
@main
struct Mail: App {
    var body: some Scene {
        WindowGroup {
            MailViewer()
        }
        Settings {
            SettingsView()
        }
    }
}
```

You can declare state in your app to share across all of its scenes. For example, you can use the `StateObject` attribute to initialize a data model, and then provide that model on a view input as an `ObservedObject` or through the environment as an `EnvironmentObject` to scenes in the app:

```
@main
struct Mail: App {
    @StateObject private var model = MailModel()

    var body: some Scene {
        WindowGroup {
            MailViewer()
                .environmentObject(model) // Passed through the environment.
        }
        Settings {
            SettingsView(model: model) // Passed as an observed object.
        }
    }
}
```

A type conforming to this protocol inherits `@preconcurrency @MainActor` isolation from the protocol if the conformance is included in the type's base declaration:

```
struct MyCustomType: Transition {
    // `@preconcurrency @MainActor` isolation by default
}
```

Isolation to the main actor is the default, but it's not required. Declare the conformance in an extension to opt out of main actor isolation:

```
extension MyCustomType: Transition {
    // `nonisolated` by default
}
```

Topics

Implementing an app

`var body: Self.Body`

The content and behavior of the app.

Required

`associatedtype Body : Scene`

The type of scene representing the content of the app.

Required

Running an app

`init()`

Creates an instance of the app using the body that you define for its content.

Required

`static func main()`

Initializes and runs the app.

See Also

Creating an app

{} Destination Video

Leverage SwiftUI to build an immersive media experience in a multiplatform app.

{} Hello World

Use windows, volumes, and immersive spaces to teach people about the Earth.

{} Backyard Birds: Building an app with SwiftData and widgets

Create an app with persistent data, interactive widgets, and an all new in-app purchase experience.

{} Food Truck: Building a SwiftUI multiplatform app

Create a single codebase and app target for Mac, iPad, and iPhone.

{} Fruta: Building a feature-rich app with SwiftUI

Create a shared codebase to build a multiplatform app that offers widgets and an App Clip.

Migrating to the SwiftUI life cycle

Use a scene-based life cycle in SwiftUI while keeping your existing codebase.