

Intro to SwiftUI for macOS



Ryan Ball
Director of Engineering



Electrona



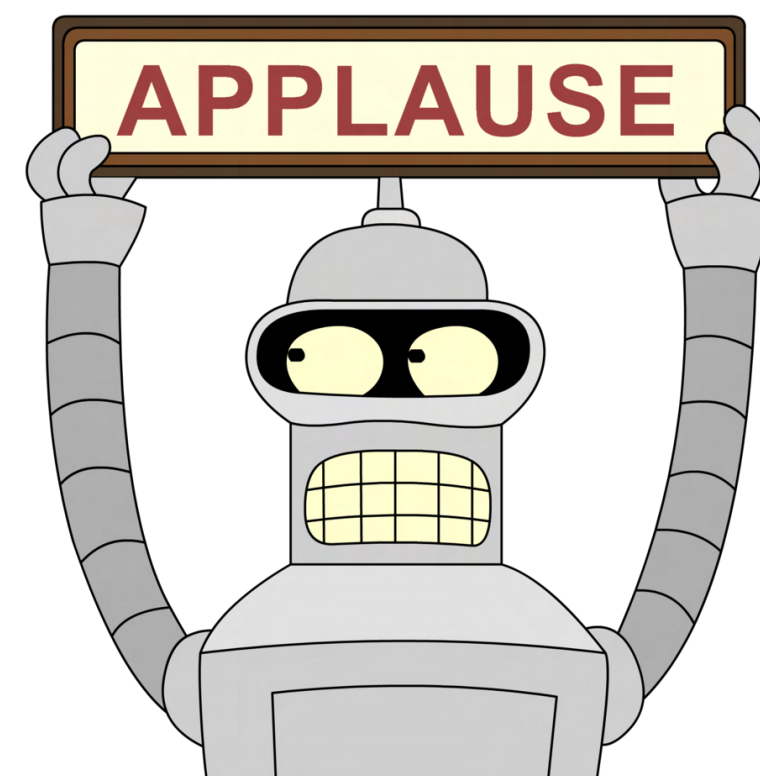
SwiftUI

Better apps. Less code.



SwiftUI

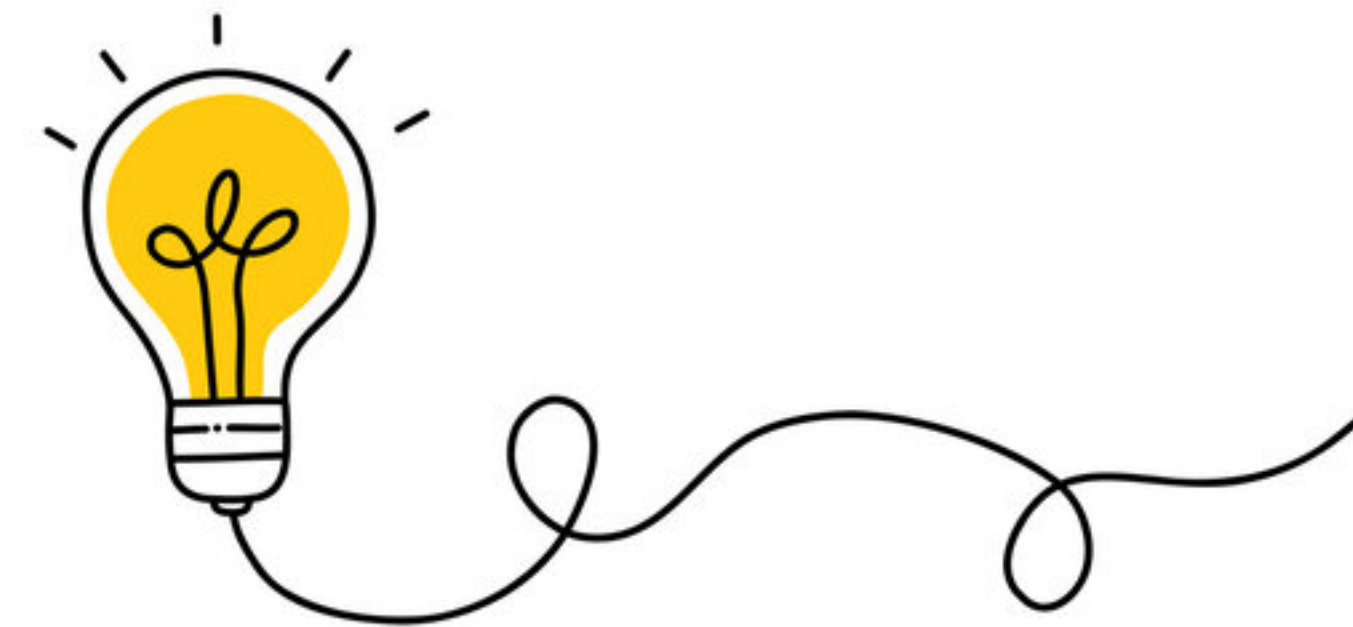
Doesn't build itself though.



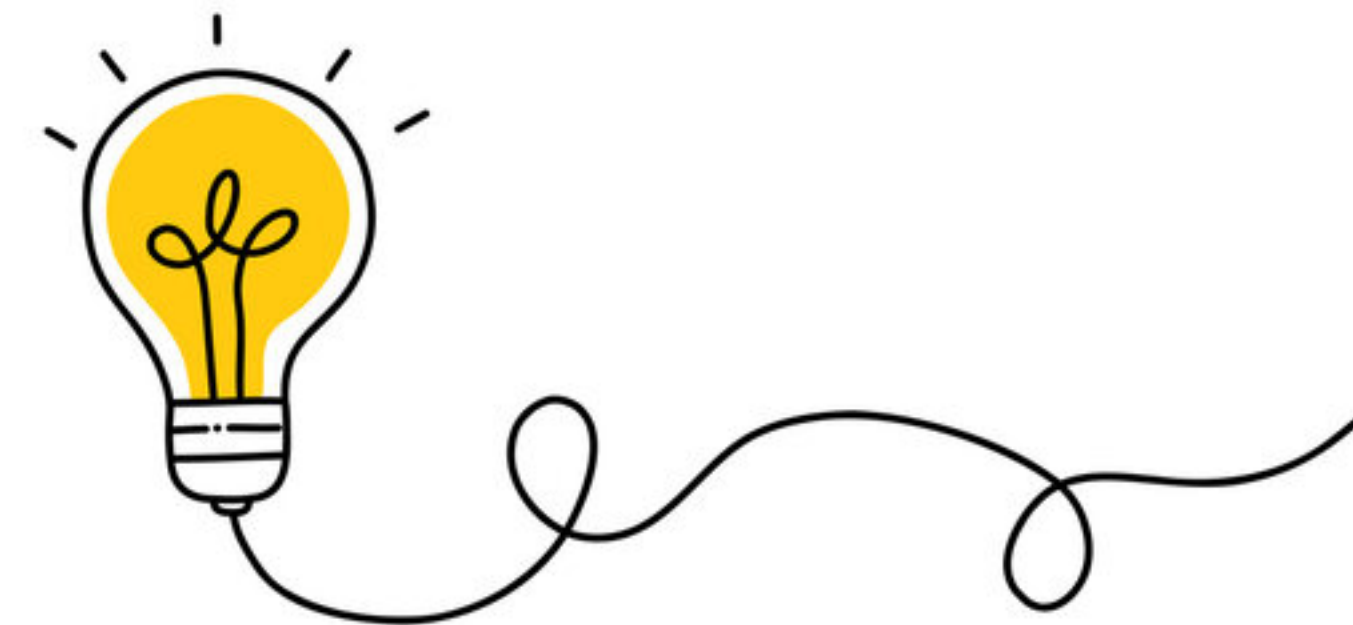
The Challenge

Build a simple SwiftUI app
for macOS
that you can **actually** use

An app to unsign a profile



An app to unsign a profile



Ever opened a signed profile?



←

→

Search

☐☐☐☐

1

Welcome

☰ Super Secret Signed Profile.mobileconfig ●

▶

☐

⋮

🔍

🔗

🔍

📁

⚠️

The file is not displayed in the text editor because it is either binary or uses an unsupported text encoding.

Open Anyway

👤

⚙️

<

⊗ 0 ⚠️ 0 🔊 0

Ln 1, Col 13

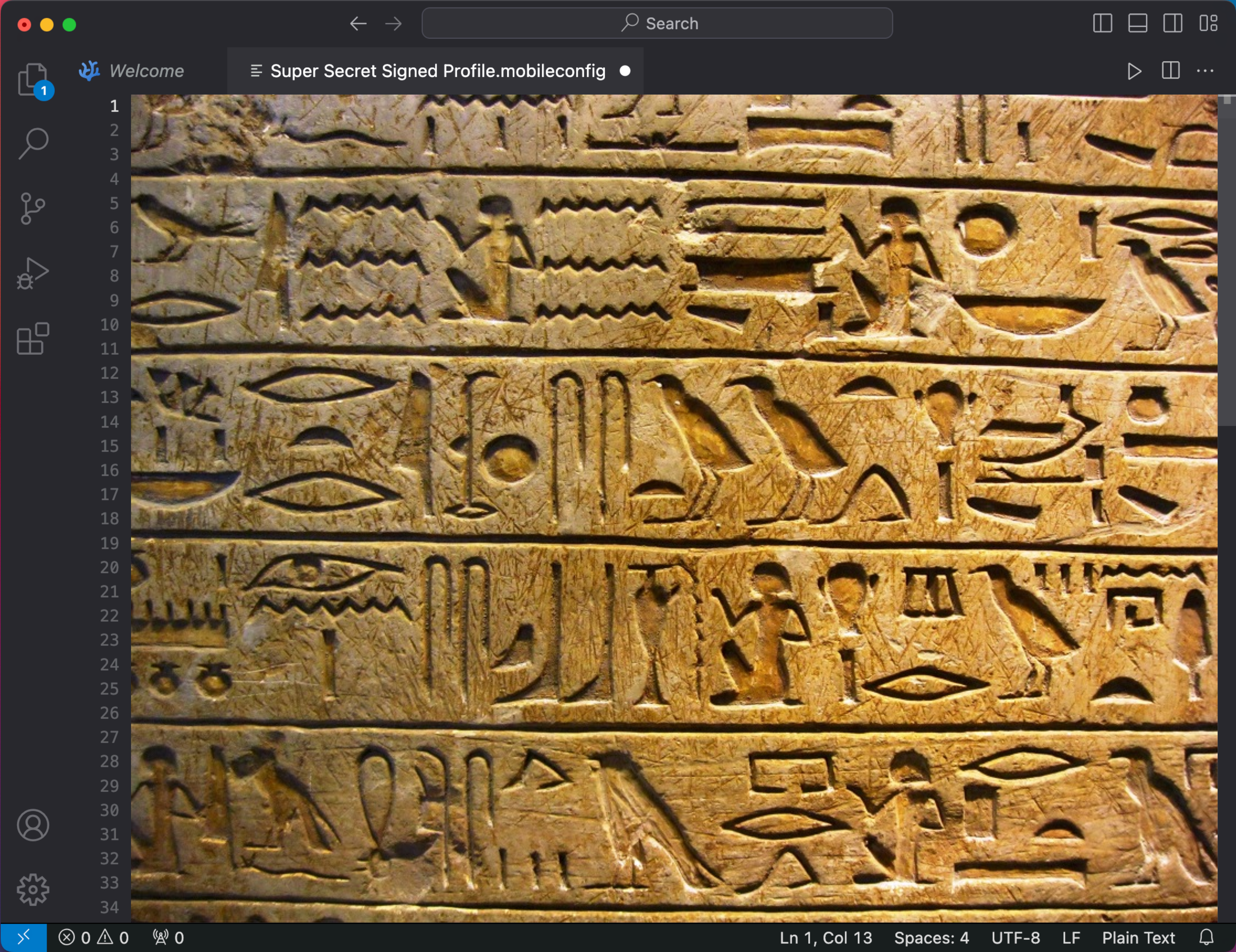
Spaces: 4

UTF-8

LF

Plain Text

🔔



macmule.com



MACMULE

DOING THE DONKEY WORK TO
MAKE YOU LOOK LIKE A SMART
ASS!!

Search ...

THE MULE'S MUSINGS
CATEGORISED

APPLICATION (10)

BECAUSE (8)

ERRORS (45)

EXTENSION ATTRIBUTE (6)

HOW TO (72)

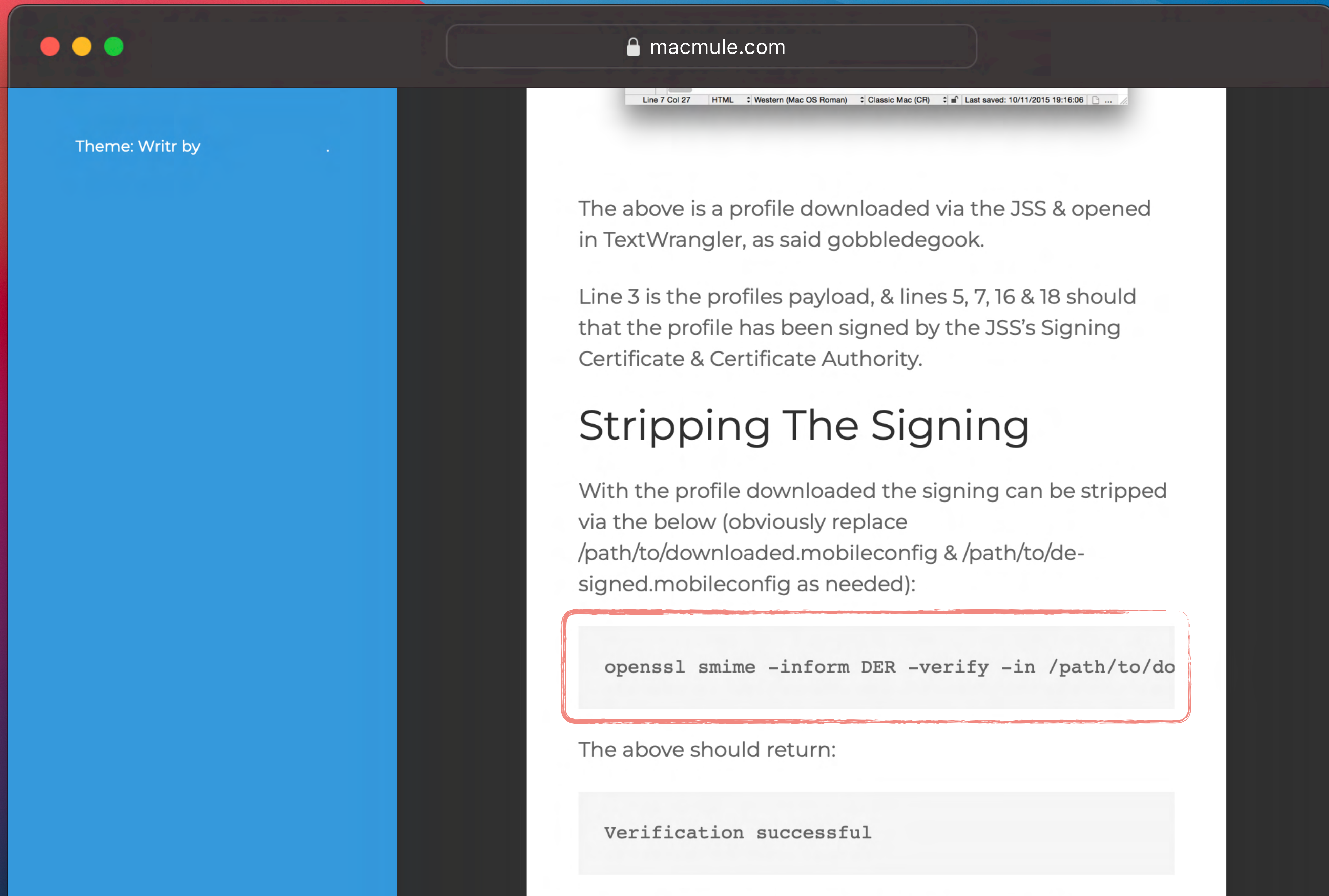
MAKING DOWNLOAD
CONFIGURATION PRO
READABLE



If you manually download a Configuration Profile from

Shout out to Ben for the
incredibly useful blog

A cartoon robot with a silver body and a black visor is holding a yellow sign with the word 'APPLAUSE' in red capital letters. The robot has a friendly expression with a grid-like mouth.



Theme: Writr by

The above is a profile downloaded via the JSS & opened in TextWrangler, as said gobbledegook.

Line 3 is the profiles payload, & lines 5, 7, 16 & 18 should that the profile has been signed by the JSS's Signing Certificate & Certificate Authority.

Stripping The Signing

With the profile downloaded the signing can be stripped via the below (obviously replace /path/to/downloaded.mobileconfig & /path/to/de-signed.mobileconfig as needed):

```
openssl smime -inform DER -verify -in /path/to/do
```

The above should return:

```
Verification successful
```


 macmule.com

Line 7 Col 27 HTML ⌵ Western (Mac OS Roman) ⌵ Classic Mac (CR) ⌵ Last saved: 10/11/2015 19:16:06

Theme: Writr by

The above is a profile downloaded via the JSS & opened in TextWrangler, as said gobbledegook.

Line 3 is the profiles payload, & lines 5, 7, 16 & 18 should

```
openssl smime -inform DER -verify -in /path/to/downloaded.mobileconfig -noverify -out /path/to/de-signed.mobileconfig
```

Stripping The Signing

```
plutil -convert xml1 /path/to/de-signed.mobileconfig
```

/path/to/downloaded.mobileconfig & /path/to/de-signed.mobileconfig as needed):

```
openssl smime -inform DER -verify -in /path/to/do
```

The above should return:

```
Verification successful
```


Let's try a shorter method



```
man security
```



```
cms [-C|-D|-E|-S] [options...] Encode or decode CMS messages.  
-C          create a CMS encrypted message  
-D          decode a CMS message  
-E          create a CMS enveloped message  
-S          create a CMS signed message
```

Decoding options:

```
-c content    use this detached content file  
-h level      generate email headers with info about CMS message (output level >= 0)  
-n           suppress output of content
```

Encoding options:

```
-r id,...      create envelope for comma-delimited list of recipients, where id can be a certificate  
               nickname or email address  
-G            include a signing time attribute  
-H hash       hash = MD2|MD4|MD5|SHA1|SHA256|SHA384|SHA512 (default: SHA1)  
-N nick       use certificate named "nick" for signing  
-P            include a SMIMECapabilities attribute  
-T            do not include content in CMS message  
-Y nick       include an EncryptionKeyPreference attribute with certificate (use "NONE" to omit)  
-Z hash       find a certificate by subject key ID
```

Common options:

```
-e envelope    specify envelope file (valid with -D or -E)  
-k keychain    specify keychain to use  
-i infile      use infile as source of data (default: stdin)  
-o outfile     use outfile as destination of data (default: stdout)
```


Our “unsign” command



```
security cms -D \  
  -i /path/to/source.mobileconfig \  
  -o /path/to/destination.mobileconfig
```


Software You'll Need



Xcode

Required



SF Symbols

Optional

Select macOS App template

Choose a template for your new project:

Multiplatform

iOS

macOS

watchOS

tvOS

visionOS

DriverKit

Other

Application



App



Document App



Game



Command Line Tool



Safari Extension App

Framework & Library



Framework



Library



Metal Library



XPC Service



Bundle

Other











Cancel

Previous

Next

Choose options for your new project:

Product Name: IntroToSwiftUIApp

Team: Ryan Ball (Personal Team)

Organization Identifier: com.ryan.ball

Bundle Identifier: com.ryan.ball.IntroToSwiftUIApp

Interface: SwiftUI

Language: Swift

Storage: None

☐ Host in CloudKit

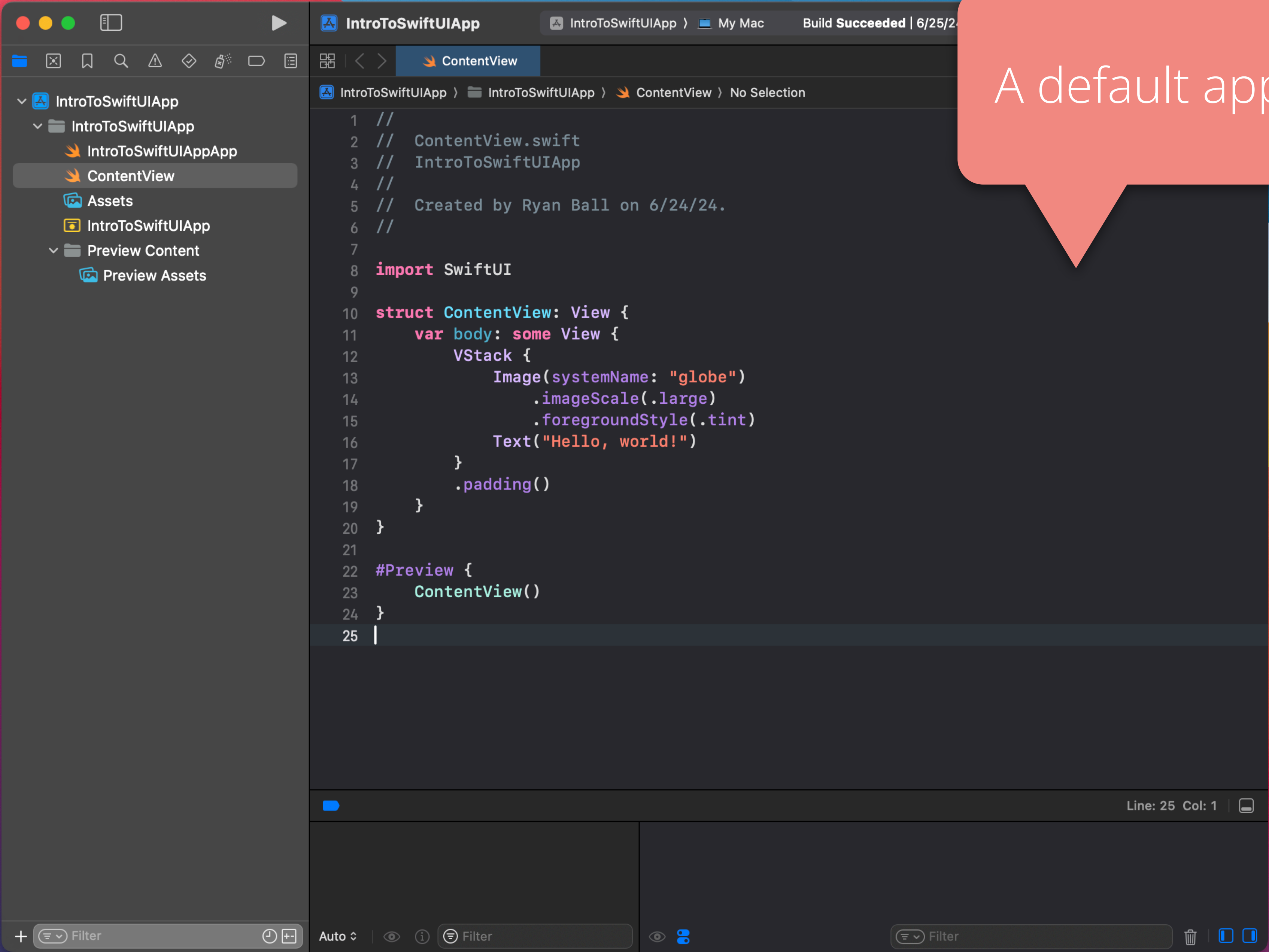
☐ Include Tests

Cancel

Previous

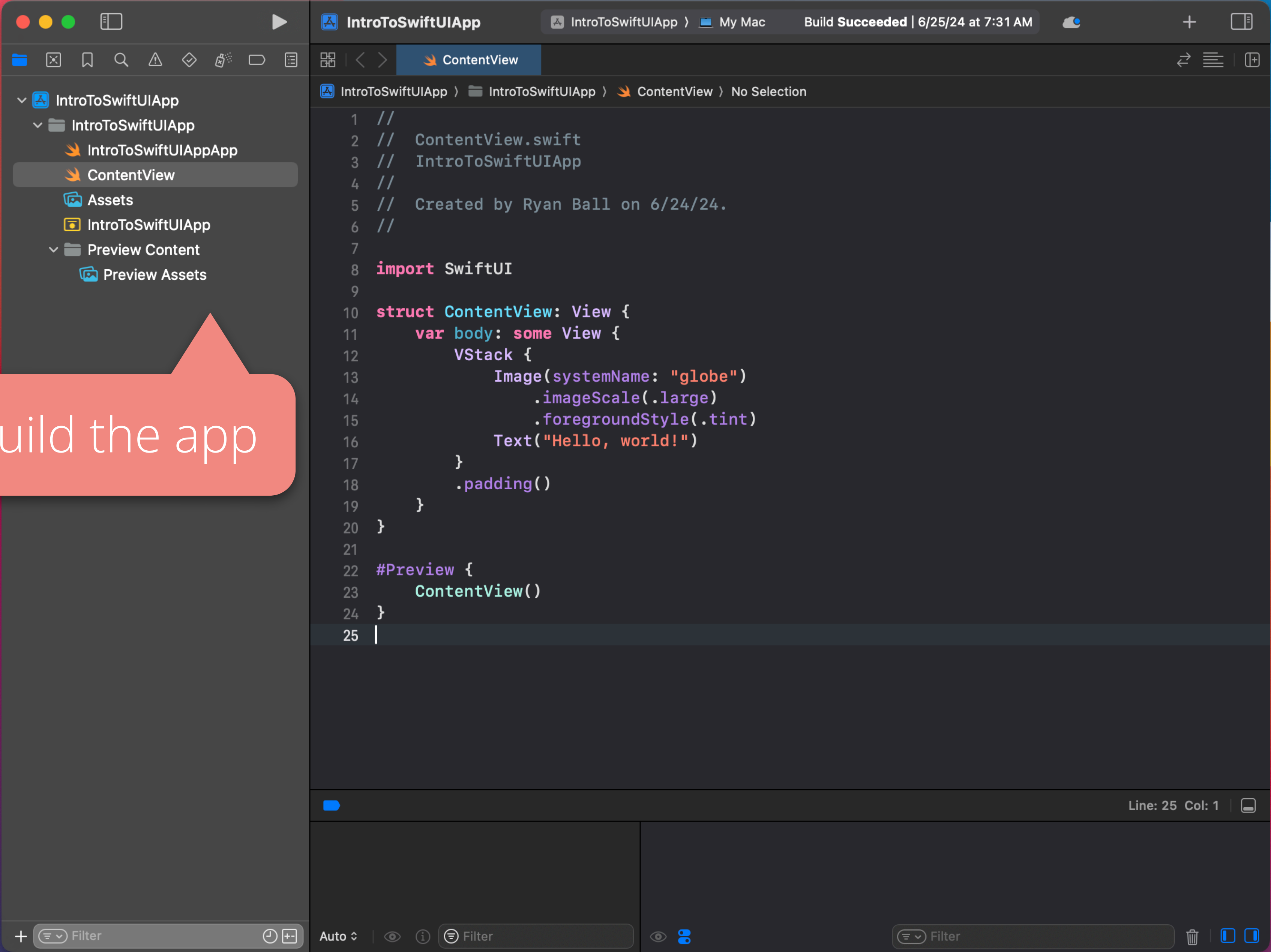
Next

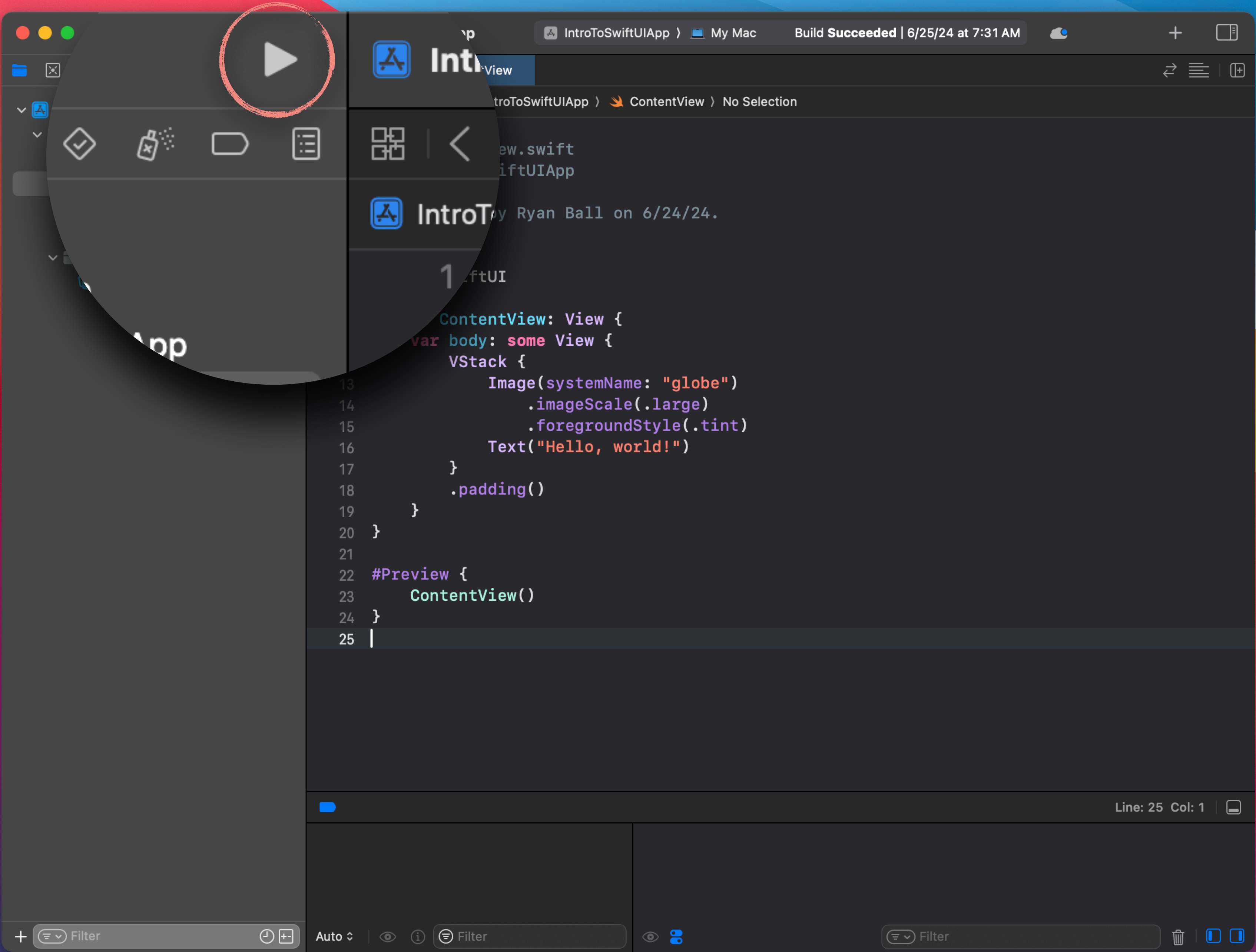
Name the project

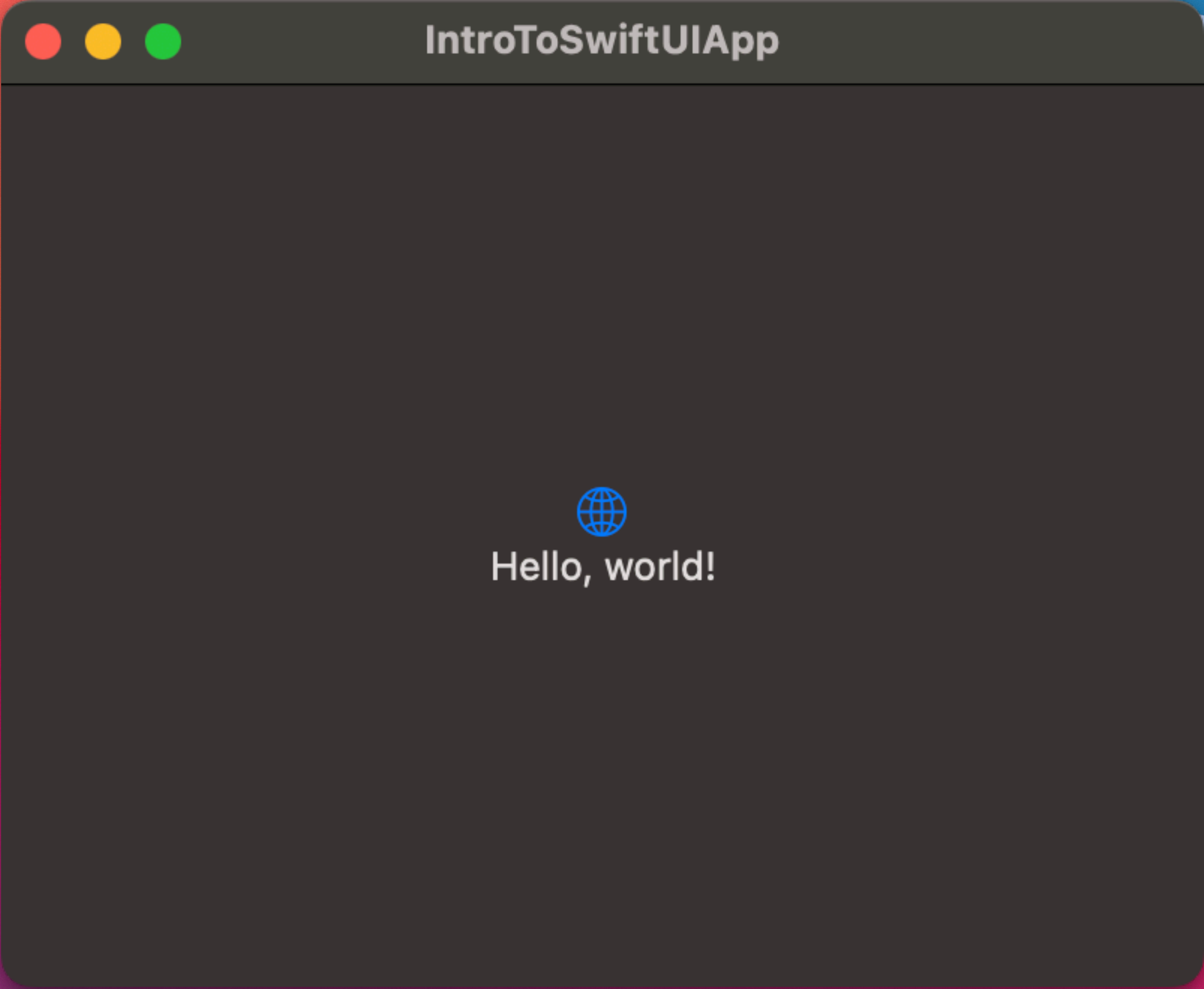


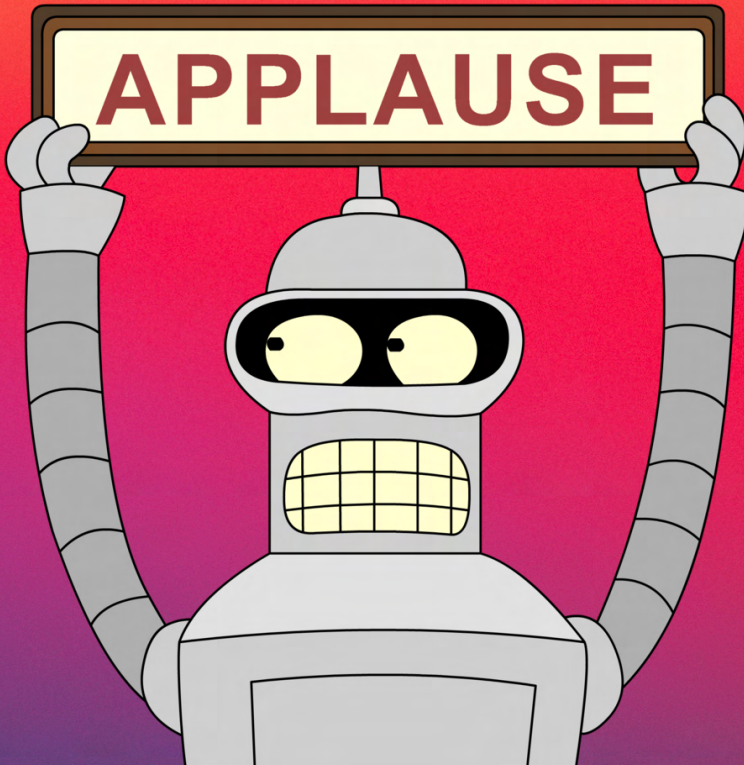
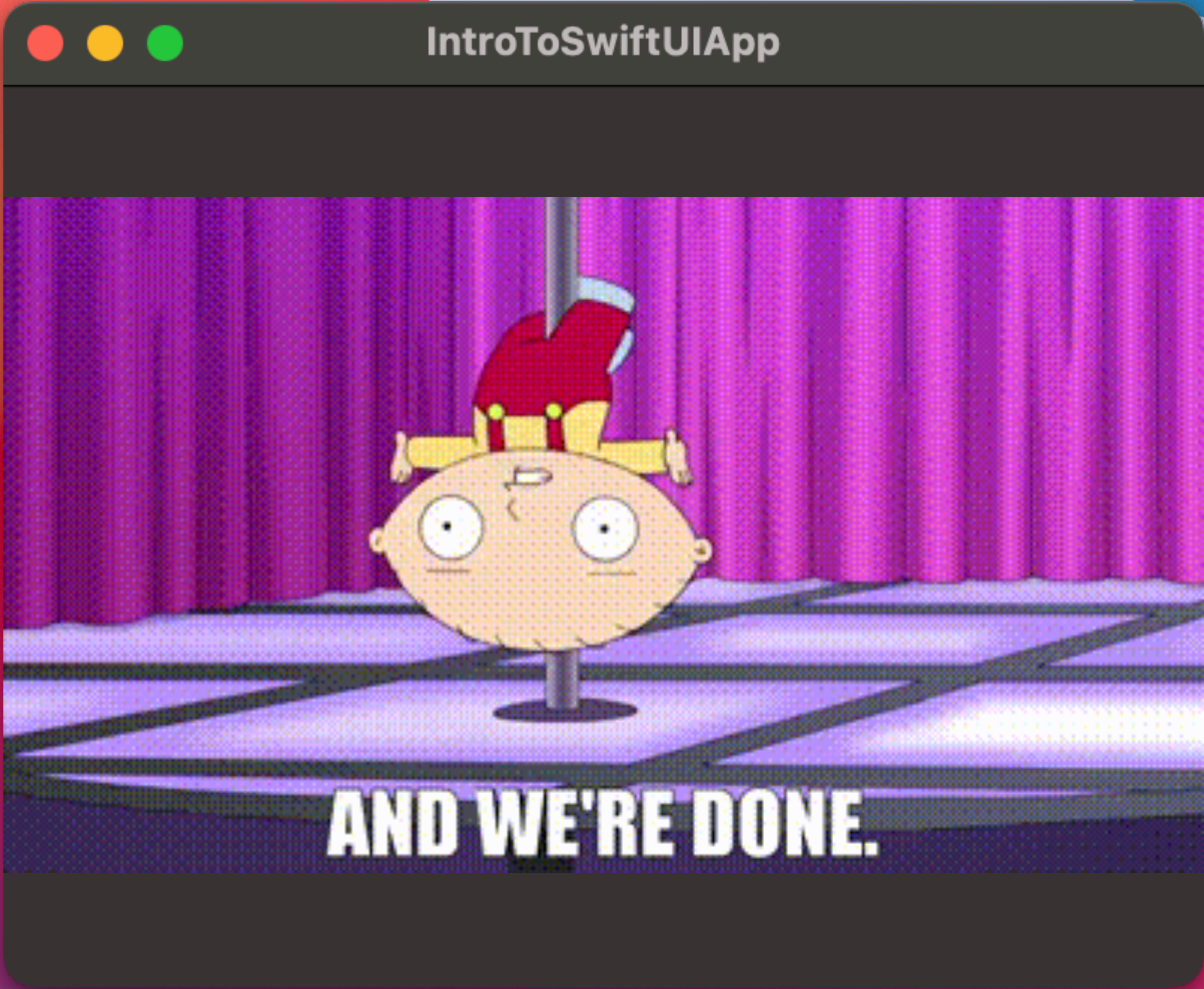
A default app is created

Build the app









Let's Dissect the Default App

The App Struct File

The App Struct File

- Includes the `@main` which marks the entry point of the app
- The body contains one or more scenes
- Data models can be initialized here and shared across all views
- Typically named after the app + “App”



The App Struct

```
1 import SwiftUI
2
3 @main
4 struct IntroToSwiftUIApp: App {
5     var body: some Scene {
6         WindowGroup {
7             ContentView()
8         }
9     }
10 }
```


SwiftUI Views

SwiftUI Views

- Represents part of the user interface
- Can be a simple text label or multiple complex interactive elements
- Views can contain other views
- Modifiers are used to change the appearance and behavior of views
- Stacks are used to arrange views



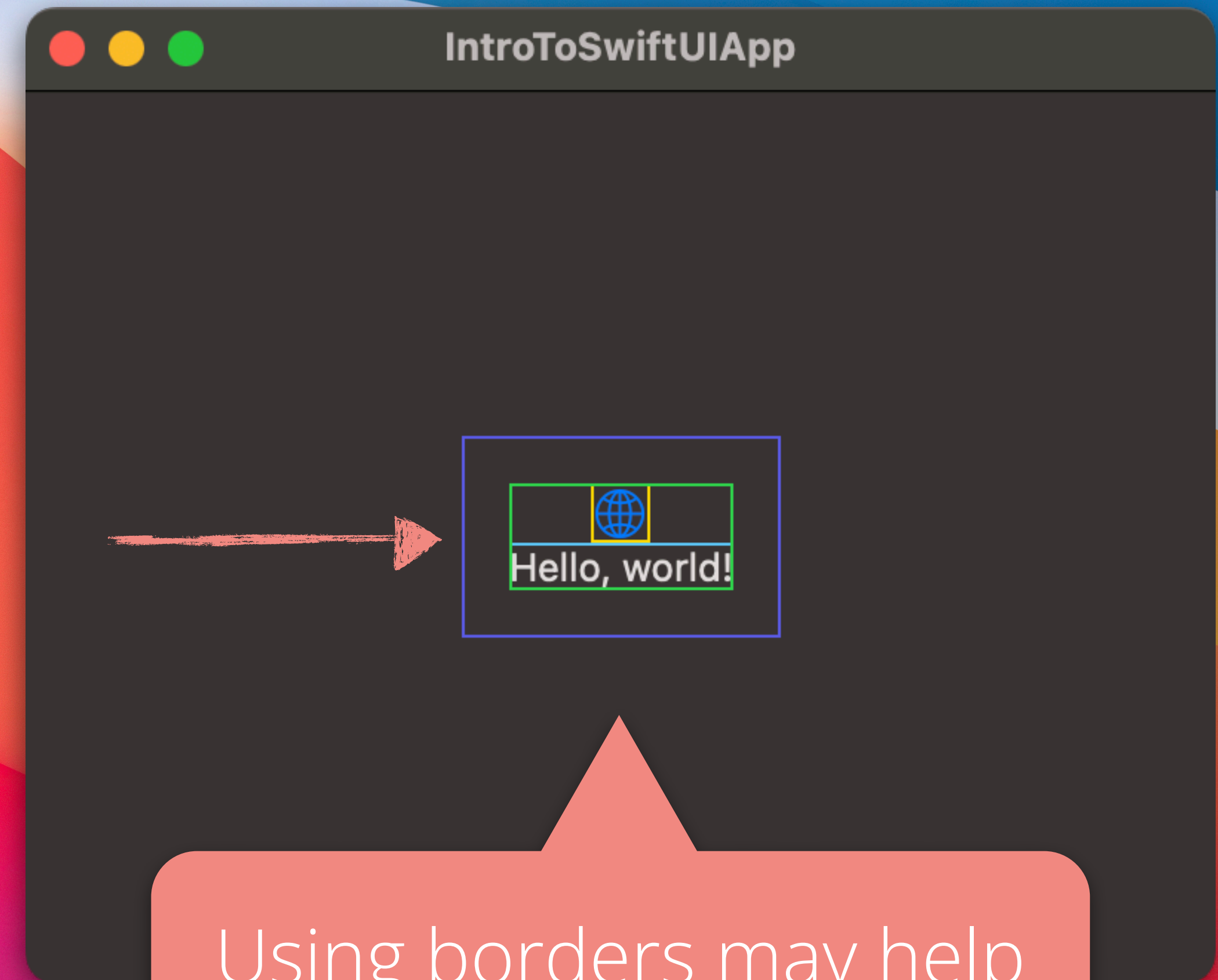

```
1 import SwiftUI
2
3 struct ContentView: View {
4     var body: some View {
5         VStack {
6             Image(systemName: "globe")
7                 .imageScale(.large)
8                 .foregroundStyle(.tint)
9             Text("Hello, world!")
10        }
11        .padding()
12    }
13 }
```

IntroToSwiftUIApp

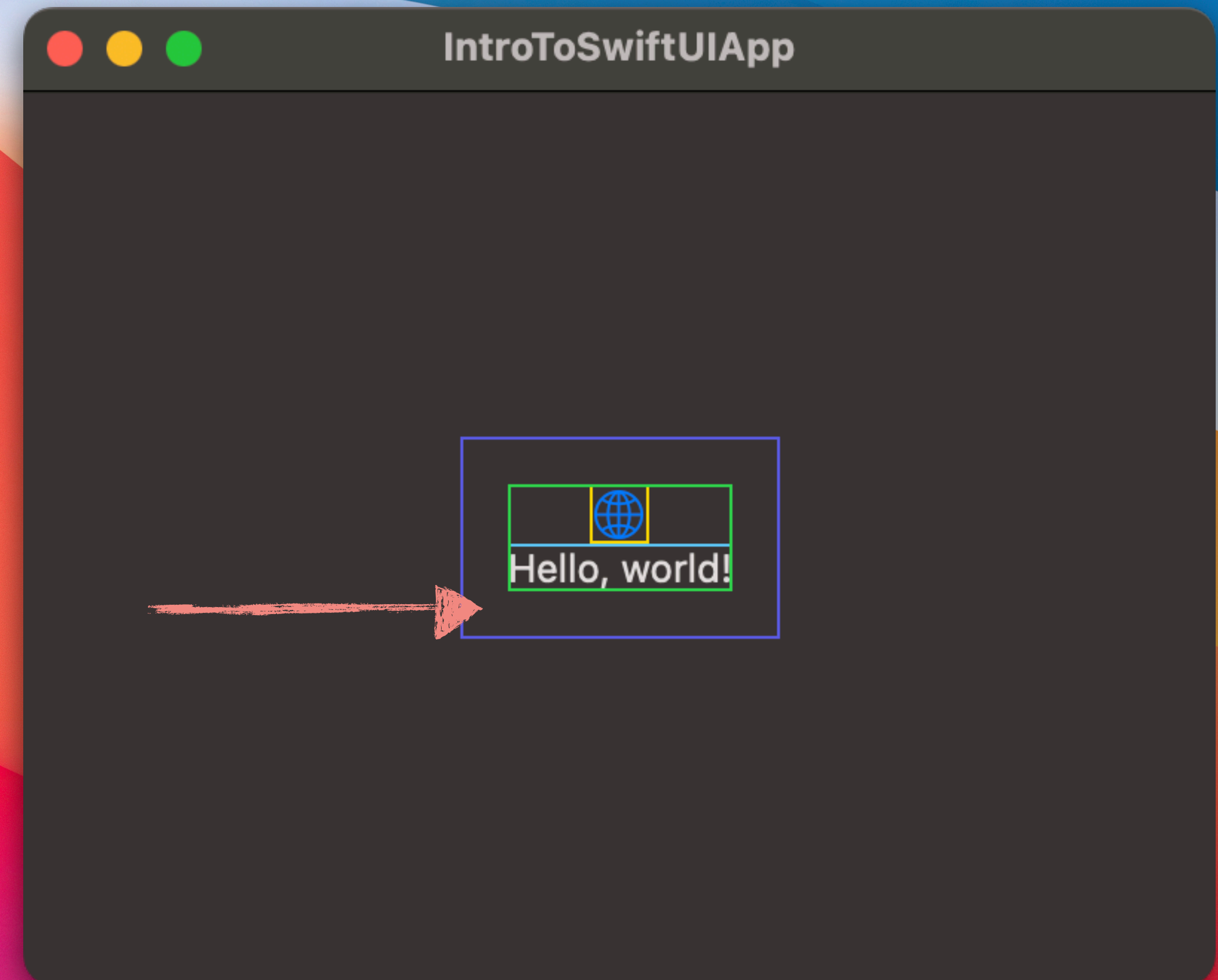


Hello, world!

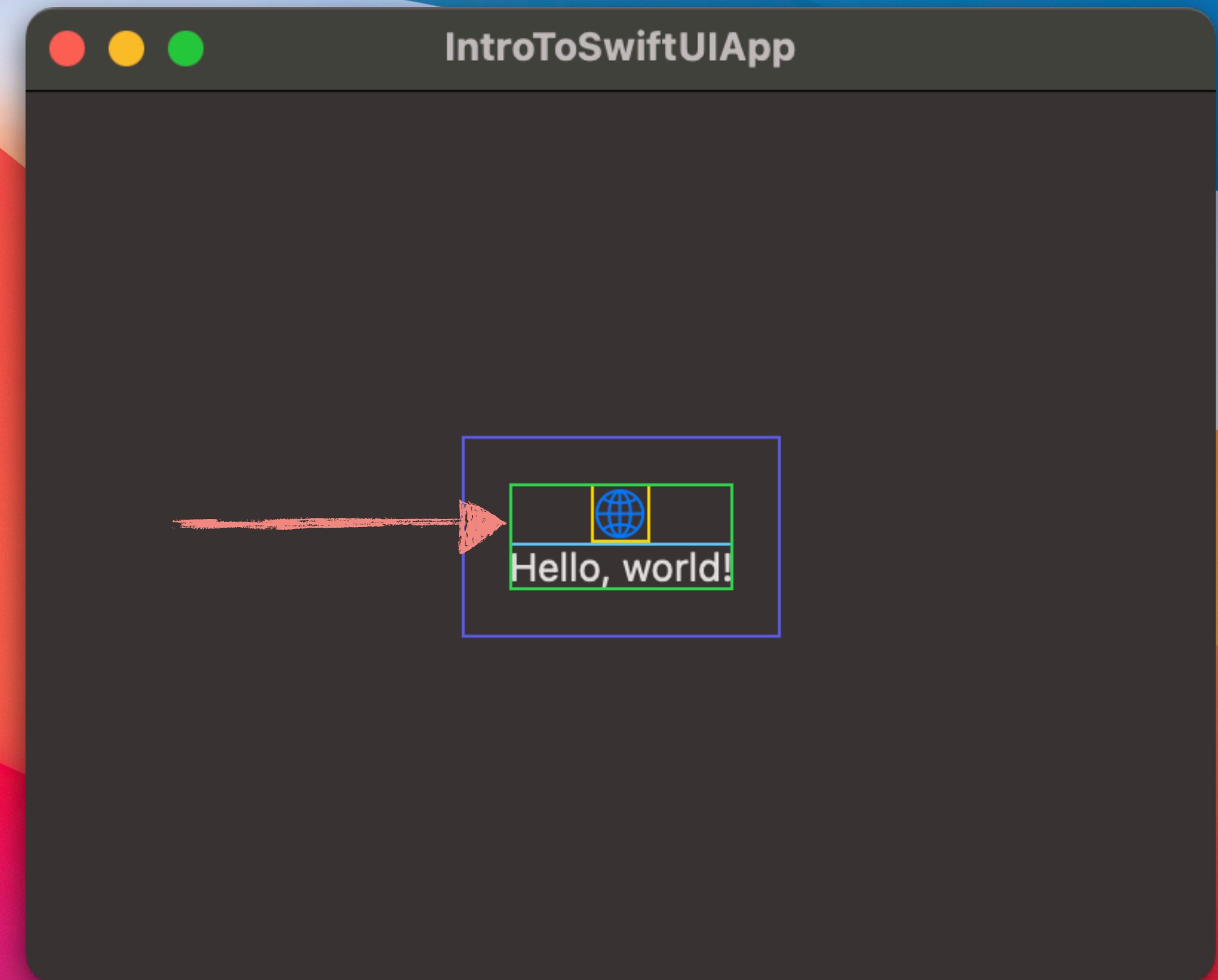

```
1 struct ContentView: View {  
2     var body: some View {  
3         VStack {  
4             Image(systemName: "globe")  
5                 .imageScale(.large)  
6                 .foregroundStyle(.tint)  
7                 .border(.yellow)  
8             Text("Hello, world!")  
9                 .border(.cyan)  
10        }  
11        .border(.green)  
12        .padding()  
13        .border(.indigo) ←  
14    }  
15 }
```



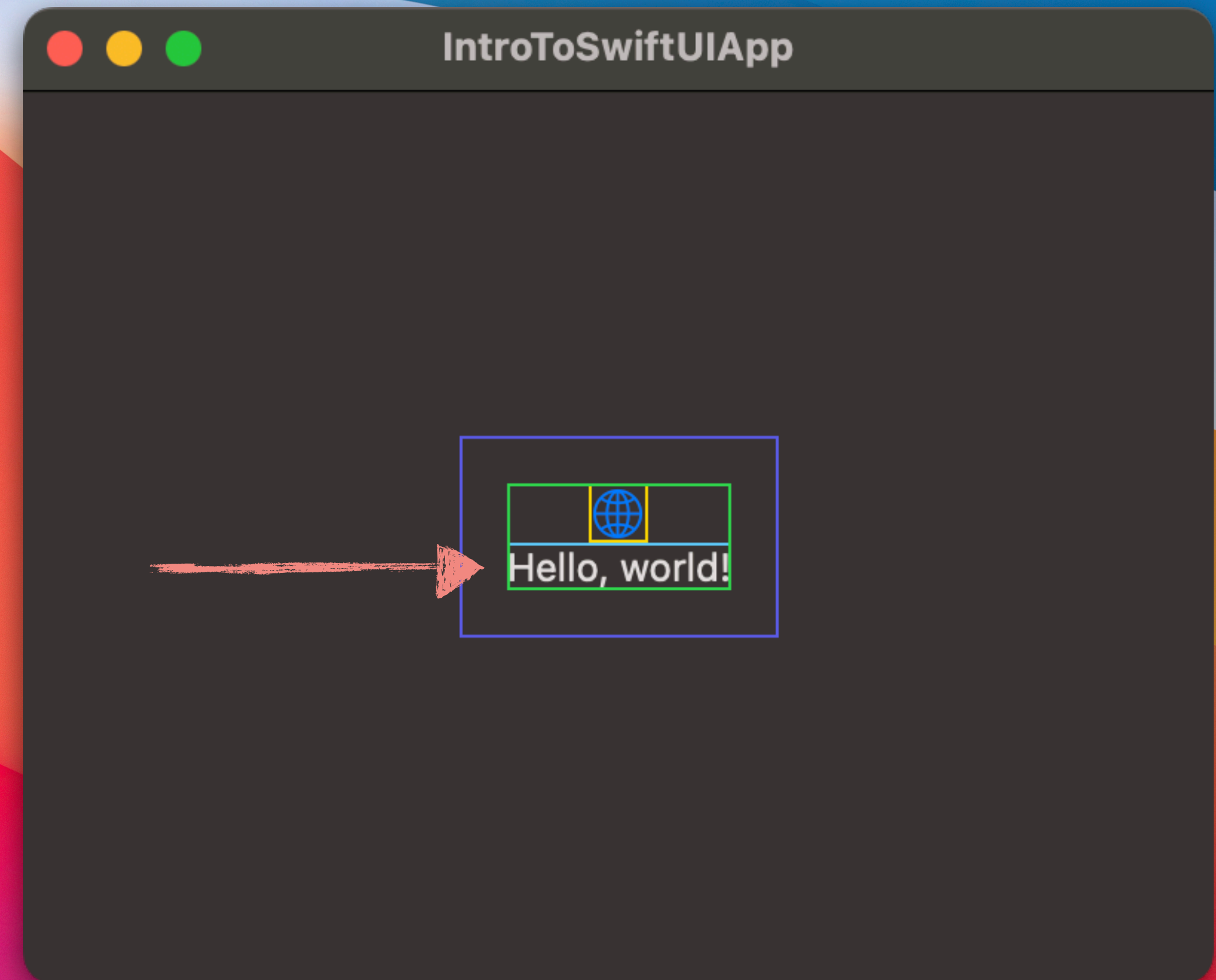

```
1 struct ContentView: View {  
2     var body: some View {  
3         VStack {  
4             Image(systemName: "globe")  
5                 .imageScale(.large)  
6                 .foregroundStyle(.tint)  
7                 .border(.yellow)  
8             Text("Hello, world!")  
9                 .border(.cyan)  
10        }  
11        .border(.green)  
12        .padding() ←  
13        .border(.indigo)  
14    }  
15 }
```




```
1 struct ContentView: View {  
2     var body: some View {  
3         VStack {  
4             Image(systemName: "globe")  
5                 .imageScale(.large)  
6                 .foregroundStyle(.tint)  
7                 .border(.yellow)  
8             Text("Hello, world!")  
9                 .border(.cyan)  
10        }  
11        .border(.green) ←  
12        .padding()  
13        .border(.indigo)  
14    }  
15 }
```



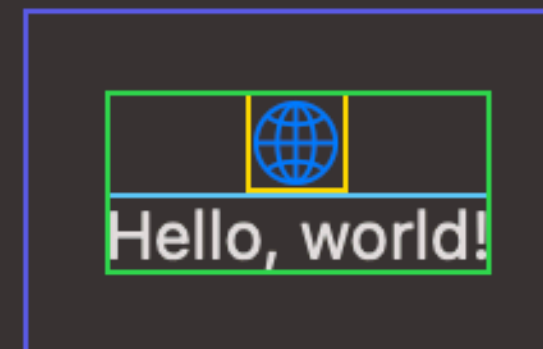

```
1 struct ContentView: View {  
2     var body: some View {  
3         VStack {  
4             Image(systemName: "globe")  
5                 .imageScale(.large)  
6                 .foregroundStyle(.tint)  
7                 .border(.yellow)  
8             Text("Hello, world!")  
9                 .border(.cyan) ←  
10        }  
11        .border(.green)  
12        .padding()  
13        .border(.indigo)  
14    }  
15 }
```



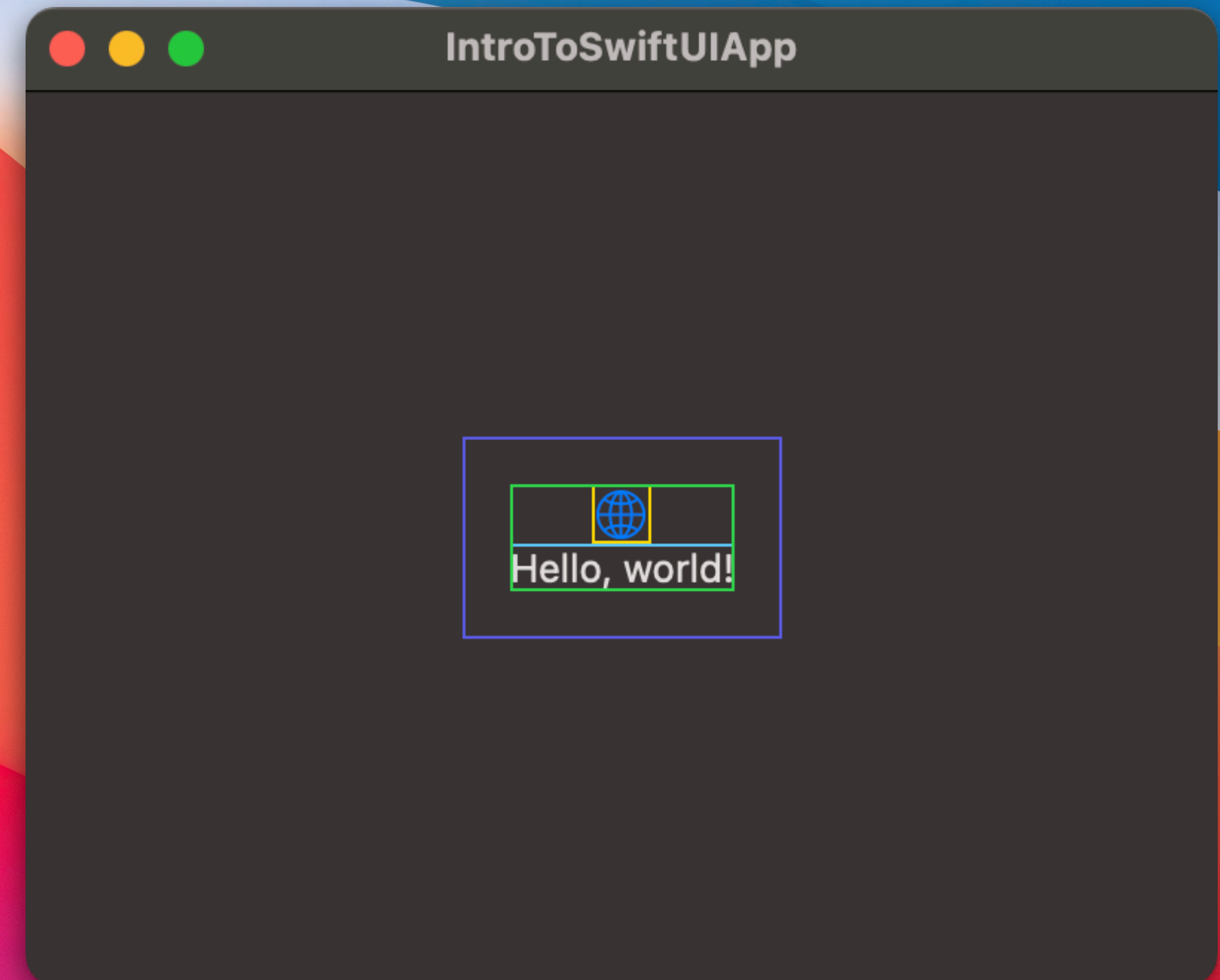

```
1 struct ContentView: View {  
2     var body: some View {  
3         VStack {  
4             Image(systemName: "globe")  
5                 .imageScale(.large)  
6                 .foregroundStyle(.tint)  
7                 .border(.yellow)  
8             Text("Hello, world!")  
9                 .border(.cyan)  
10        }  
11        .border(.green)  
12        .padding()  
13        .border(.indigo)  
14    }  
15 }
```

These are views

IntroToSwiftUIApp



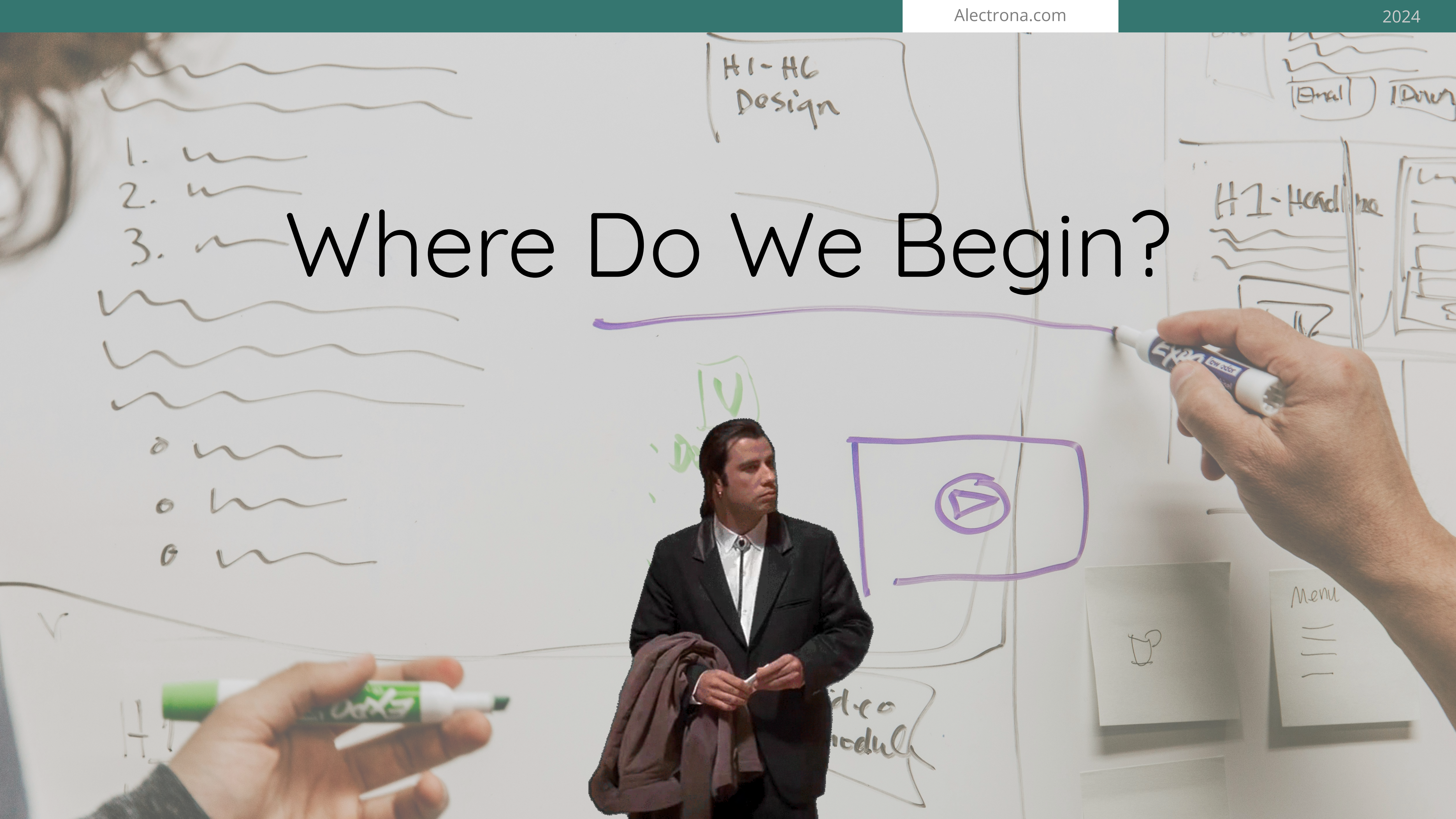

```
1 struct ContentView: View {  
2     var body: some View {  
3         VStack {  
4             Image(systemName: "globe")  
5                 .imageScale(.large) ←  
6                 .foregroundColor(.tint) ←  
7                 .border(.yellow) ←  
8             Text("Hello, world!")  
9                 .border(.cyan) ←  
10        }  
11        .border(.green) ←  
12        .padding() ←  
13        .border(.indigo) ←  
14    }  
15 }
```



These are view modifiers

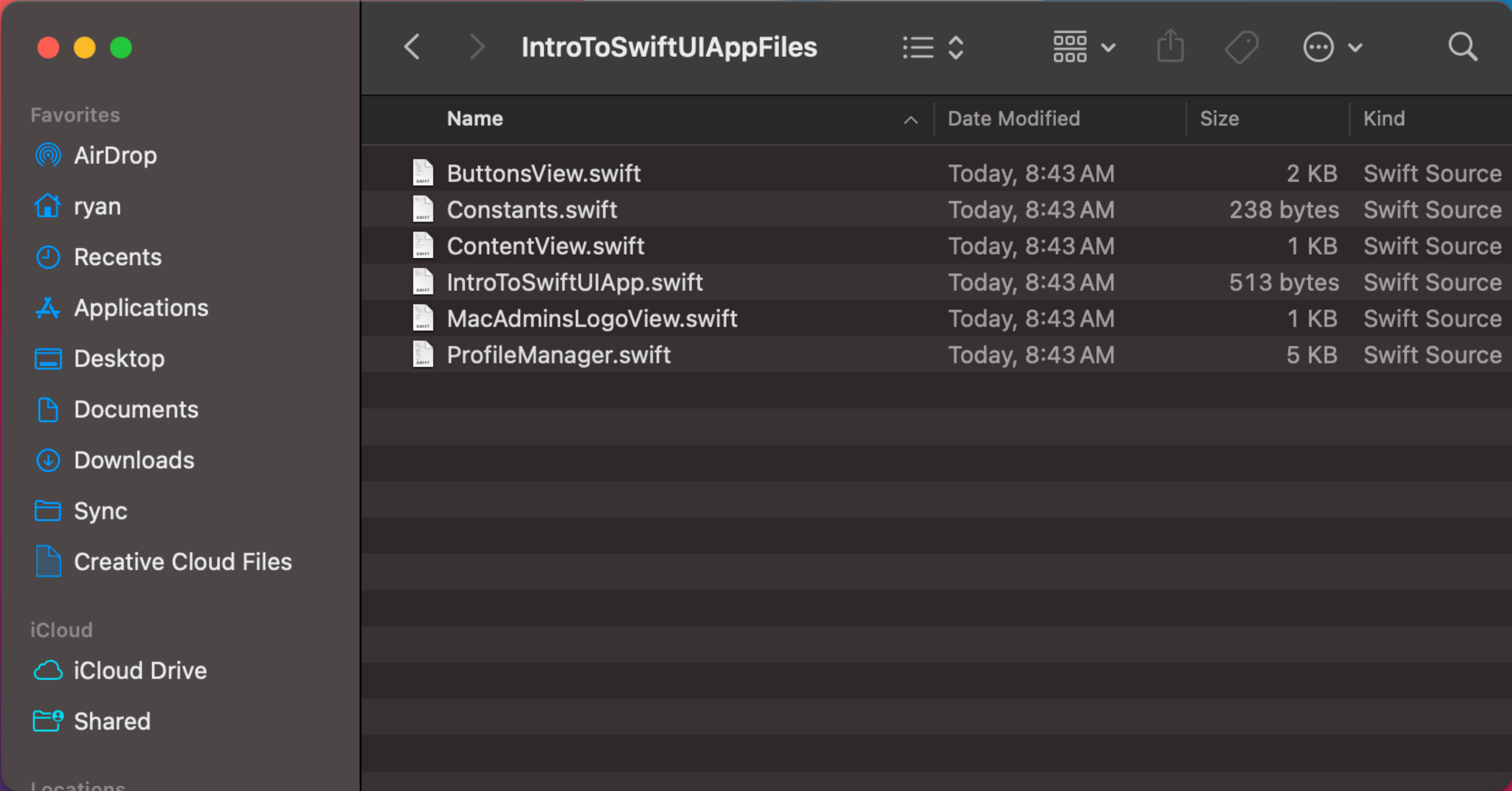
Let's Construct **Our** App

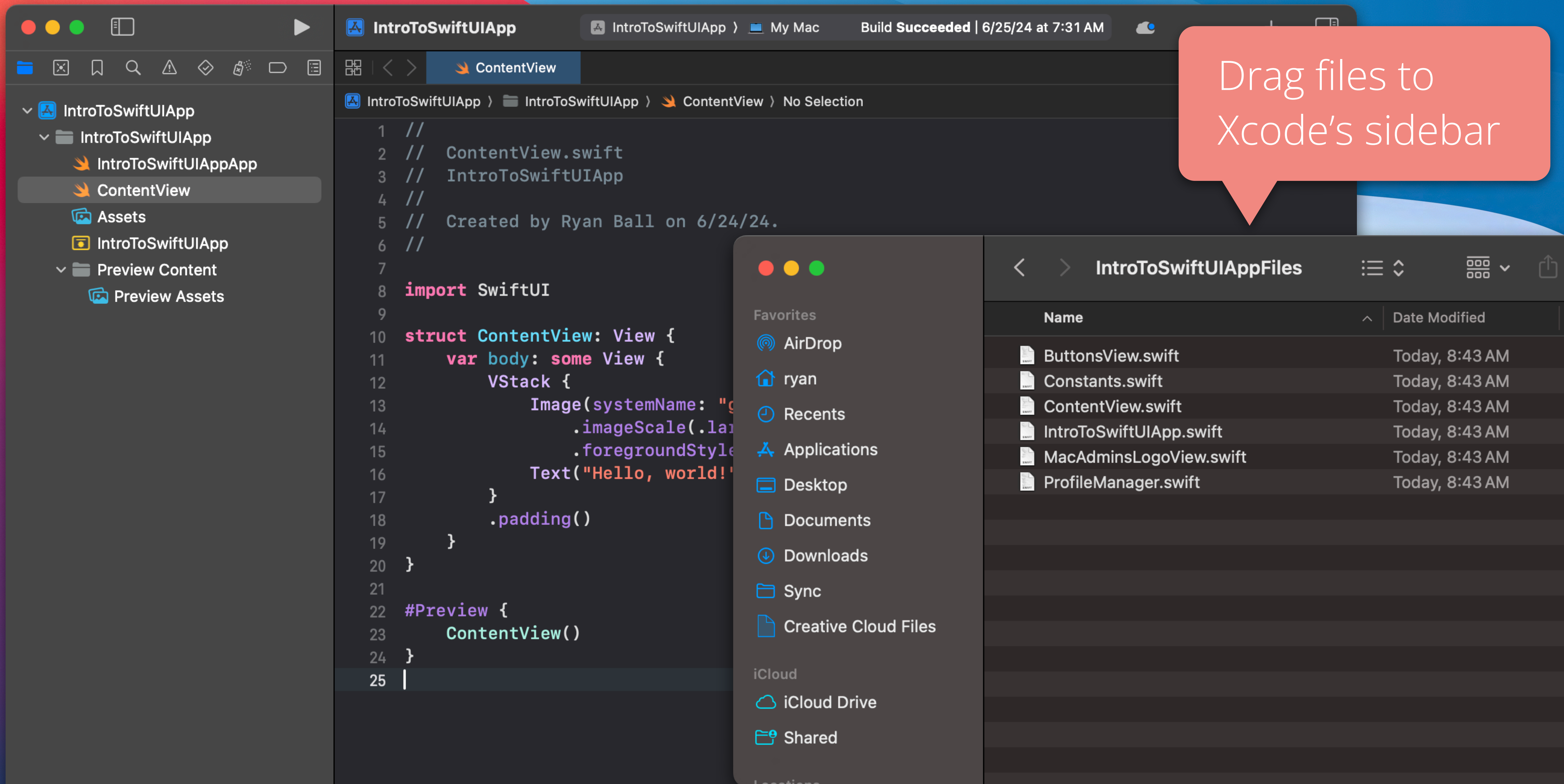
Where Do We Begin?





bit.ly/psu-intro-swiftui-files





IntroToSwiftUIApp

IntroToSwiftUIApp

IntroToSwiftUIAppApp

ContentView

Assets

IntroToSwiftUIApp

Preview Content

Preview Assets

ContentView

IntroToSwiftUIApp > IntroToSwiftUIApp > ContentView > No Selection

1 //

2 // ContentView.swift

3 // IntroToSwiftUIApp

4 //

5 // Created by Ryan Ball on 6/24/24.

6 //

7

8 import SwiftUI

9

10 struct ContentView: View {

11 var body: some View {

12 VStack {

13 Image(systemName: "globe")

14 .imageScale(.large)

15 .foregroundStyle(.teal)

16 Text("Hello, world!")

17 }

18 .padding()

19 }

20 }

21

22 #Preview {

23 ContentView()

24 }

25 |

AirDrop

ryan

Recents

Applications

Desktop

Documents

Downloads

Sync

Creative Cloud Files

iCloud

iCloud Drive

Shared

IntroToSwiftUIAppFiles

Name	Date Modified
ButtonsView.swift	Today, 8:43 AM
Constants.swift	Today, 8:43 AM
ContentView.swift	Today, 8:43 AM
IntroToSwiftUIApp.swift	Today, 8:43 AM
MacAdminsLogoView.swift	Today, 8:43 AM
ProfileManager.swift	Today, 8:43 AM

AirDrop

ryan

Recents

Applications

Desktop

Documents

Downloads

Sync

Creative Cloud Files

iCloud

iCloud Drive

Shared

IntroToSwiftUIApp

IntroToSwiftUIApp

IntroToSwiftUIAppApp

Assets

IntroToSwiftUIApp

Preview Content

Preview Assets

ContentView

IntroToSwiftUIApp > IntroToSwiftUIApp > ContentView > No Selection

```
1 //
2 // ContentView.swift
3 // IntroToSwiftUIApp
4 //
5 // Created by Ryan Ball on 6/24/24.
6 //
7
8 import SwiftUI
9
10 struct ContentView: View {
11     var body: some View {
12         VStack {
13             Image(systemName: "globe")
14               .imageScale(.large)
15               .foregroundStyle(.teal)
16             Text("Hello, world!")
17         }
18         .padding()
19     }
20 }
21
22 #Preview {
23     ContentView()
24 }
25 |
```

IntroToSwiftUIAppFiles

Name	Date Modified
ButtonsView.swift	Today, 8:43 AM
Constants.swift	Today, 8:43 AM
ContentView.swift	Today, 8:43 AM
IntroToSwiftUIApp.swift	Today, 8:43 AM
MacAdminsLogoView.swift	Today, 8:43 AM
ProfileManager.swift	Today, 8:43 AM

AirDrop

ryan

Recents

Applications

Desktop

Documents

Downloads

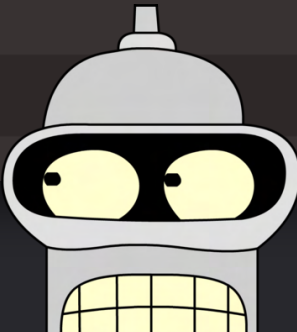
Sync

Creative Cloud Files

iCloud

iCloud Drive

Shared



Choose options for adding these files:

Destination: ☒ Copy items if needed

Added folders: ☒ Create groups
☐ Create folder references

Add to targets: ☒  IntroToSwiftUIApp



Cancel

Finish

Constants.swift

```
1 //  
2 // Constants.swift  
3 // IntroToSwiftUIApp  
4 //  
5 // Created by Ryan Ball on 6/21/24.  
6 //  
7  
8 import Foundation  
9  
10 let psuLogo = URL(string: "https://macadmins.psu.edu/files/2024/01/2024-MacAdmins-Logo-Circle-Blue-  
7295dd5c06f66b58-300x283.png")  
11
```

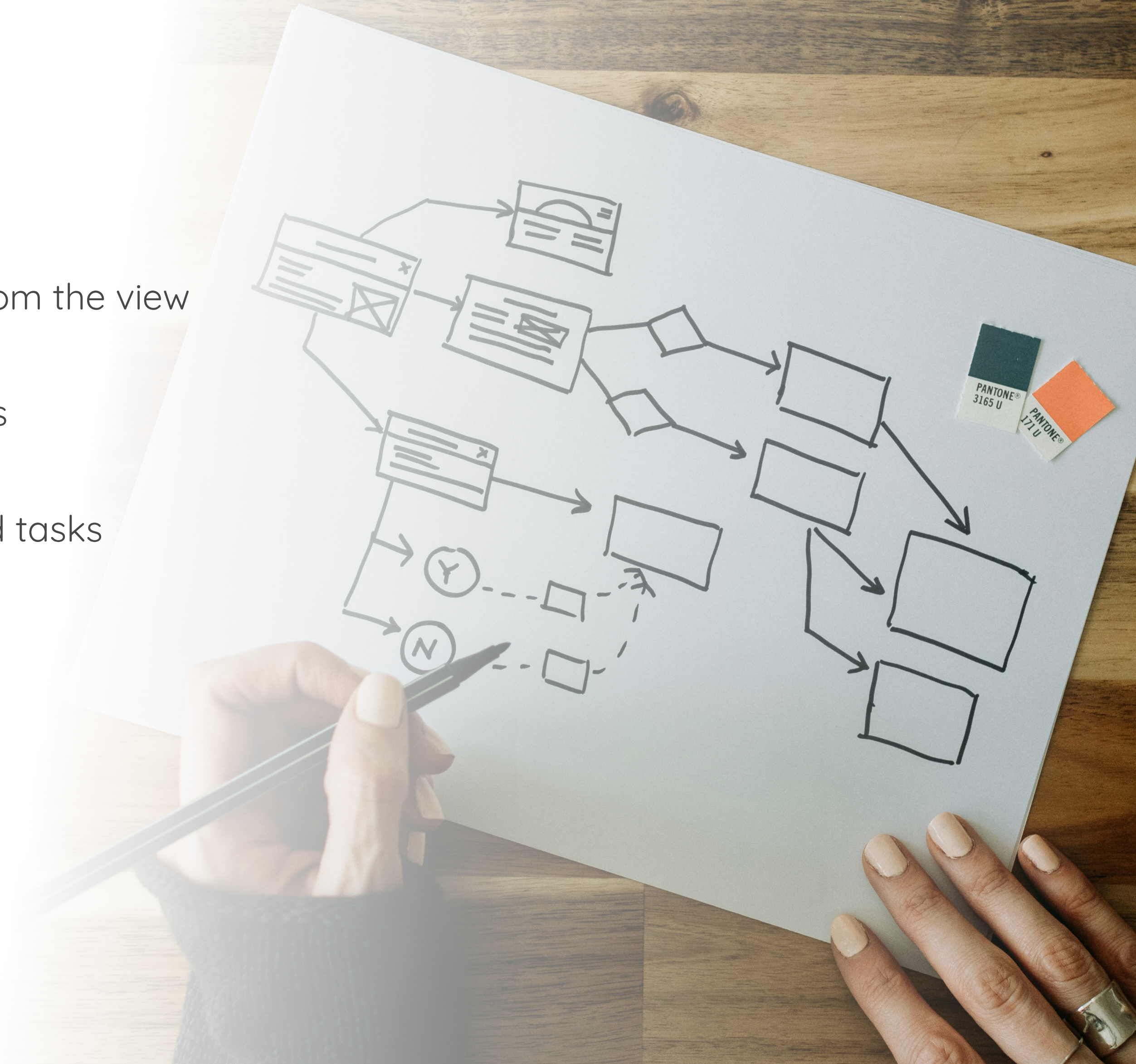

Global variables are available to the entire project

```
1 //  
2 // Constants.swift  
3 // IntroToSwiftUIApp  
4 //  
5 // Created by Ryan Ball on 6/21/24.  
6 //  
7  
8 import Foundation  
9  
10 let psuLogo = URL(string: "https://macadmins.psu.edu/files/2024/01/2024-MacAdmins-Logo-Circle-Blue-7295dd5c06f66b58-300x283.png")  
11
```


SwiftUI Data Model

SwiftUI Data Model

- Data models (view models) separate the business logic from the view
- Notifies the view of changes using **@Published** properties
- Manages and propagates errors that occur in background tasks
- BY FAR the most complex component of this app



ProfileManager.swift

```
8 import SwiftUI
9 import UniformTypeIdentifiers
10
11 class ProfileManager: NSObject, ObservableObject {
12     @Published var selectedProfile: URL? {
13
14         // Perform an action when this property is set
15         didSet {
16             if let profile = selectedProfile {
17
18                 // Set the profileName property to the name of the profile
19                 self.profileName = profile.deletingPathExtension().lastPathComponent
20             }
21         }
22     }
23
24     @Published var profileName: String?
25     @Published var profileIcon: NSImage
26     @Published var shouldPresentAlert: Bool
27     @Published var alertMessage: String
28
29     // Initialize the class and set up properties
30     override init() {
31
```




```
8 import SwiftUI
9 import UniformTypeIdentifiers
10
11 class ProfileManager: NSObject, ObservableObject {
12     @Published var selectedProfile: URL? {
13
14         // Perform an action when this property is set
15         didSet {
16             if let profile = selectedProfile {
17
18                 // Set the profileName property to the name of the profile
19                 self.profileName = profile.deletingPathExtension().lastPathComponent
20             }
21         }
22     }
23
24     @Published var profileName: String?
25     @Published var profileIcon: NSImage
26     @Published var shouldPresentAlert: Bool
27     @Published var alertMessage: String
28
29     // Initialize the class and set up properties
30     override init() {
31
```



```
8 import SwiftUI
9 import UniformTypeIdentifiers
10
11 class ProfileManager: NSObject, ObservableObject {
12     @Published var selectedProfile: URL? {
13
14         // Perform an action when this property is set
15         didSet {
16             if let profile = selectedProfile {
17
18                 // Set the profileName property to the name of the profile
19                 self.profileName = profile.deletingPathExtension().lastPathComponent
20             }
21         }
22     }
23
24     @Published var profileName: String?
25     @Published var profileIcon: NSImage
26     @Published var shouldPresentAlert: Bool
27     @Published var alertMessage: String
28
29     // Initialize the class and set up properties
30     override init() {
31
```



```
8 import SwiftUI
9 import UniformTypeIdentifiers
10
11 class ProfileManager: NSObject, ObservableObject {
12     @Published var selectedProfile: URL? { ←
13
14         // Perform an action when this property is set
15         didSet {
16             if let profile = selectedProfile {
17
18                 // Set the profileName property to the name of the profile
19                 self.profileName = profile.deletingPathExtension().lastPathComponent
20             }
21         }
22     }
23
24     @Published var profileName: String?
25     @Published var profileIcon: NSImage
26     @Published var shouldPresentAlert: Bool
27     @Published var alertMessage: String
28
29     // Initialize the class and set up properties
30     override init() {
31
```



```
8 import SwiftUI
9 import UniformTypeIdentifiers
10
11 class ProfileManager: NSObject, ObservableObject {
12     @Published var selectedProfile: URL? {
13
14         // Perform an action when this property is set
15         didSet {
16             if let profile = selectedProfile {
17
18                 // Set the profileName property to the name of the profile
19                 self.profileName = profile.deletingPathExtension().lastPathComponent
20             }
21         }
22     }
23
24     @Published var profileName: String?
25     @Published var profileIcon: NSImage
26     @Published var shouldPresentAlert: Bool
27     @Published var alertMessage: String
28
29     // Initialize the class and set up properties
30     override init() {
31
```



```
18 // Set the profileName property to the name of the profile
19 self.profileName = profile.deletingPathExtension().lastPathComponent
20 }
21 }
22 }
23
24 @Published var profileName: String?
25 @Published var profileIcon: UIImage
26 @Published var shouldPresentAlert: Bool
27 @Published var alertMessage: String
28
29 // Initialize the class and set up properties
30 override init() {
31
32     self.selectedProfile = nil
33     self.profileName = nil
34     self.profileIcon = NSWorkspace.shared.icon(for: .mobileconfig)
35     self.shouldPresentAlert = false
36     self.alertMessage = ""
37 }
38
39 // Function to open a file open panel
40 func openFilePanel() {
41     let panel = NSOpenPanel()
42     panel.canChooseFiles = true
43     panel.canChooseDirectories = false
44     panel.allowsMultipleSelection = false
45     panel.allowedContentTypes = [.mobileconfig]
```



```
34     self.profileIcon = NSWorkspace.shared.icon(for: .mobileconfig)
35     self.shouldPresentAlert = false
36     self.alertMessage = ""
37 }
38
39 // Function to open a file open panel
40 func openFilePanel() {
41     let panel = NSOpenPanel()
42     panel.canChooseFiles = true
43     panel.canChooseDirectories = false
44     panel.allowsMultipleSelection = false
45     panel.allowedContentTypes = [.mobileconfig]
46
47     panel.begin { result in
48         if result == .OK, let url = panel.url {
49
50             // Set the published property on the main thread
51             DispatchQueue.main.async {
52                 self.selectedProfile = url
53             }
54         }
55     }
56 }
57
58 // Function to open a file save panel
59 // This will allow saving an unsigned version of the profile to the selected location.
60 func saveFilePanel() {
61     let panel = NSSavePanel()
```



```
56 }
57
58 // Function to open a file save panel
59 // This will allow saving an unsigned version of the profile to the selected location.
60 func saveFilePanel() {
61     let panel = NSSavePanel()
62     let fileName = selectedProfile?.lastPathComponent ?? "profile.mobileconfig"
63     panel.nameFieldStringValue = fileName
64     panel.allowedContentTypes = [.mobileconfig]
65
66     panel.begin { result in
67
68         // Ensure our optional variable is set
69         guard let selectedProfile = self.selectedProfile else {
70             fatalError("Should never happen")
71         }
72
73         // If the user chooses a destination location, unsign the profile and save it
74         if result == .OK, let destinationProfile = panel.url {
75             DispatchQueue.main.async {
76                 self.unsignProfile(from: selectedProfile, to: destinationProfile)
77             }
78         }
79     }
80 }
```

```
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
84
```



```
56 }
57
58 // Function to open a file save panel
59 // This will allow saving an unsigned version of the profile to the selected location.
60 func saveFilePanel() {
61     let panel = NSSavePanel()
62     let fileName = selectedProfile?.lastPathComponent ?? "profile.mobileconfig"
63     panel.nameFieldStringValue = fileName
64     panel.allowedContentTypes = [.mobileconfig]
65
66     panel.begin { result in
67
68         // Ensure our optional variable is set
69         guard let selectedProfile = self.selectedProfile else {
70             fatalError("Should never happen")
71         }
72
73         // If the user chooses a destination location, unsign the profile and save it
74         if result == .OK, let destinationProfile = panel.url {
75             DispatchQueue.main.async {
76                 self.unsignProfile(from: selectedProfile, to: destinationProfile)
77             }
78         }
79     }
80 }
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
```



```
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
84
85     // Perform the process on a background thread
86     DispatchQueue.global(qos: .userInitiated).async {
87         let process = Process()
88         process.executableURL = URL(filePath: "/usr/bin/security")
89         process.arguments = ["cms", "-D", "-i", source.path, "-o", destination.path]
90
91         // Capture the output in a pipe
92         let pipe = Pipe()
93         process.standardOutput = pipe
94         process.standardError = pipe
95
96         // Execute the command and wait for it to finish
97         process.launch()
98         process.waitUntilExit()
99
100        // Read the output while the command runs
101        let data = pipe.fileHandleForReading.readDataToEndOfFile()
102        let output = String(data: data, encoding: .utf8)
103
104        // Assume success if the command exits normally
105        if process.terminationStatus == 0 {
106
107            // Handle success on the main thread
108            DispatchQueue.main.async {
```



```
81  
82 // Function to unsign a mobileconfig file using the security command  
83 private func unsignProfile(from source: URL, to destination: URL) {
```

```
84  
85 // Perform the process on a background thread  
86 DispatchQueue.global(qos: .userInitiated).async {  
87     let process = Process()  
88     process.executableURL = URL(filePath: "/usr/bin/security")  
89     process.arguments = ["cms", "-D", "-i", source.path, "-o", destination.path]
```

```
90  
91     // Capture the output in a pipe
```

```
92     let pipe = Pipe()  
93     process.standardOutput = pipe  
94     process.standardError = pipe
```

```
95  
96     // Execute the command and wait for it to finish  
97     process.launch()  
98     process.waitUntilExit()
```

```
99  
100    // Read the output while the command runs  
101    let data = pipe.fileHandleForReading.readDataToEndOfFile()  
102    let output = String(data: data, encoding: .utf8)
```

```
103  
104    // Assume success if the command exits normally  
105    if process.terminationStatus == 0 {
```

```
106  
107        // Handle success on the main thread  
108        DispatchQueue.main.async {
```



```
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
84
85     // Perform the process on a background thread
86     DispatchQueue.global(qos: .userInitiated).async {
87         let process = Process()
88         process.executableURL = URL(filePath: "/usr/bin/security")
89         process.arguments = ["cms", "-D", "-i", source.path, "-o", destination.path]
90
91         // Capture the output in a pipe
92         let pipe = Pipe()
93         process.standardOutput = pipe
94         process.standardError = pipe
95
96         // Execute the command and wait for it to finish
97         process.launch()
98         process.waitUntilExit()
99
100        // Read the output while the command runs
101        let data = pipe.fileHandleForReading.readDataToEndOfFile()
102        let output = String(data: data, encoding: .utf8)
103
104        // Assume success if the command exits normally
105        if process.terminationStatus == 0 {
106
107            // Handle success on the main thread
108            DispatchQueue.main.async {
```



```
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
84
85     // Perform the process on a background thread
86     DispatchQueue.global(qos: .userInitiated).async {
87         let process = Process()
88         process.executableURL = URL(filePath: "/usr/bin/security")
89         process.arguments = ["cms", "-D", "-i", source.path, "-o", destination.path]
90
91         // Capture the output in a pipe
92         let pipe = Pipe()
93         process.standardOutput = pipe
94         process.standardError = pipe
95
96         // Execute the command and wait for it to finish
97         process.launch()
98         process.waitUntilExit()
99
100        // Read the output while the command runs
101        let data = pipe.fileHandleForReading.readDataToEndOfFile()
102        let output = String(data: data, encoding: .utf8)
103
104        // Assume success if the command exits normally
105        if process.terminationStatus == 0 {
106
107            // Handle success on the main thread
108            DispatchQueue.main.async {
```



```
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
84
85     // Perform the process on a background thread
86     DispatchQueue.global(qos: .userInitiated).async {
87         let process = Process()
88         process.executableURL = URL(filePath: "/usr/bin/security")
89         process.arguments = ["cms", "-D", "-i", source.path, "-o", destination.path]
90
91         // Capture the output in a pipe
92         let pipe = Pipe()
93         process.standardOutput = pipe
94         process.standardError = pipe
95
96         // Execute the command and wait for it to finish
97         process.launch()
98         process.waitForExit()
99
100         // Read the output while the command runs
101         let data = pipe.fileHandleForReading.readDataToEndOfFile()
102         let output = String(data: data, encoding: .utf8)
103
104         // Assume success if the command exits normally
105         if process.terminationStatus == 0 {
106
107             // Handle success on the main thread
108             DispatchQueue.main.async {
```



```
104 // Assume success if the command exits normally
105 if process.terminationStatus == 0 { ←
106
107     // Handle success on the main thread
108     DispatchQueue.main.async { ←
109         print("Mobileconfig file unsigned successfully: \(destination.path)")
110     }
111 } else {
112     // The command exited abnormally
113     // Use the main thread to set our properties for an error alert
114     DispatchQueue.main.async {
115
116         // TODO: Replace with custom errors
117         guard let output = output else {
118             self.alertMessage = "Unknown Error"
119             self.shouldPresentAlert = true
120             return
121         }
122
123         if output.contains("problem decoding") {
124             self.alertMessage = "This profile is not signed or is invalid."
125             self.shouldPresentAlert = true
126         } else {
127             self.alertMessage = output
128             self.shouldPresentAlert = true
129         }
130     }
131 }
```



```
104 // Assume success if the command exits normally
105 if process.terminationStatus == 0 {
106
107     // Handle success on the main thread
108     DispatchQueue.main.async {
109         print("Mobileconfig file unsigned successfully: \(destination.path)")
110     }
111 } else { 
112     // The command exited abnormally
113     // Use the main thread to set our properties for an error alert
114     DispatchQueue.main.async { 
115
116         // TODO: Replace with custom errors
117         guard let output = output else {
118             self.alertMessage = "Unknown Error"
119             self.shouldPresentAlert = true
120             return
121         }
122
123         if output.contains("problem decoding") {
124             self.alertMessage = "This profile is not signed or is invalid."
125             self.shouldPresentAlert = true
126         } else {
127             self.alertMessage = output
128             self.shouldPresentAlert = true
129         }
130     }
131 }
```



```
104 // Assume success if the command exits normally
105 if process.terminationStatus == 0 {
106
107     // Handle success on the main thread
108     DispatchQueue.main.async {
109         print("Mobileconfig file unsigned successfully: \(destination.path)")
110     }
111 } else {
112     // The command exited abnormally
113     // Use the main thread to set our properties for an error alert
114     DispatchQueue.main.async {
115
116         // TODO: Replace with custom errors
117         guard let output = output else {
118             self.alertMessage = "Unknown Error"
119             self.shouldPresentAlert = true
120             return
121         }
122
123         if output.contains("problem decoding") {
124             self.alertMessage = "This profile is not signed or is invalid."
125             self.shouldPresentAlert = true
126         } else {
127             self.alertMessage = output
128             self.shouldPresentAlert = true
129         }
130     }
131 }
```



```
104 // Assume success if the command exits normally
105 if process.terminationStatus == 0 {
106
107     // Handle success on the main thread
108     DispatchQueue.main.async {
109         print("Mobileconfig file unsigned successfully: \(destination.path)")
110     }
111 } else {
112     // The command exited abnormally
113     // Use the main thread to set our properties for an error alert
114     DispatchQueue.main.async {
115
116         // TODO: Replace with custom errors
117         guard let output = output else {
118             self.alertMessage = "Unknown Error"
119             self.shouldPresentAlert = true
120             return
121         }
122
123         if output.contains("problem decoding") {
124             self.alertMessage = "This profile is not signed or is invalid."
125             self.shouldPresentAlert = true
126         } else {
127             self.alertMessage = output
128             self.shouldPresentAlert = true
129         }
130     }
131 }
```



```
118         self.alertMessage = "Unknown Error"
119         self.shouldPresentAlert = true
120         return
121     }
122
123     if output.contains("problem decoding") {
124         self.alertMessage = "This profile is not signed or is invalid."
125         self.shouldPresentAlert = true
126     } else {
127         self.alertMessage = output
128         self.shouldPresentAlert = true
129     }
130 }
131 }
132 }
133 }
134 }
135
136 // Extend UTType to include the mobileconfig extension
137 extension UTType {
138     static var mobileconfig: UTType {
139         UTType(filenameExtension: "mobileconfig")!
140     }
141 }
142
```



```
56 }
57
58 // Function to open a file save panel
59 // This will allow saving an unsigned version of the profile to the selected location.
60 func saveFilePanel() {
61     let panel = NSSavePanel()
62     let fileName = selectedProfile?.lastPathComponent ?? "profile.mobileconfig"
63     panel.nameFieldStringValue = fileName
64     panel.allowedContentTypes = [.mobileconfig]
65
66     panel.begin { result in
67
68         // Ensure our optional variable is set
69         guard let selectedProfile = self.selectedProfile else {
70             fatalError("Should never happen")
71         }
72
73         // If the user chooses a destination location, unsign the profile and save it
74         if result == .OK, let destinationProfile = panel.url {
75             DispatchQueue.main.async {
76                 self.unsignProfile(from: selectedProfile, to: destinationProfile)
77             }
78         }
79     }
80 }
81
82 // Function to unsign a mobileconfig file using the security command
83 private func unsignProfile(from source: URL, to destination: URL) {
```

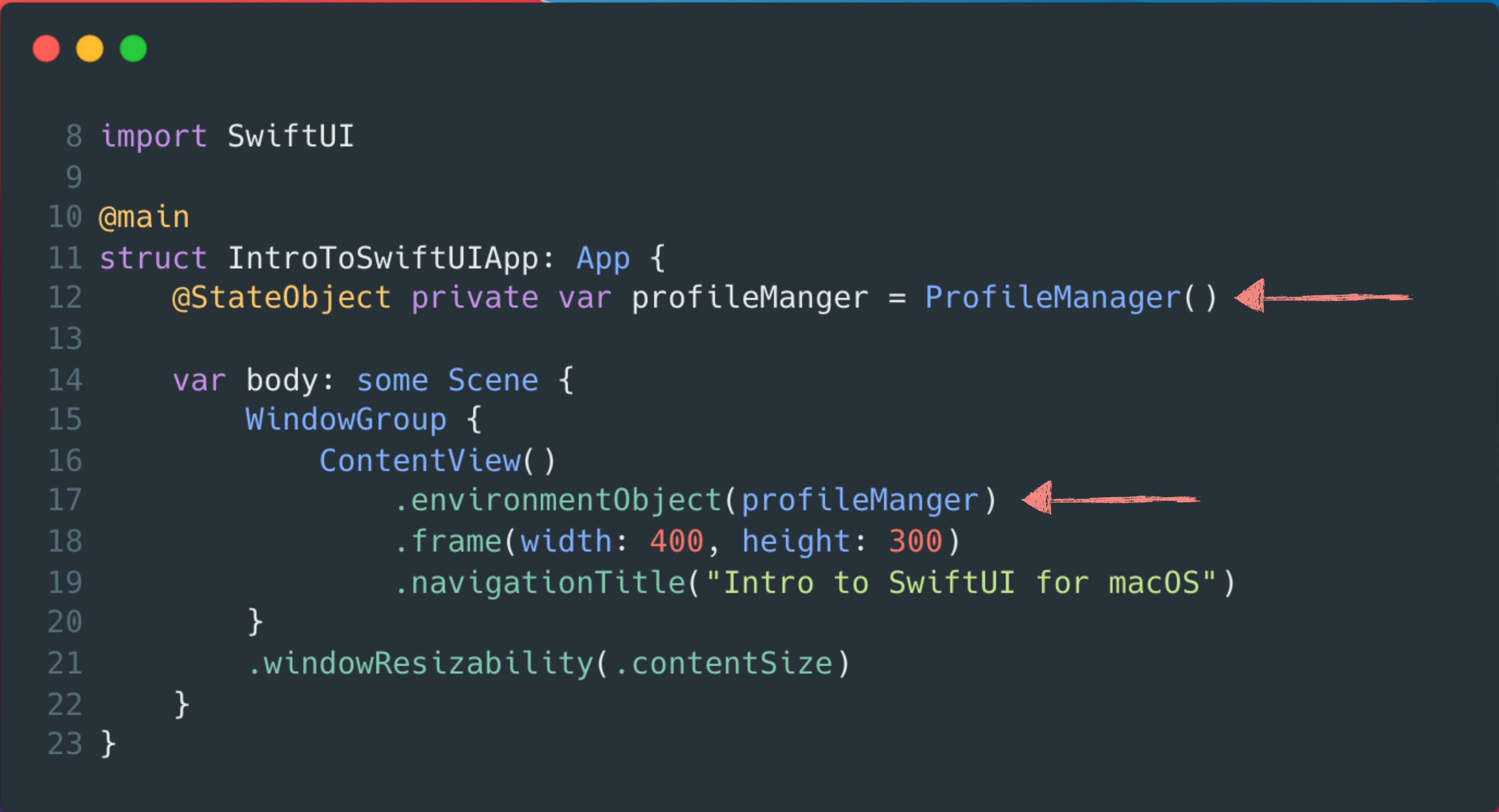


```
118         self.alertMessage = "Unknown Error"
119         self.shouldPresentAlert = true
120         return
121     }
122
123     if output.contains("problem decoding") {
124         self.alertMessage = "This profile is not signed or is invalid."
125         self.shouldPresentAlert = true
126     } else {
127         self.alertMessage = output
128         self.shouldPresentAlert = true
129     }
130 }
131 }
132 }
133 }
134 }
135
136 // Extend UTType to include the mobileconfig extension
137 extension UTType {
138     static var mobileconfig: UTType {
139         UTType(filenameExtension: "mobileconfig")!
140     }
141 }
142
```

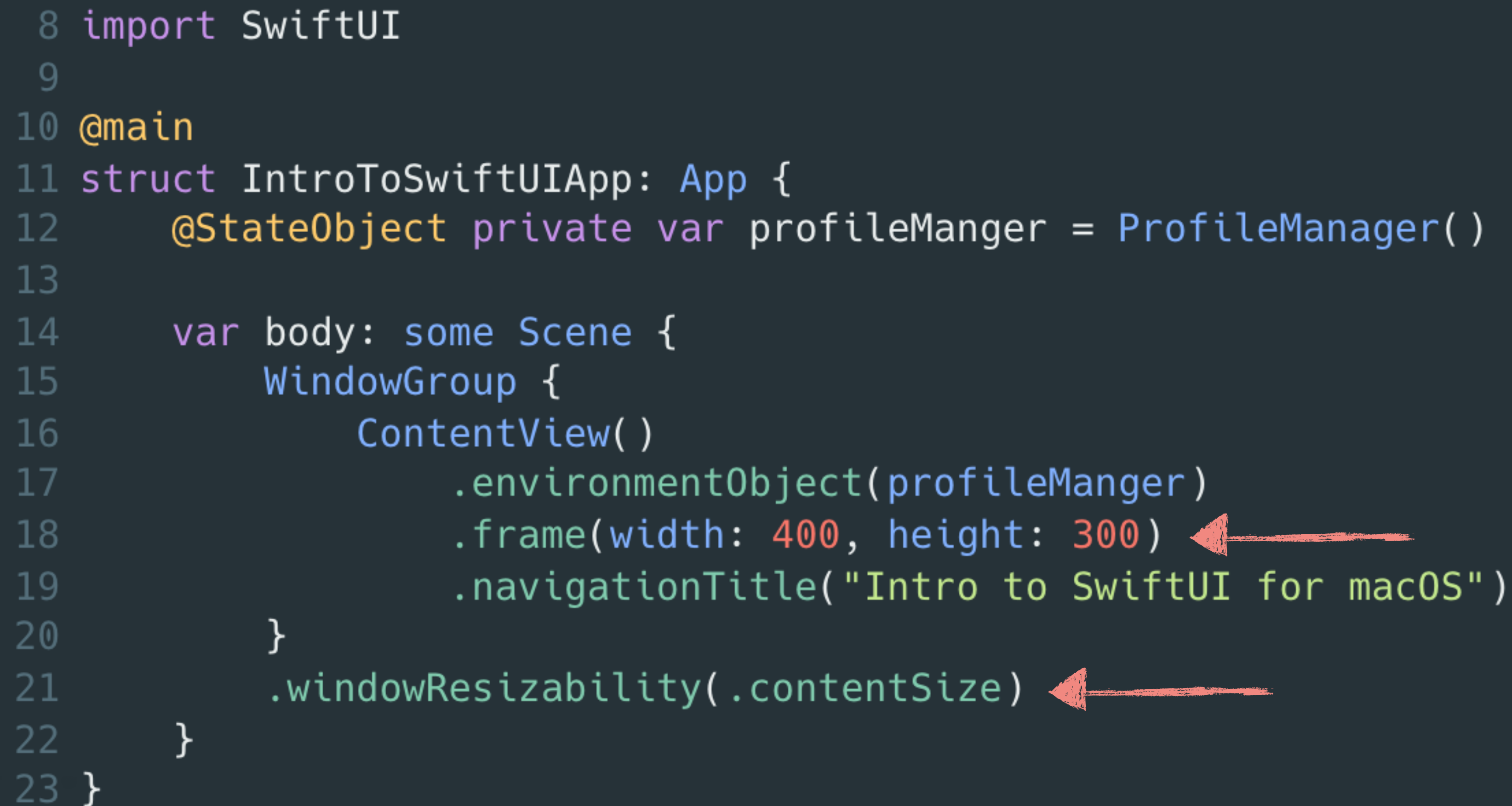

Data Model Questions

IntroToSwiftUIApp.swift

```
8 import SwiftUI
9
10 @main
11 struct IntroToSwiftUIApp: App {
12     @StateObject private var profileManger = ProfileManager()
13
14     var body: some Scene {
15         WindowGroup {
16             ContentView()
17                 .environmentObject(profileManger)
18                 .frame(width: 400, height: 300)
19                 .navigationTitle("Intro to SwiftUI for macOS")
20         }
21         .windowResizability(.contentSize)
22     }
23 }
```

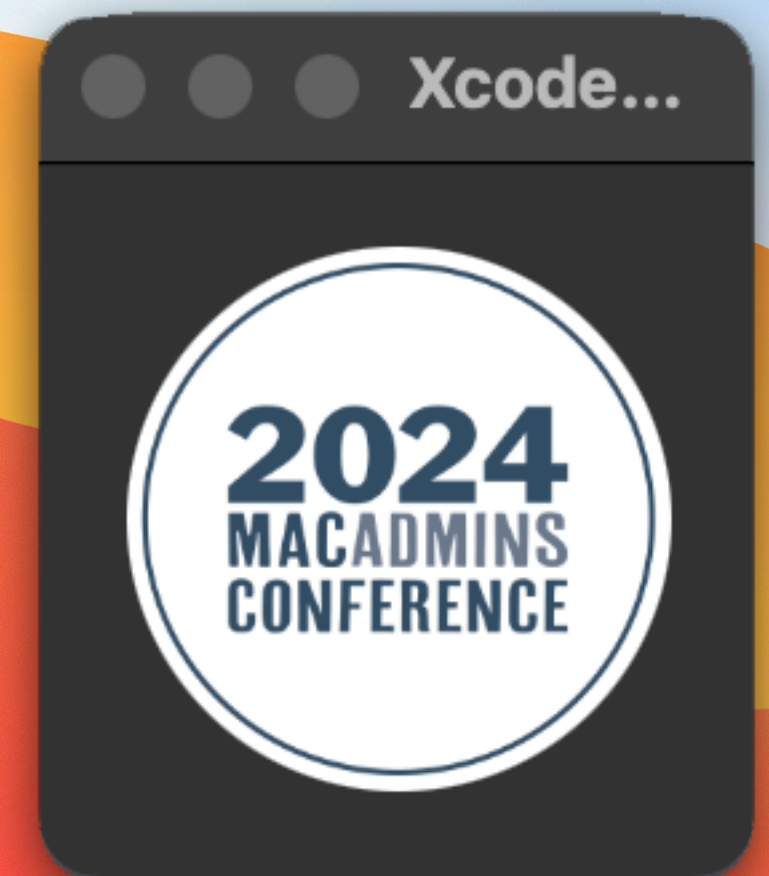
```
8 import SwiftUI
9
10 @main
11 struct IntroToSwiftUIApp: App {
12     @StateObject private var profileManger = ProfileManager()
13
14     var body: some Scene {
15         WindowGroup {
16             ContentView()
17                 .environmentObject(profileManger)
18                 .frame(width: 400, height: 300)
19                 .navigationTitle("Intro to SwiftUI for macOS")
20         }
21         .windowResizability(.contentSize)
22     }
23 }
```

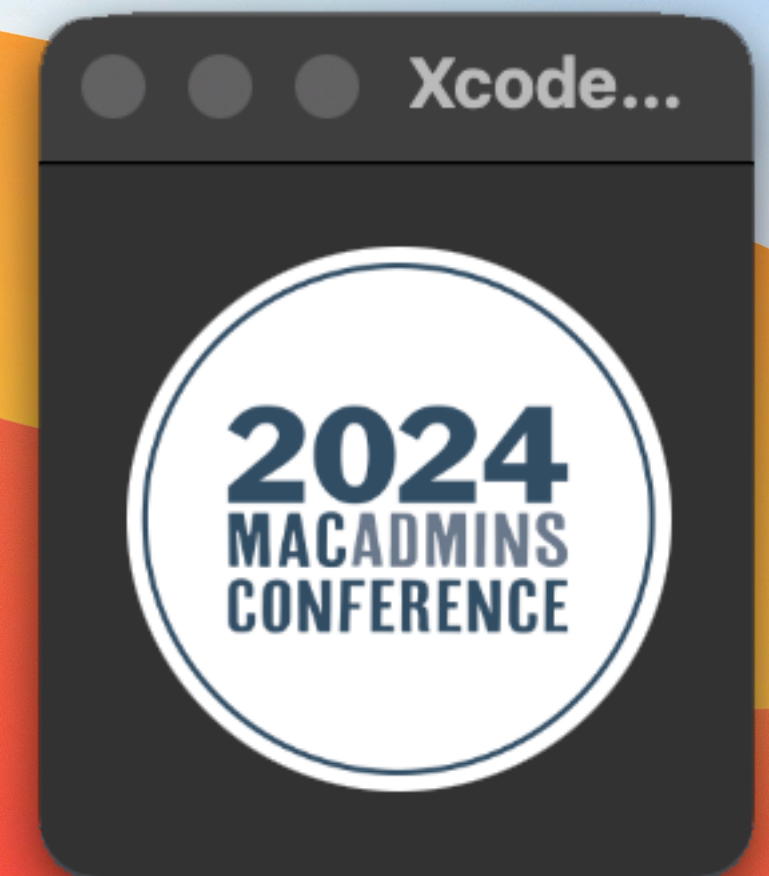
```
8 import SwiftUI
9
10 @main
11 struct IntroToSwiftUIApp: App {
12     @StateObject private var profileManger = ProfileManager()
13
14     var body: some Scene {
15         WindowGroup {
16             ContentView()
17                 .environmentObject(profileManger)
18                 .frame(width: 400, height: 300)
19                 .navigationTitle("Intro to SwiftUI for macOS")
20         }
21         .windowResizability(.contentSize)
22     }
23 }
```


MacAdminsLogoView.swift

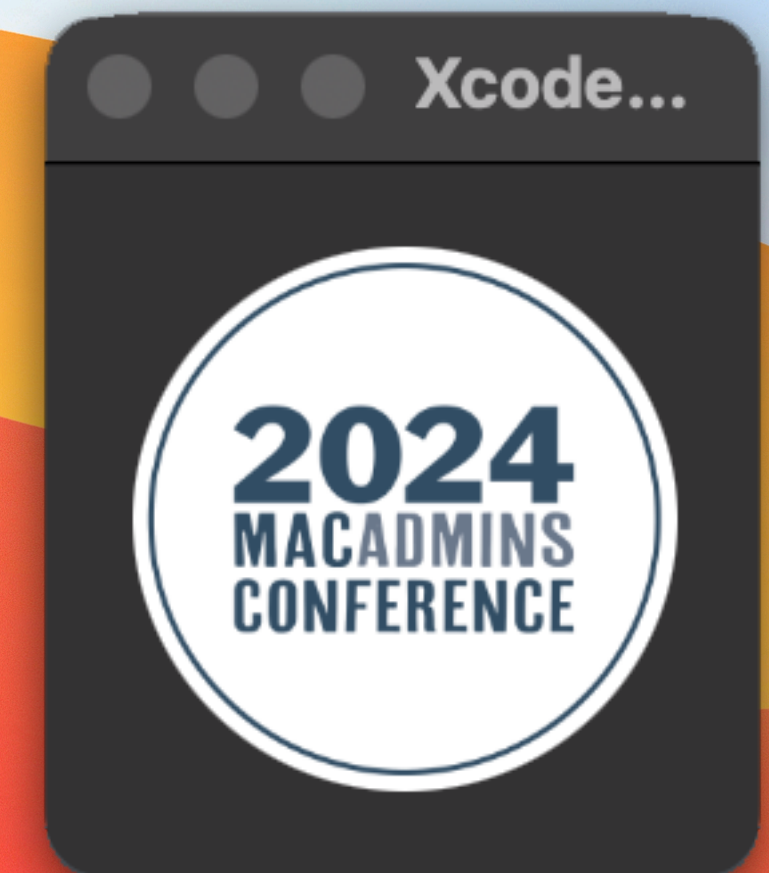
```
8 import SwiftUI
9
10 struct MacAdminsLogoView: View {
11     @State private var isHovering: Bool = false
12
13     var body: some View {
14         AsyncImage(url: psuLogo, content: { image in
15             image.resizable()
16                 .aspectRatio(contentMode: .fit)
17
18             // Add a white circle background to the logo
19             .background(
20                 Circle()
21                     .fill(.white)
22                     .frame(height: 102)
23                     .shadow(radius: 2)
24
25                 // The image is not quite centered
26                 .offset(x: 0.5, y: 0)
27             )
28     },
```



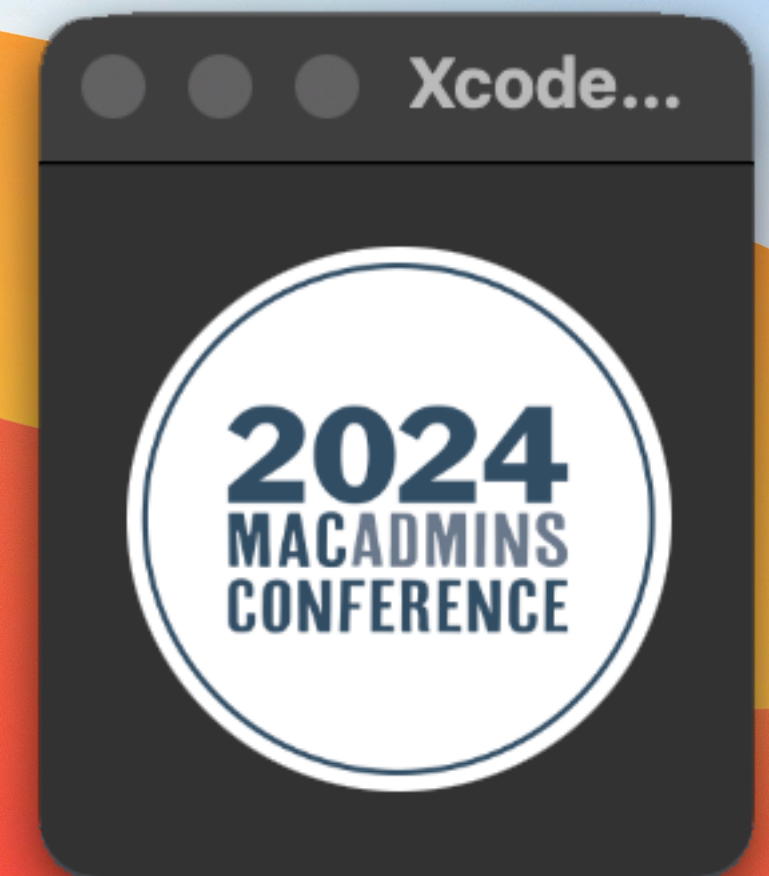

```
12
13     var body: some View {
14         AsyncImage(url: psuLogo, content: { image in
15             image.resizable()
16                 .aspectRatio(contentMode: .fit)
17
18             // Add a white circle background to the logo
19             .background(
20                 Circle()
21                     .fill(.white)
22                     .frame(height: 102)
23                     .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         })
28     },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
```



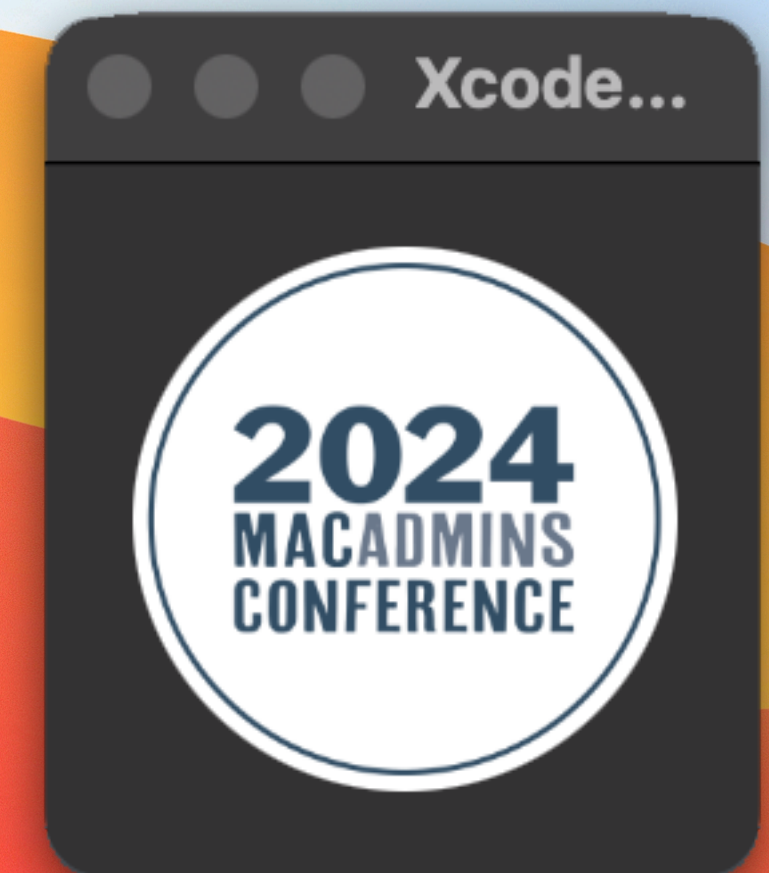

```
12
13 var body: some View {
14     AsyncImage(url: psuLogo, content: { image in
15         image.resizable()
16         .aspectRatio(contentMode: .fit)
17
18         // Add a white circle background to the logo
19         .background(
20             Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         )
28     },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
```



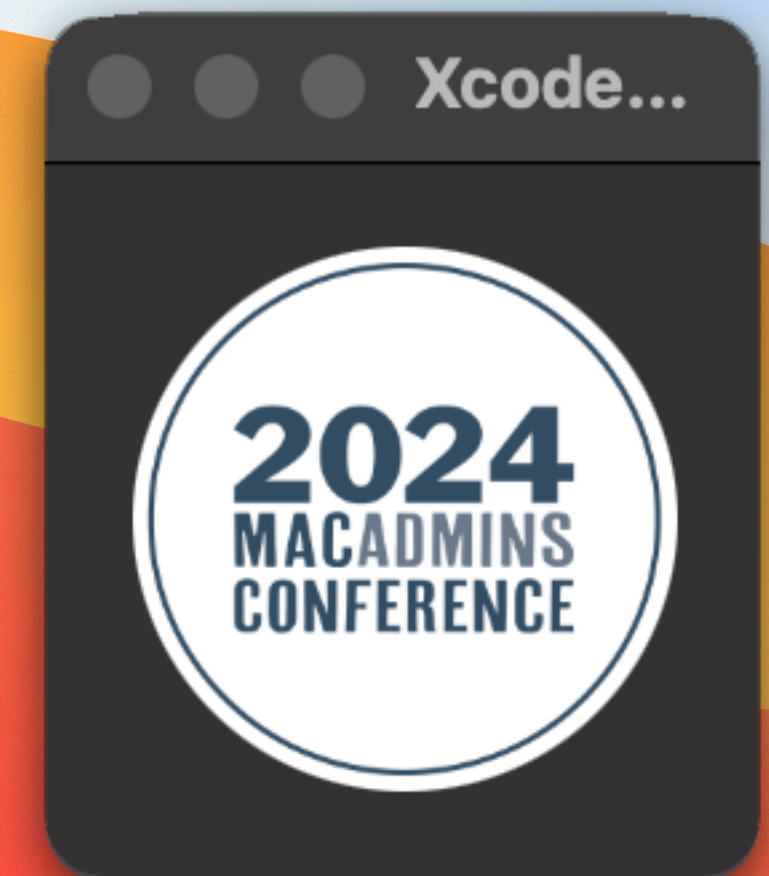

```
12
13 var body: some View {
14     AsyncImage(url: psuLogo, content: { image in
15         image.resizable()
16         .aspectRatio(contentMode: .fit)
17
18         // Add a white circle background to the logo
19         .background(
20             Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         )
28     },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
```



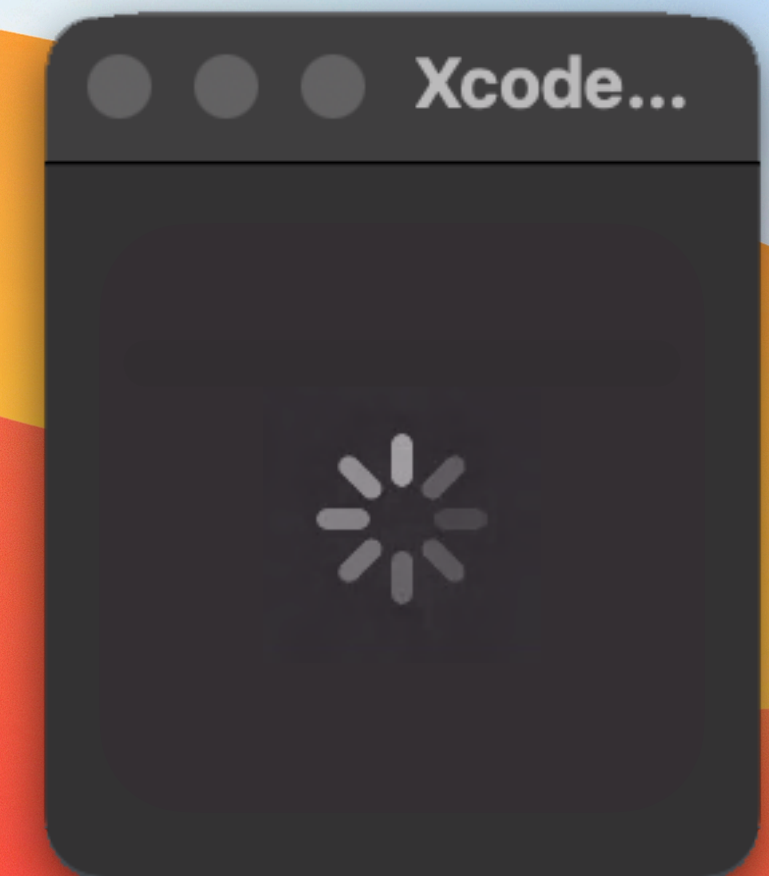

```
12
13 var body: some View {
14     AsyncImage(url: psuLogo, content: { image in
15         image.resizable()
16             .aspectRatio(contentMode: .fit)
17
18         // Add a white circle background to the logo
19         .background(
20             Circle()
21                 .fill(.white)
22                 .frame(height: 102)
23                 .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         )
28     },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
37     })
38 }
```



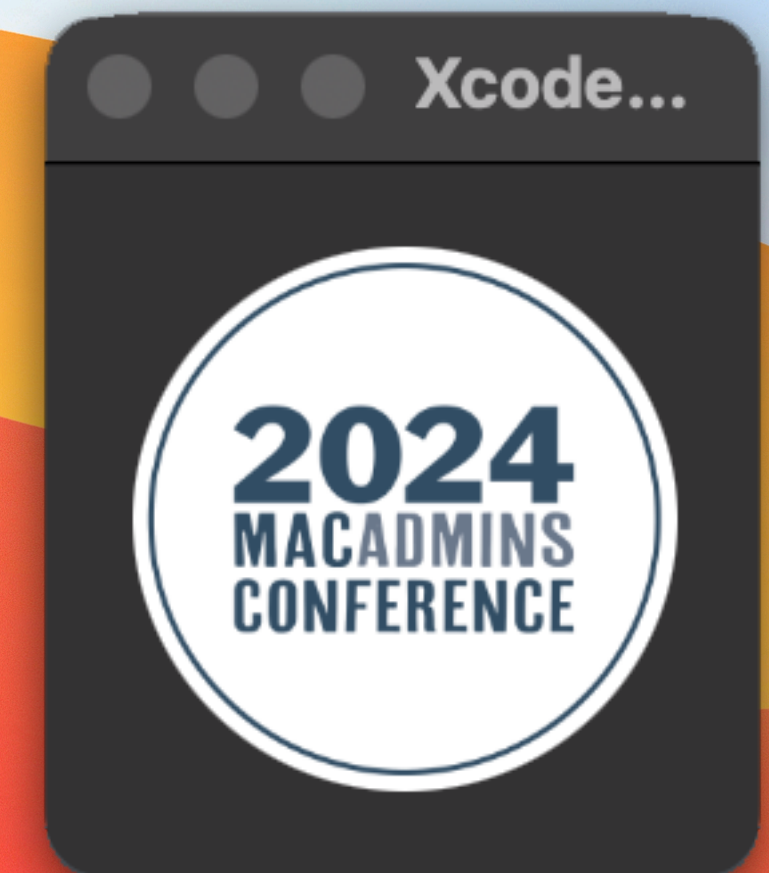

```
12
13 var body: some View {
14     AsyncImage(url: psuLogo, content: { image in
15         image.resizable()
16         .aspectRatio(contentMode: .fit)
17
18         // Add a white circle background to the logo
19         → .background(
20             Circle()
21             .fill(.white)
22             .frame(height: 102)
23             → .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         )
28     },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
```



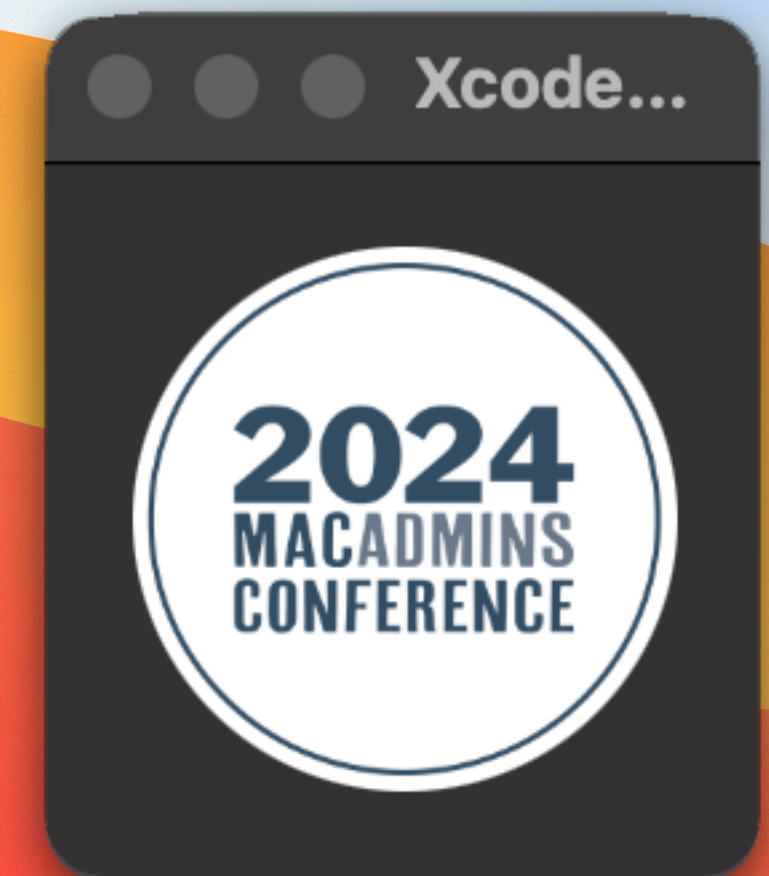

```
12
13 var body: some View {
14     AsyncImage(url: psuLogo, content: { image in
15         image.resizable()
16         .aspectRatio(contentMode: .fit)
17
18         // Add a white circle background to the logo
19         .background(
20             Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         )
28     },
29     placeholder: {
30         ProgressView() ←
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
```



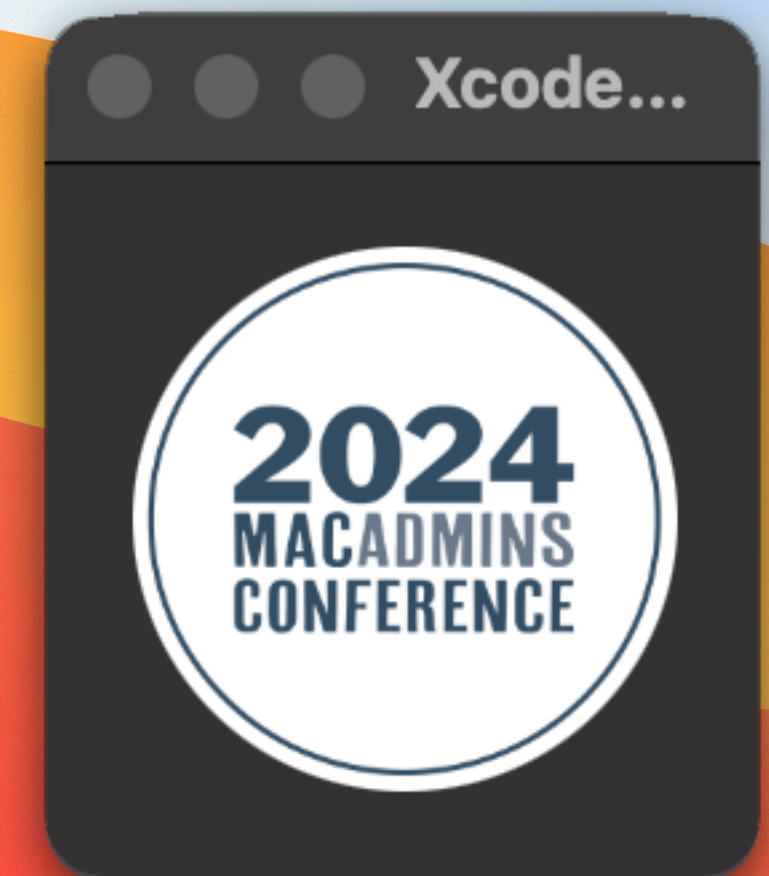

```
20         Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25         // The image is not quite centered
26         .offset(x: 0.5, y: 0)
27     )
28 },
29     placeholder: {
30     ProgressView()
31 })
32 .onHover(perform: { hovering in
33     isHovering = hovering
34 })
35 .popover(isPresented: $isHovering, content: {
36     Text("Get your hover on!")
37     .padding()
38 })
39 .frame(width: 102, height: 102)
40 .padding()
41 }
42 }
```



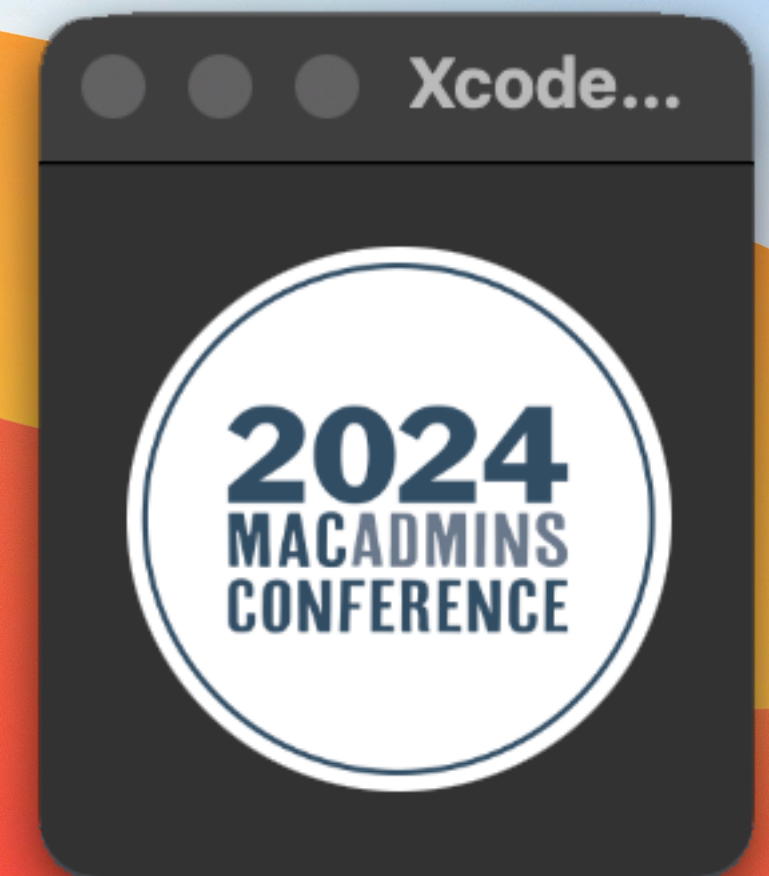

```
20         Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25             // The image is not quite centered
26             .offset(x: 0.5, y: 0)
27         )
28     },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
37         .padding()
38     })
39     .frame(width: 102, height: 102)
40     .padding()
41 }
42 }
```




```
8 import SwiftUI
9
10 struct MacAdminsLogoView: View {
11     @State private var isHovering: Bool = false ←
12
13     var body: some View {
14         AsyncImage(url: psuLogo, content: { image in
15             image.resizable()
16                 .aspectRatio(contentMode: .fit)
17
18             // Add a white circle background to the logo
19             .background(
20                 Circle()
21                     .fill(.white)
22                     .frame(height: 102)
23                     .shadow(radius: 2)
24
25                 // The image is not quite centered
26                 .offset(x: 0.5, y: 0)
27             )
28         },
```




```
20         Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25         // The image is not quite centered
26         .offset(x: 0.5, y: 0)
27     )
28 },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
37         .padding()
38     })
39     .frame(width: 102, height: 102)
40     .padding()
41 }
42 }
```

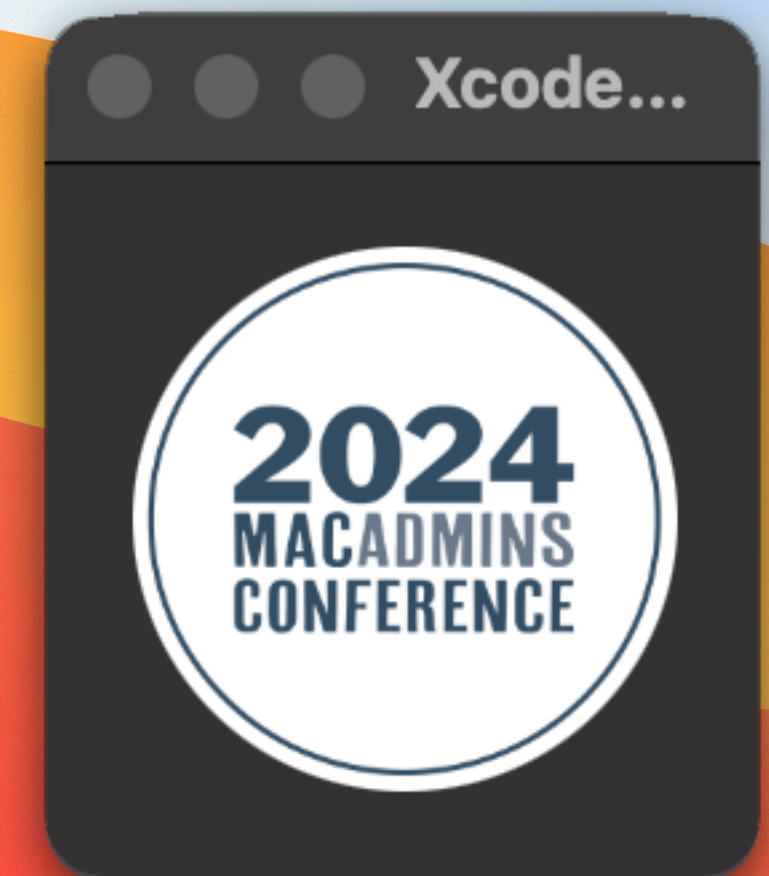



```
20         Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25         // The image is not quite centered
26         .offset(x: 0.5, y: 0)
27     )
28 },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!") ←
37         .padding()
38     })
39     .frame(width: 102, height: 102)
40     .padding()
41 }
42 }
```

Get your hover on!

2024
MACADMIN'S
CONFERENCE


```
20         Circle()
21             .fill(.white)
22             .frame(height: 102)
23             .shadow(radius: 2)
24
25         // The image is not quite centered
26         .offset(x: 0.5, y: 0)
27     )
28 },
29     placeholder: {
30         ProgressView()
31     })
32     .onHover(perform: { hovering in
33         isHovering = hovering
34     })
35     .popover(isPresented: $isHovering, content: {
36         Text("Get your hover on!")
37         .padding()
38     })
39     .frame(width: 102, height: 102)
40     .padding()
41 }
42 }
```



Logo View Questions

ButtonsView.swift

```
8 import SwiftUI
9
10 struct ButtonsView: View {
11     @EnvironmentObject var profileManager: ProfileManager
12
13     var body: some View {
14
15         // A horizontal stack containing our two buttons
16         HStack(spacing: 20) { ←
17
18             // The "Select Profile" button
19             Button(action: {
20                 profileManager.openFilePanel()
21             }, label: {
22                 HStack(spacing: 10) {
23                     Text("Select Profile")
24                     Image(systemName: "checkmark.seal.f")
25                 }
26             })
27             .keyboardShortcut(KeyboardShortcut("o"))
28
```

Xcode Previews

Select Profile



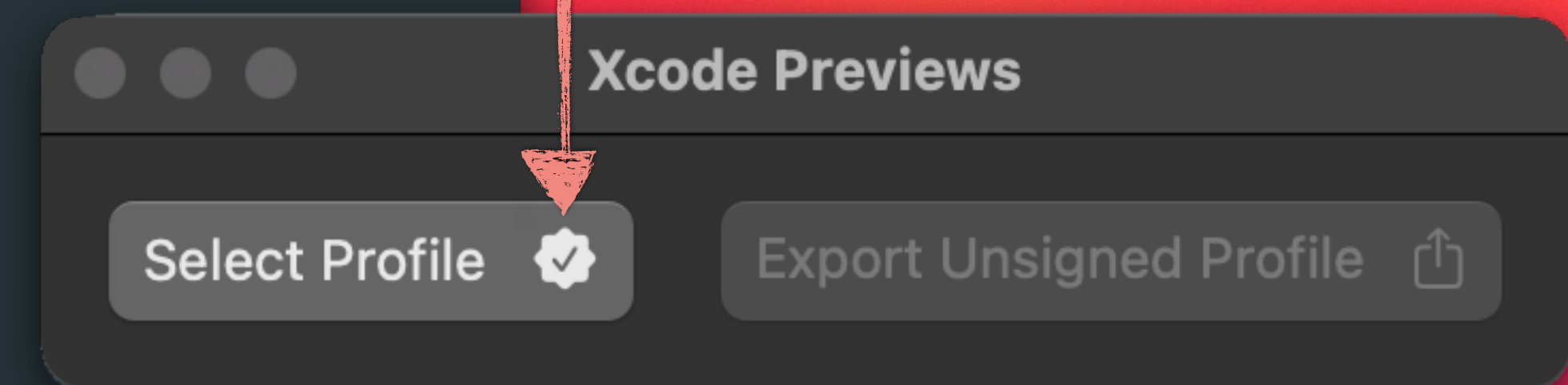
Export Unsigned Profile

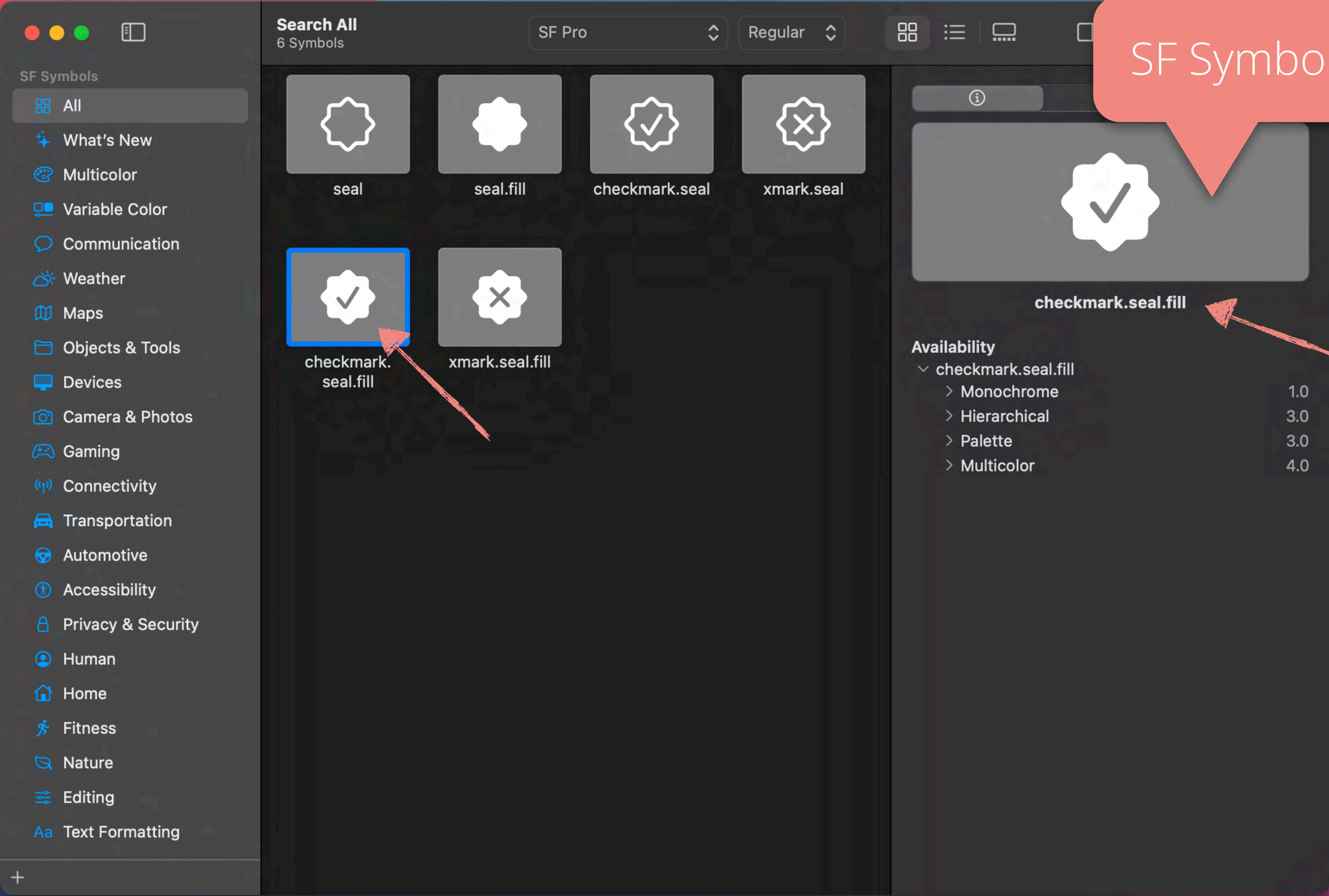



```
12
13 var body: some View {
14
15     // A horizontal stack containing our two buttons
16     HStack(spacing: 20) {
17
18         // The "Select Profile" button
19         Button(action: {
20             profileManager.openFilePanel()
21         }, label: {
22             HStack(spacing: 10) {
23                 Text("Select Profile")
24                 Image(systemName: "checkmark.seal.fill")
25             }
26         })
27         .keyboardShortcut(KeyboardShortcut("o"))
28
29         // The "Export" button
30         Button(action: {
31             profileManager.saveFilePanel()
32         }, label: {
33             HStack(spacing: 10) {
34                 Text("Export Unsigned Profile")
35                 Image(systemName: "square.and.arrow.up")
36             }
37         })
38     }
39 }
```

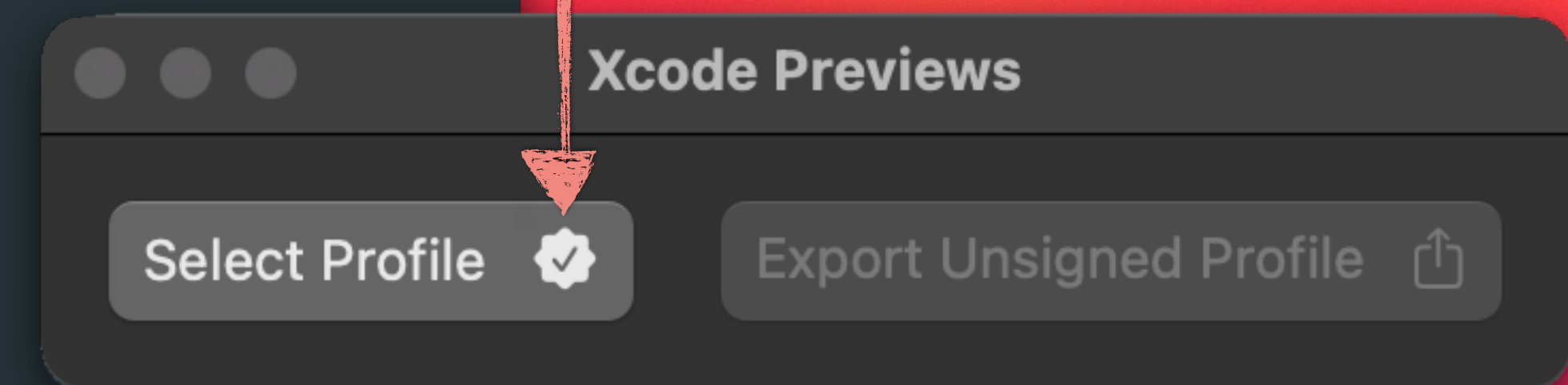



```
12
13 var body: some View {
14
15     // A horizontal stack containing our two buttons
16     HStack(spacing: 20) {
17
18         // The "Select Profile" button
19         Button(action: {
20             profileManager.openFilePanel()
21         }, label: {
22             HStack(spacing: 10) {
23                 Text("Select Profile")
24                 Image(systemName: "checkmark.seal.fill")
25             }
26         })
27         .keyboardShortcut(KeyboardShortcut("o"))
28
29         // The "Export" button
30         Button(action: {
31             profileManager.saveFilePanel()
32         }, label: {
33             HStack(spacing: 10) {
34                 Text("Export Unsigned Profile")
35                 Image(systemName: "square.and.arrow.up")
36             }
37         })
38     }
39 }
```

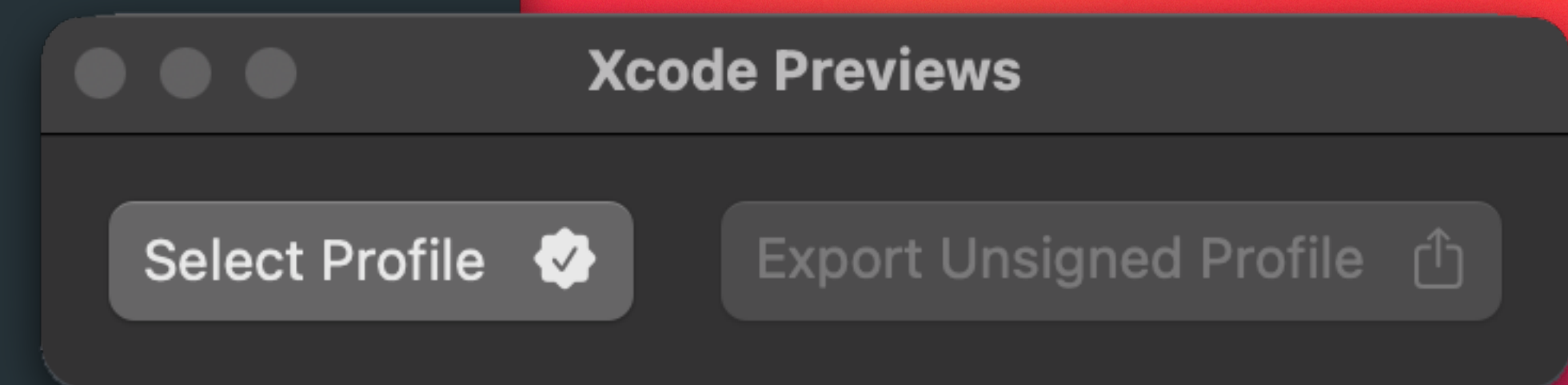




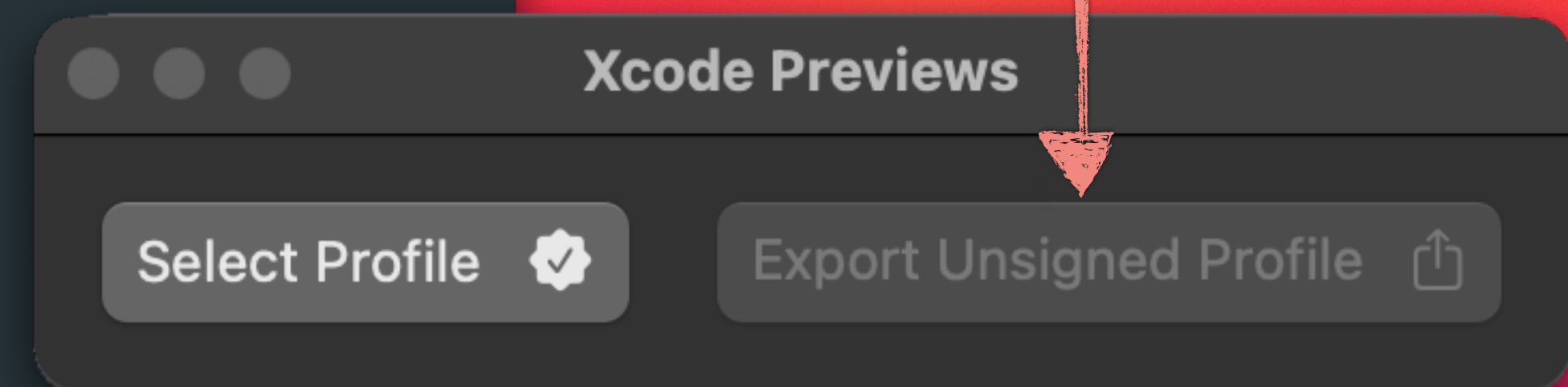

```
12
13 var body: some View {
14
15     // A horizontal stack containing our two buttons
16     HStack(spacing: 20) {
17
18         // The "Select Profile" button
19         Button(action: {
20             profileManager.openFilePanel()
21         }, label: {
22             HStack(spacing: 10) {
23                 Text("Select Profile")
24                 Image(systemName: "checkmark.seal.fill")
25             }
26         })
27         .keyboardShortcut(KeyboardShortcut("o"))
28
29         // The "Export" button
30         Button(action: {
31             profileManager.saveFilePanel()
32         }, label: {
33             HStack(spacing: 10) {
34                 Text("Export Unsigned Profile")
35                 Image(systemName: "square.and.arrow.up")
36             }
37         })
38     }
39 }
```



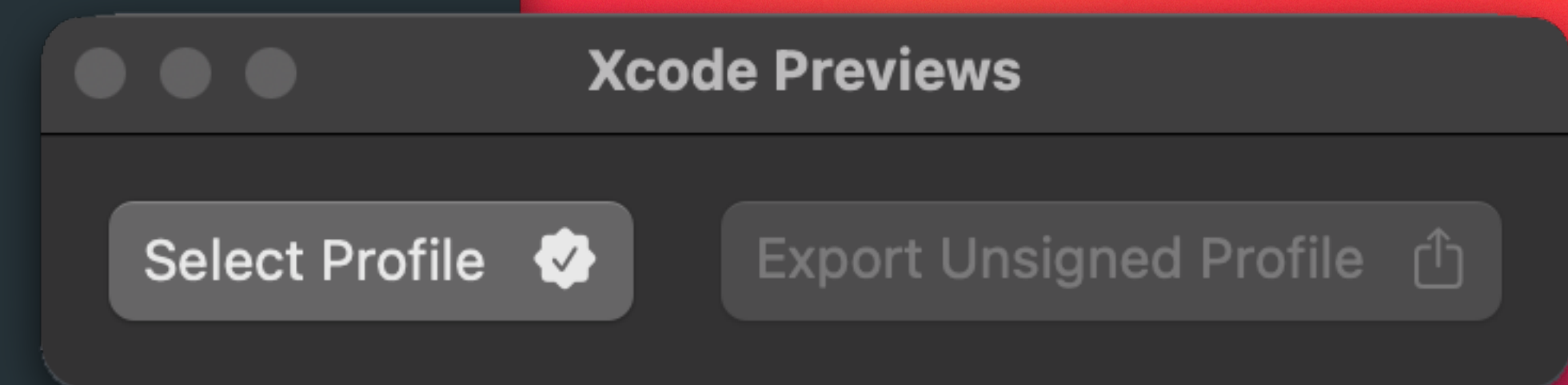

```
12
13 var body: some View {
14
15     // A horizontal stack containing our two buttons
16     HStack(spacing: 20) {
17
18         // The "Select Profile" button
19         Button(action: {
20             profileManager.openFilePanel()
21         }, label: {
22             HStack(spacing: 10) {
23                 Text("Select Profile")
24                 Image(systemName: "checkmark.seal.fill")
25             }
26         })
27         .keyboardShortcut(KeyboardShortcut("o"))
28
29         // The "Export" button
30         Button(action: {
31             profileManager.saveFilePanel()
32         }, label: {
33             HStack(spacing: 10) {
34                 Text("Export Unsigned Profile")
35                 Image(systemName: "square.and.arrow.up")
36             }
37         })
38     }
39 }
```



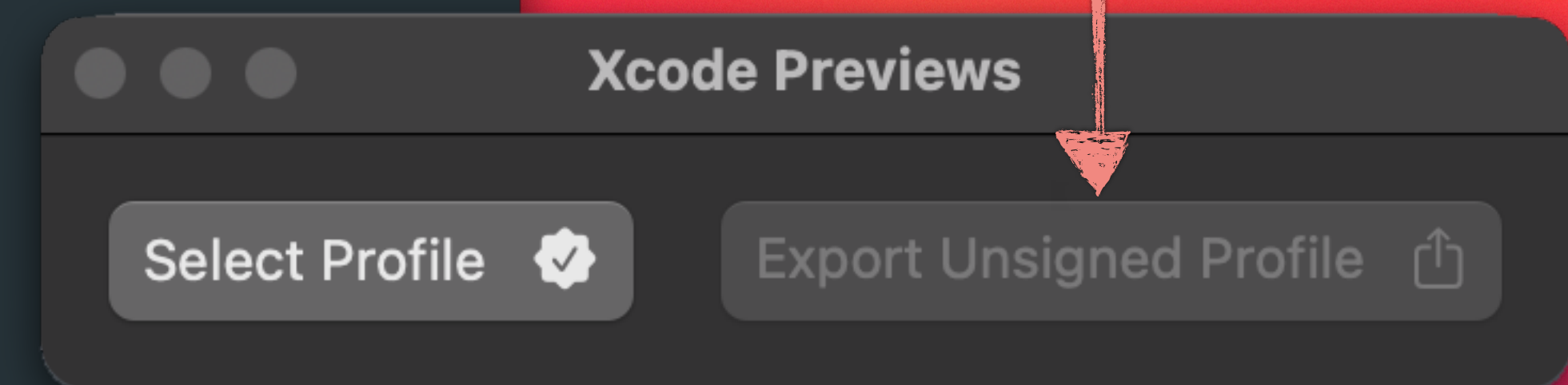

```
24         Image(systemName: "checkmark.seal.fill")
25     }
26 })
27 .keyboardShortcut(KeyboardShortcut("o"))
28
29 // The "Export" button
30 Button(action: {
31     profileManager.saveFilePanel()
32 }, label: {
33     HStack(spacing: 10) {
34         Text("Export Unsigned Profile")
35         Image(systemName: "square.and.arrow.up")
36     }
37 })
38 .keyboardShortcut(KeyboardShortcut("s"))
39
40 // Disable the button if no profile is selected
41 .disabled(profileManager.selectedProfile == nil)
42 }
43 .controlSize(.extraLarge)
44 .padding()
45 }
46 }
```



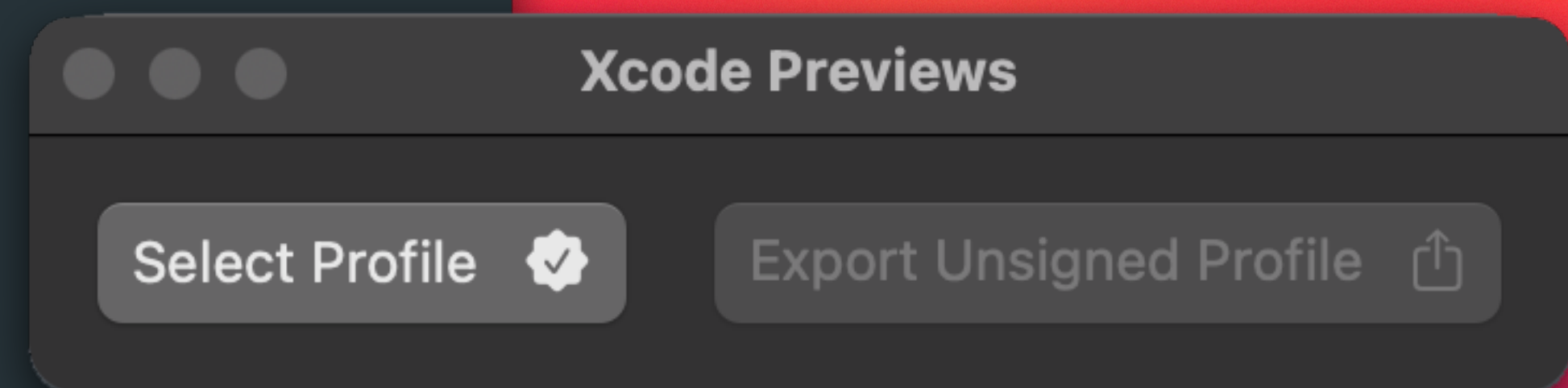

```
24         Image(systemName: "checkmark.seal.fill")
25     }
26 })
27 .keyboardShortcut(KeyboardShortcut("o"))
28
29 // The "Export" button
30 Button(action: {
31     profileManager.saveFilePanel()
32 }, label: {
33     HStack(spacing: 10) {
34         Text("Export Unsigned Profile")
35         Image(systemName: "square.and.arrow.up")
36     }
37 })
38 .keyboardShortcut(KeyboardShortcut("s"))
39
40 // Disable the button if no profile is selected
41 .disabled(profileManager.selectedProfile == nil)
42 }
43 .controlSize(.extraLarge)
44 .padding()
45 }
46 }
```




```
24         Image(systemName: "checkmark.seal.fill")
25     }
26 })
27 .keyboardShortcut(KeyboardShortcut("o"))
28
29 // The "Export" button
30 Button(action: {
31     profileManager.saveFilePanel()
32 }, label: {
33     HStack(spacing: 10) {
34         Text("Export Unsigned Profile")
35         Image(systemName: "square.and.arrow.up")
36     }
37 })
38 .keyboardShortcut(KeyboardShortcut("s"))
39
40 // Disable the button if no profile is selected
41 .disabled(profileManager.selectedProfile == nil)
42 }
43 .controlSize(.extraLarge)
44 .padding()
45 }
46 }
```




```
24         Image(systemName: "checkmark.seal.fill")
25     }
26 })
27 .keyboardShortcut(KeyboardShortcut("o"))
28
29 // The "Export" button
30 Button(action: {
31     profileManager.saveFilePanel()
32 }, label: {
33     HStack(spacing: 10) {
34         Text("Export Unsigned Profile")
35         Image(systemName: "square.and.arrow.up")
36     }
37 })
38 .keyboardShortcut(KeyboardShortcut("s"))
39
40 // Disable the button if no profile is selected
41 .disabled(profileManager.selectedProfile == nil)
42 }
43 .controlSize(.extraLarge)
44 .padding()
45 }
46 }
```



Buttons View Questions

ContentView.swift

```
8 import SwiftUI
9
10 struct ContentView: View {
11     @EnvironmentObject var profileManager: ProfileManager
12
13     var body: some View {
14         VStack {
15
16             // Our logo view
17             MacAdminsLogoView()
18
19             // Display the name of the selected profile
20             if let profileName = profileManager.profileName {
21                 HStack(spacing: 3) {
22                     Image(nsImage: profileManager.profileIcon)
23                         .resizable()
24                         .scaledToFit()
25                         .frame(height: 18)
26                     Text(profileName)
27                 }
28                 .padding()
29             } else {
30                 Text("No profile selected")
31                     .padding()
32             }
33 }
```



```
8 import SwiftUI
9
10 struct ContentView: View {
11     @EnvironmentObject var profileManager: ProfileManager ←
12
13     var body: some View {
14         VStack {
15
16             // Our logo view
17             MacAdminsLogoView() ←
18
19             // Display the name of the selected profile
20             if let profileName = profileManager.profileName {
21                 HStack(spacing: 3) {
22                     Image(nsImage: profileManager.profileIcon)
23                         .resizable()
24                         .scaledToFit()
25                         .frame(height: 18)
26                     Text(profileName)
27                 }
28                 .padding()
29             } else {
30                 Text("No profile selected")
31                     .padding()
32             }
33 }
```



```
9
10 struct ContentView: View {
11     @EnvironmentObject var profileManager: ProfileManager
12
13     var body: some View {
14         VStack {
15
16             // Our logo view
17             MacAdminsLogoView()
18
19             // Display the name of the selected profile
20             if let profileName = profileManager.profileName {
21                 HStack(spacing: 3) {
22                     Image(nsImage: profileManager.profileIcon)
23                         .resizable()
24                         .scaledToFit()
25                         .frame(height: 18)
26                     Text(profileName)
27                 }
28                 .padding()
29             } else {
30                 Text("No profile selected")
31                     .padding()
32             }
33
34             // Display our buttons
35             ButtonsView()
36         }
37     }
}
```



```
9
10 struct ContentView: View {
11     @EnvironmentObject var profileManager: ProfileManager
12
13     var body: some View {
14         VStack {
15
16             // Our logo view
17             MacAdminsLogoView()
18
19             // Display the name of the selected profile
20             if let profileName = profileManager.profileName {
21                 HStack(spacing: 3) {
22                     Image(nsImage: profileManager.profileIcon)
23                     .resizable()
24                     .scaledToFit()
25                     .frame(height: 18)
26                     Text(profileName)
27                 }
28                 .padding()
29             } else {
30                 Text("No profile selected")
31                 .padding()
32             }
33
34             // Display our buttons
35             ButtonsView()
36         }
37     }

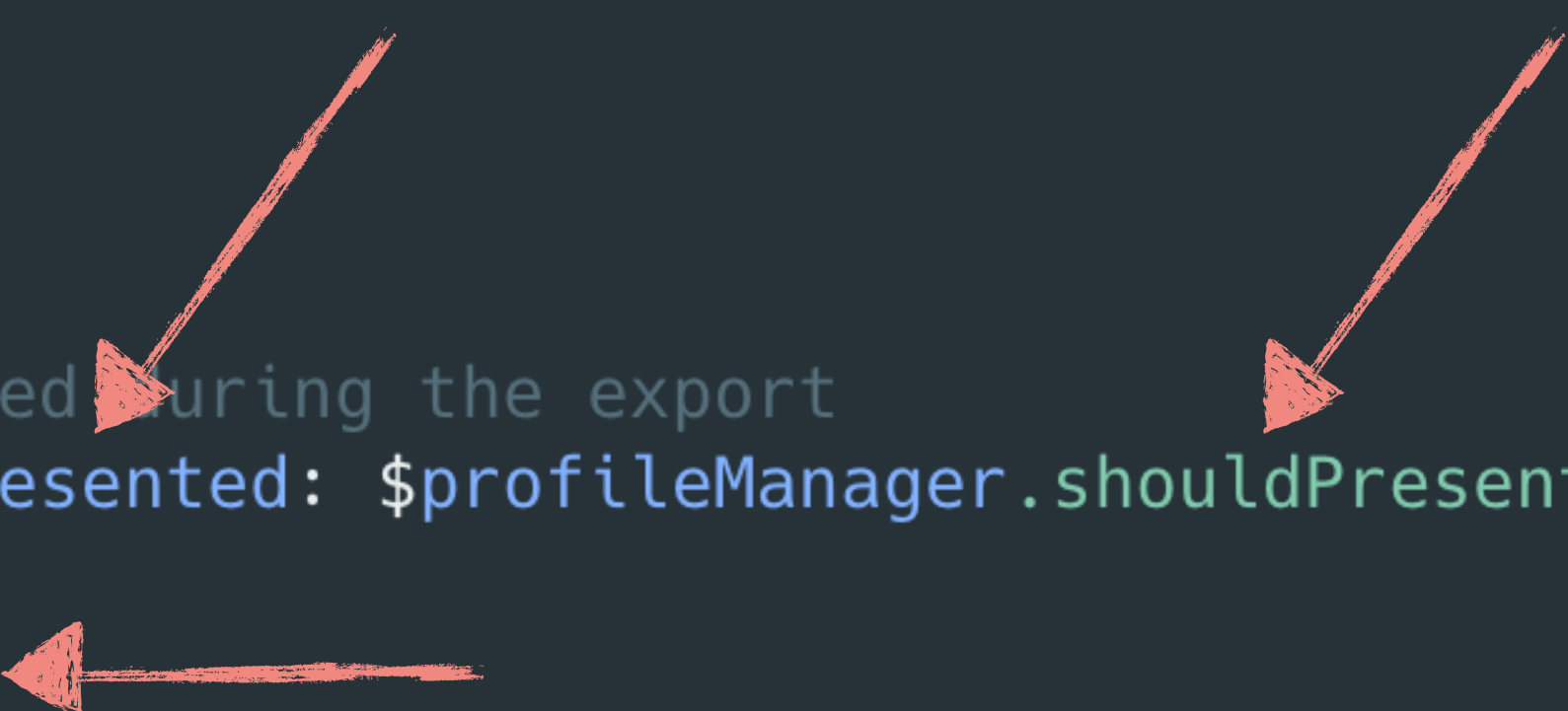
```



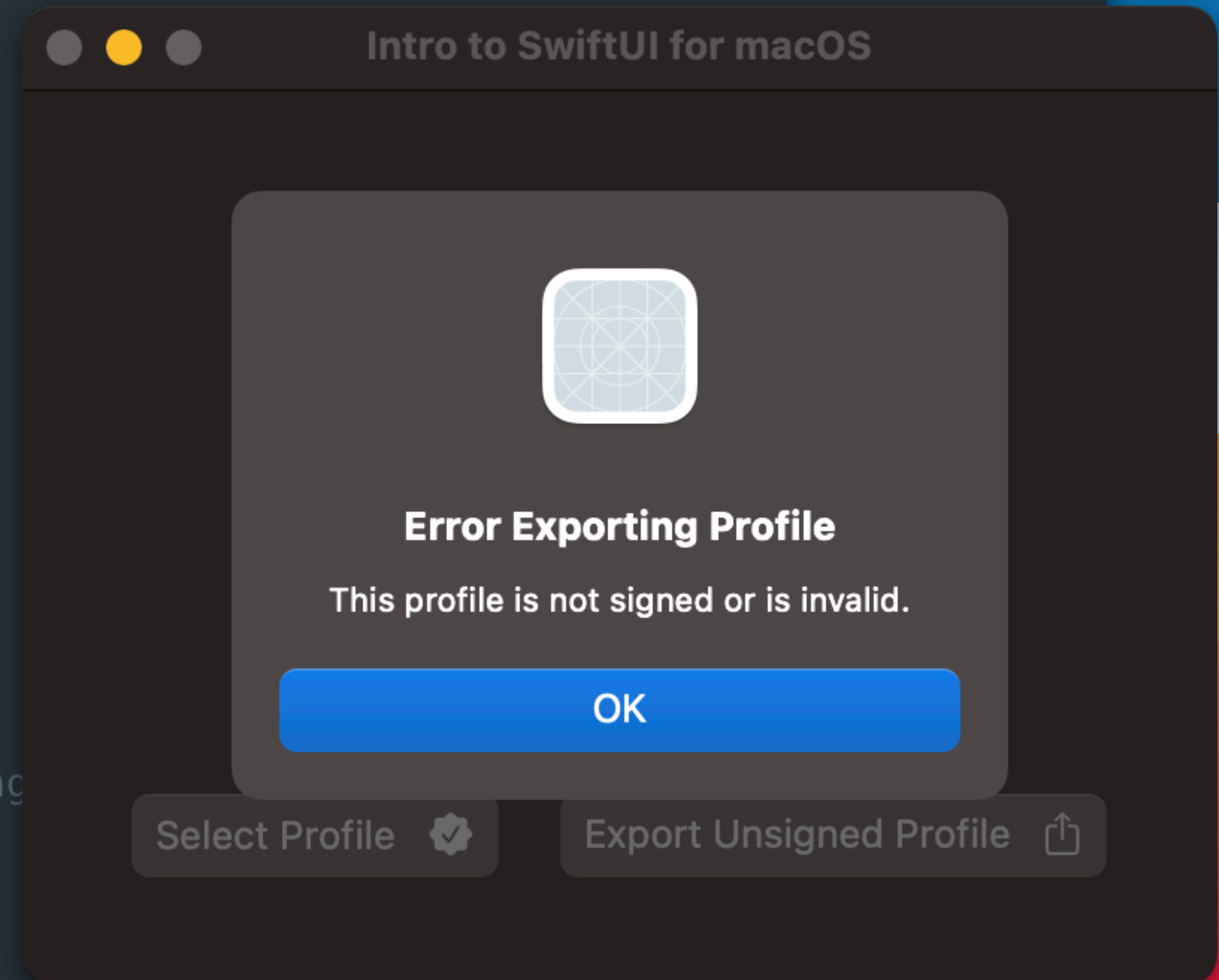
```
19 // Display the name of the selected profile
20 if let profileName = profileManager.profileName {
21     HStack(spacing: 3) {
22         Image(nsImage: profileManager.profileIcon)
23             .resizable()
24             .scaledToFit()
25             .frame(height: 18)
26         Text(profileName)
27     }
28     .padding()
29 } else {
30     Text("No profile selected")
31     .padding()
32 }
33
34 // Display our buttons
35 ButtonsView()
36 }
37
38 // Present an alert if an error occurred during the export
39 .alert("Error Exporting Profile", isPresented: $profileManager.shouldPresentAlert) {
40     message: {
41         Text(profileManager.alertMessage)
42     }
43     .frame(maxWidth: .infinity, maxHeight: .infinity)
44 }
45 }
```



```
19 // Display the name of the selected profile
20 if let profileName = profileManager.profileName {
21     HStack(spacing: 3) {
22         Image(nsImage: profileManager.profileIcon)
23             .resizable()
24             .scaledToFit()
25             .frame(height: 18)
26         Text(profileName)
27     }
28     .padding()
29 } else {
30     Text("No profile selected")
31     .padding()
32 }
33
34 // Display our buttons
35 ButtonsView()
36 }
37
38 // Present an alert if an error occurred during the export
39 .alert("Error Exporting Profile", isPresented: $profileManager.shouldPresentAlert) {
40 } message: {
41     Text(profileManager.alertMessage)
42 }
43 .frame(maxWidth: .infinity, maxHeight: .infinity)
44 }
45 }
```




```
19 // Display the name of the selected profile
20 if let profileName = profileManager.profileName {
21     HStack(spacing: 3) {
22         Image(nsImage: profileManager.profileIcon)
23             .resizable()
24             .scaledToFit()
25             .frame(height: 18)
26         Text(profileName)
27     }
28     .padding()
29 } else {
30     Text("No profile selected")
31     .padding()
32 }
33
34 // Display our buttons
35 ButtonsView()
36 }
37
38 // Present an alert if an error occurred during
39 .alert("Error Exporting Profile", isPresented:
40 } message: {
41     Text(profileManager.alertMessage)
42 }
43 .frame(maxWidth: .infinity, maxHeight: .infinity)
44 }
45 }
```



Does the app function
for you?



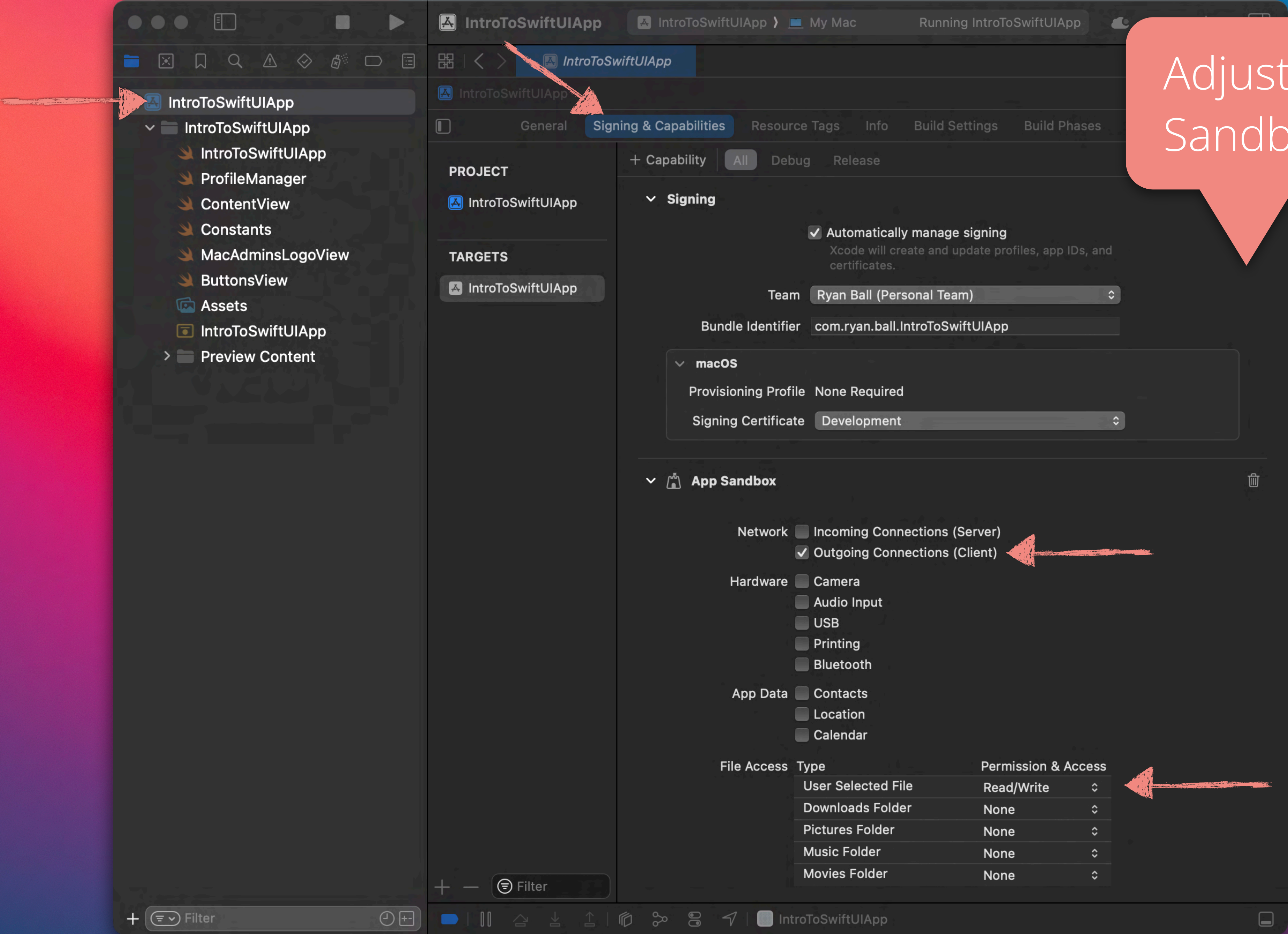
App Sandbox



App Sandbox

- Isolates apps from each other and from the system
- The sandbox can limit potential damage by compromised apps
- Apps can request access to specific user directories via user consent
- These protections help to improve security and privacy





Adjust the App Sandbox settings

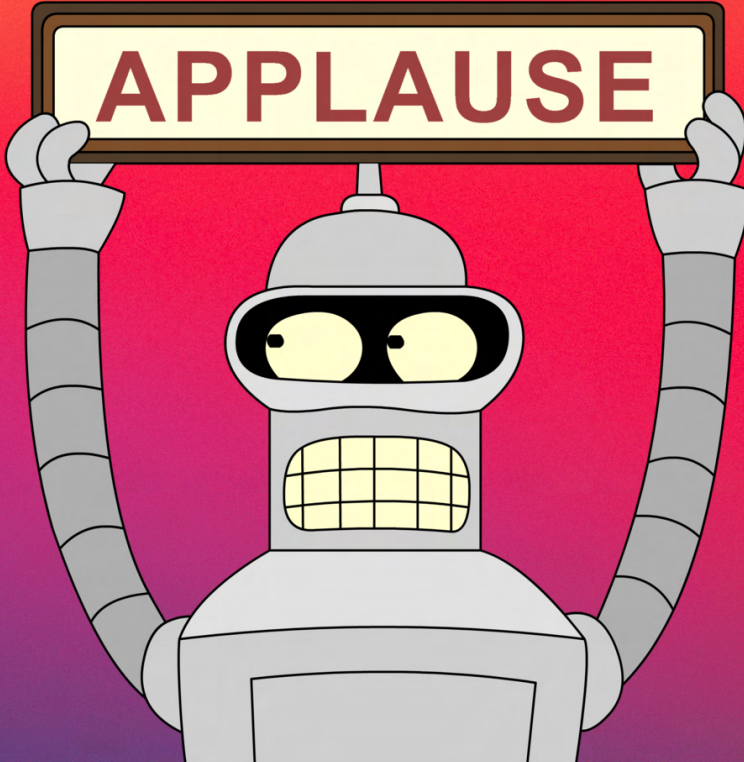
Intro to SwiftUI for macOS



No profile selected

Select Profile

Export Unsigned Profile



Thank You

We'd love to connect and discuss this topic more!

*Visit alectrona.com/MacAdmins2024 for
presentation slides and more info!*

Give us feedback on today's talk:
bit.ly/psumac-2024-19



Ryan Ball, Director of Engineering

linkedin.com/in/ryangball

ryan@alectrona.com



Electrona

Electrona.com