SWIFTUI LESSONS

FOR UI/APPKIT DEVELOPERS

Federico Zanetello

★★★★★ fivestars.blog • @zntfdr

SWIFTUI IS NOT UIAPPKIT 2

SWIFTUI IS NOT UIKIT/APPKIT 2

- Complete paradigm shift (imperative ➡️ declarative)
- We're all starting from scratch
- Knowledge of earlier UI framework is:
  - 👍🏻 insightful
  - 😕 not helpful
  - 😮 might even be a disadvantage (!!!)

SWIFTUI VIEWS ARE RECIPES

SWIFTUI VIEWS ARE RECIPES

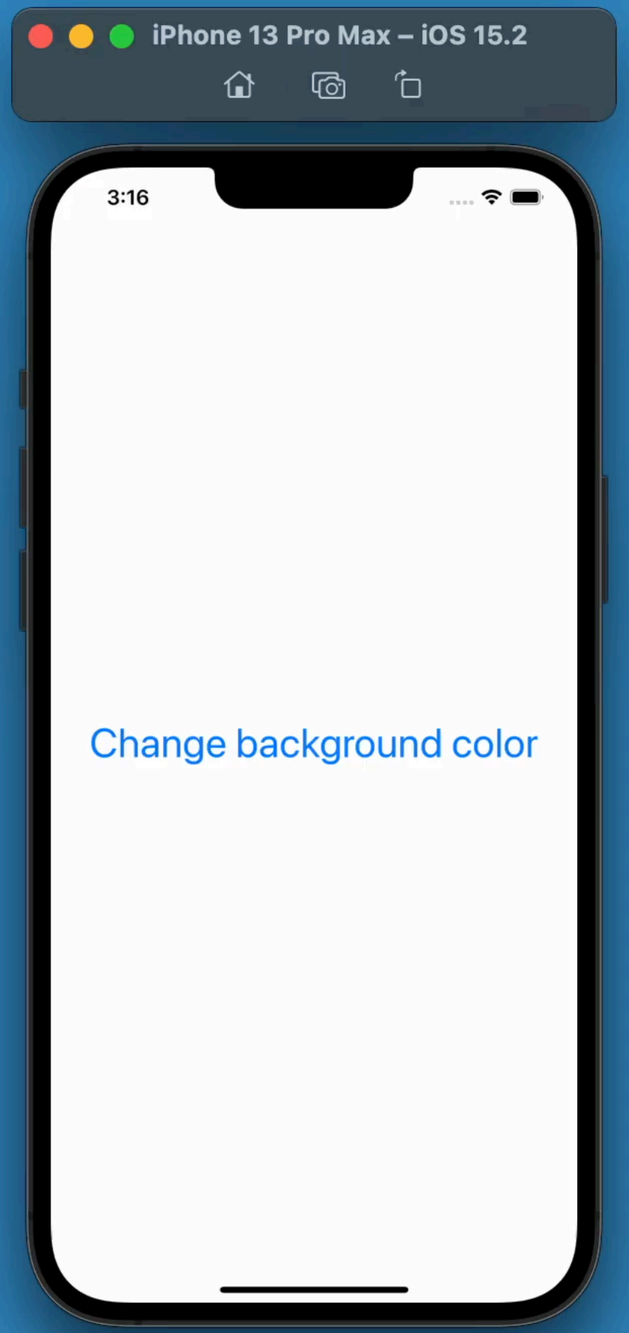
- where we declare how views look and behave...
- ...but no longer manage transitions & similar
- we give up full control on views

NEW WAY TO MAKE CHANGES VIA STATE

```
struct FSView: View {
  @State private var backgroundColor: Color = .white

  var body: some View {
    ZStack {
      backgroundColor.ignoresSafeArea()

      Button("Change background color") {
        backgroundColor = [.red, .white, .black]
          .filter { $0 != backgroundColor }
          .randomElement()!
      }
    }
  }
}
```



NEW VIEW COMMUNICATION WAYS

» Bindings

```
struct FSView: View {
  @State private var isOn = false

  var body: some View {
    Toggle(
      "Enable X",
      isOn: $isOn
    )
  }
}
```

IT'S EXPECTED TO STRUGGLE WHEN LEARNING SWIFTUI

THE SWIFTUI STRUGGLE

- even simple things are going to be hard
- all of us are still figuring it out
- when people say something is impossible in SwiftUI, it means they haven't figured it out yet
- the struggle is real...until things will *just* click

EVERYTHING IS A View, NOT EVERY View IS A VIEW

EVENTS ARE OBSERVED AND DELIVERED TO VIEWS

- onAppear
- onDisappear
- task
- onReceive
- onChange
- onDrag
- onDrop
- onHover
- onSubmit
- …

Views can be containers[a]

```
struct ContainerView: View {
  @StateObject var model = ContainerModel()

  var body: some View {
    ActualUI(
      elements: model.elements,
      onElementTap: model.onTap(element:)
    )
    .onAppear(perform: model.onAppear)
  }
}
```

[a] https://swiftwithmajid.com/2019/07/31/introducing-container-views-in-swiftui/

SWIFTUI COORDINATOR ARCHITECTURE[β]

```
struct FlowCoordinator: View {
  @State private var routes: [Route<Screen>] = [.root(.firstScreen)]
  var onCompletion: () -> Void

  var body: some View {
    Router($routes) { screen, _ in
      switch screen {
        case .firstScreen:
          FirstScreen(onCompletion: { routes.push(.secondScreen) })
        case .secondScreen:
          SecondScreen(onCompletion: onCompletion)
      }
    }
  }
}
```

[β] https://github.com/johnpatrickmorgan/FlowStacks

SWIFTUI IS SLOW*

SWIFTUI IS SLOW*

- Optimizations become implementation details
  - newer Xcode, faster app! 🏎
  - there's a limit to how much we can get for free
- *SwiftUI is as performant as our code makes it so:
  - The more parameters/states/dependencies a view has, the less performance we might get[γ]
    - Isolate state as much as possible[δ]
    - Publish only relevant changes
    - Make each view observe as little as possible[ε]

[γ] https://www.fivestars.blog/articles/app-state/
[δ] https://www.wwdcnotes.com/notes/wwdc21/10022/
[ε] https://github.com/cookednick/Observable/

FEEDBACK ASSISTANT[λ] IS YOUR (NEW) FRIEND

[λ] https://feedbackassistant.apple.com

FEEDBACK ASSISTANT PRO TIPS

- Be as narrow and concise as possible
- File as early as possible
- Follow-up when new SDKs are out
- Add reproduction code (for bugs)
- Bonus: attach a video demo
- Describe your case (for suggestions)

UIKIT/APPKIT ARE NOT GOING ANYWHERE

UIKIT/APPKIT ARE NOT GOING ANYWHERE 🤗

- SwiftUI uses both AppKit and UIKit behind the scenes
- SwiftUI is just another tool in our belt
- it's ok to mix and match!
  - UIViewRepresentable NSViewRepresentable
  - UIViewControllerRepresentable NSViewControllerRepresentable
  - UIHostingController NSHostingController
- Feel free to experiment!

SWIFTUI LESSONS

FOR UI/APPKIT DEVELOPERS

Federico Zanetello

★★★★★ fivestars.blog • @zntfdr