SwiftUI

What is it?

Why is it?

4 Core Principles

Declarative

- describe the result

Automatic functionality

- Sane defaults
- Built in accessability, dark mode, localizability, spacing/insets

Compositional

- Small components make up bigger UI

Simple state management

What I like

- Less code
- Easier state management
- Realtime preview
- Code is the single source of truth
- Apple controls the whole stack:
  - chips -> device -> language -> OS -> SDK
- Works with UIKit

What I don't like

- More View types: CollectionView,etc
- Errors are complex
- Data flow is not truly one-way
- Not OSS/Poor documentation
- iOS 13 and up only
- I feel old 👴

Demo

- Tutorial

Data flow

@State

- locally owned by view and requires initial default value
- Value type
- Use should be limited because State data is owned by view, while putting it in model is preferable
- Something that controls what is currently visible (on off switch) seems like ideal use

```
struct PlayerView : View {
  let episode: Episode
  @State private var isPlaying: Bool = false

  var body: some View {
    VStack {
      Text(episode.title)
      Text(episode.showTitle).font(.caption).foregroundColor(.gray)
      Button(action: {
        self.isPlaying.toggle()
      }) {
        Image(systemName: isPlaying ? "pause.circle" : "play.circle")
      }
    }
  }
}
```

@Binding

- not owned and does not require initial default
- Set by **`$propertyName`** and passed to child view that owns it as a @State
- Read and write

```
struct PlayButton : View {
  @Binding var isPlaying: Bool

  var body: some View {
    Button(action: {
      self.isPlaying.toggle()
    }) {
      Image(systemName: isPlaying ? "pause.circle" : "play.circle")
    }
  }
}
```

BindableObject

- data external to the View (stuff other than touch events, etc)
- Good for existing model integration
  - use RepresentableContext for sharing w/ UIView/VCs
- Comform store for example to BindableObject protocol **`didChange`**
- Use **`@ObjectBinding`** property wrapper
- Reference
- Add Combine Publisher to use

```
struct MyView : View {
  @ObjectBinding var model: MyModelObject
  ...
}
MyView(model: modelInstance)
```

@EnvironmentObject

- Widely shared data like theme etc

```
struct PlayerView : View {
  @EnvironmentObject var player: PodcastPlayerStore

  var body: some View {
    VStack {
      Text(player.currentEpisode.title)
        .foregroundColor(isPlaying ? .white : .gray)
      Text(player.currentEpisode.showTitle)
        .font(.caption).foregroundColor(.gray)

      PlayButton(isPlaying: $player.isPlaying)
      Text("\(player.currentTime, formatter: playheadTimeFormatter)")
    }
  }
}
```