- **Project:** Guess the Flags
- **Instructor**: Paul Hudson
- **Instruction**: https://www.hackingwithswift.com/100/swiftui/20
  https://www.hackingwithswift.com/100/swiftui/21
  https://github.com/twostraws/HackingWithSwift/tree/main/SwiftUI/project2
  twostraws/HackingWithSwift: The project source code for hackingwithswift.com (github.com)
  https://www.hackingwithswift.com/books/ios-swiftui/guess-the-flag-wrap-up

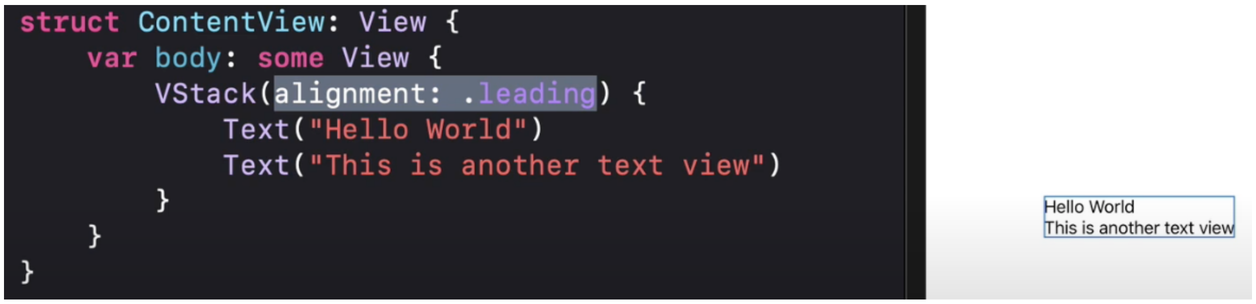
**Guess the Flags** requires users to pick the flag corresponding to the given country's name. After the users pick their answer, show their scores, and proceed to continue to the next round.

- Challenge 1: Store and display players' score
  - Add 1 when they guess the correct flag
  - Deduct 0.5 when they guess the wrong flag
- Challenge 2: Show the current score right after three flags
- Challenge 3:

## Theory Part

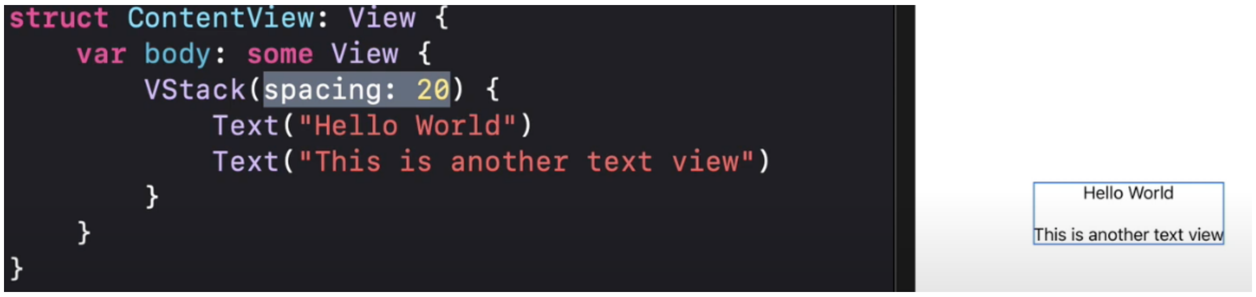
### 1/. Stacks

- **VStack** (vertical stack)
  - VStack(alignment: .leading)



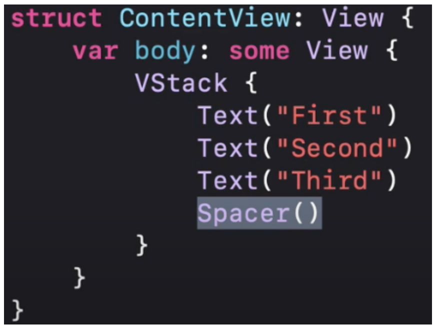
```
struct ContentView: View {
    var body: some View {
        VStack(alignment: .leading) {
            Text("Hello World")
            Text("This is another text view")
        }
    }
}
```

  - VStack(spacing: 20)

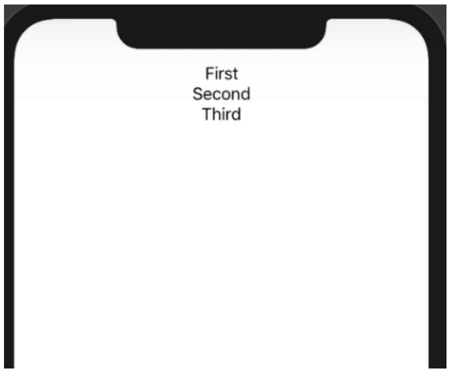


```
struct ContentView: View {
    var body: some View {
        VStack(spacing: 20) {
            Text("Hello World")
            Text("This is another text view")
        }
    }
}
```

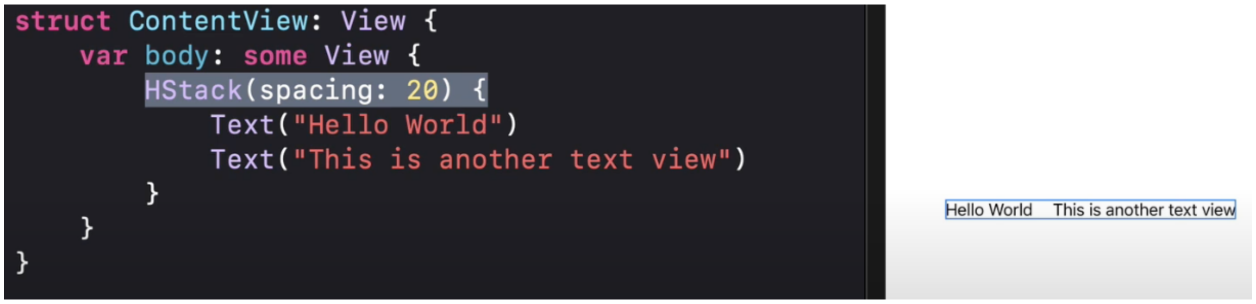
  - VStack and Spacer(): spacer pushes things to the top



```
struct ContentView: View {
    var body: some View {
        VStack {
            Text("First")
            Text("Second")
            Text("Third")
            Spacer()
        }
    }
}
```

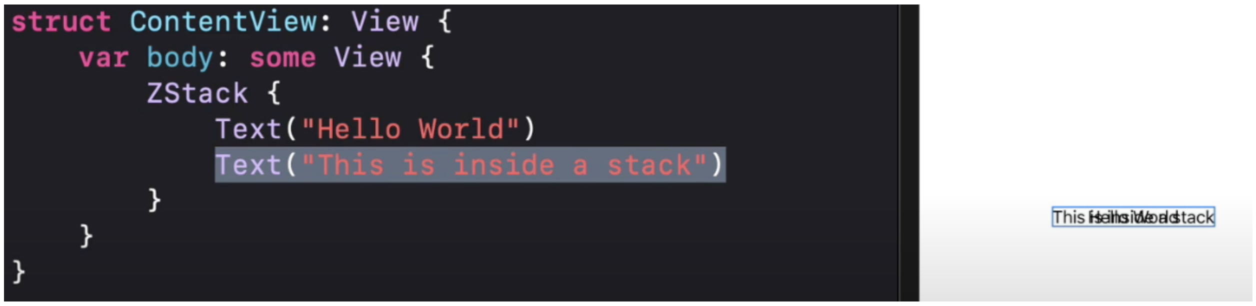


- **HStack** (horizontal stack)
    - HStack(spacing: 20)



```
struct ContentView: View {
    var body: some View {
        HStack(spacing: 20) {
            Text("Hello World")
            Text("This is another text view")
        }
    }
}
```

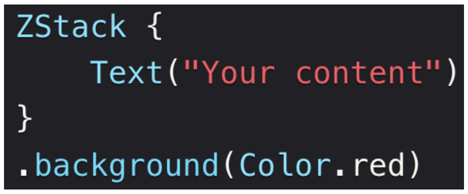
- **ZStack** (depth stack)
    - One thing is on top of another



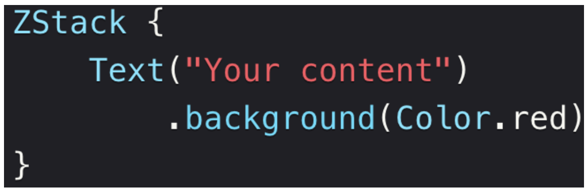
```
struct ContentView: View {
    var body: some View {
        ZStack {
            Text("Hello World")
            Text("This is inside a stack")
        }
    }
}
```

**2/. Adding colors** (Colors are views in SwiftUI)

- **Highlight the text:** .background(Color.red)

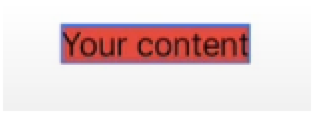


```
ZStack {
    Text("Your content")
}
.background(Color.red)
```

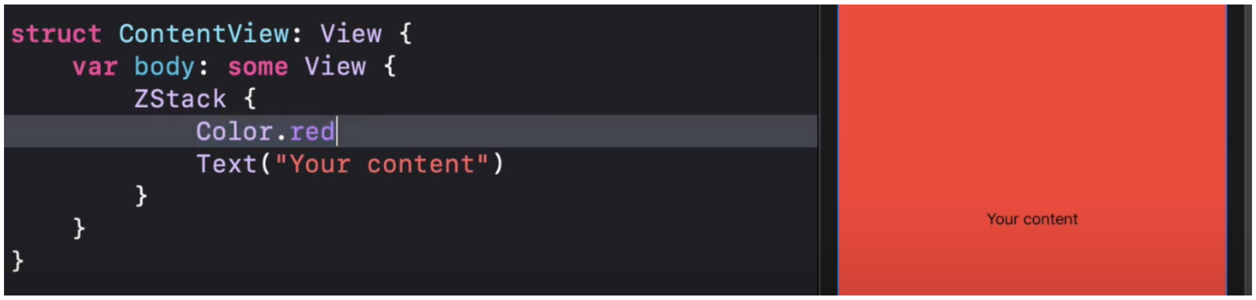


```
ZStack {
    Text("Your content")
        .background(Color.red)
}
```

^^^ these 2 codes work similar to each other

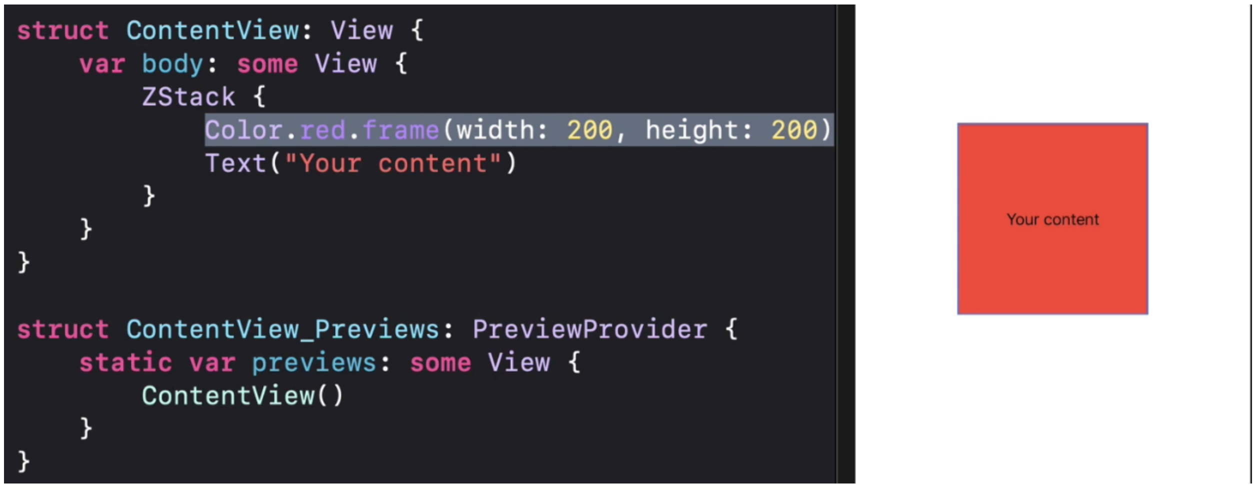


- **Make the whole screen red:** Color.red



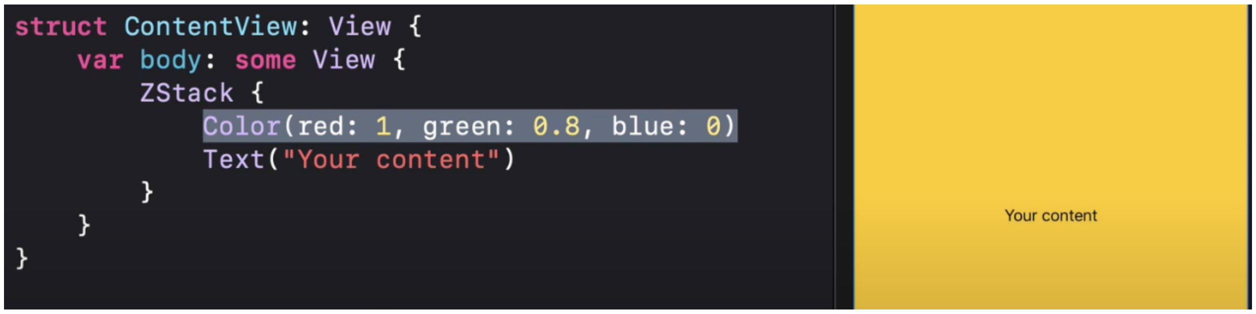
```
struct ContentView: View {
    var body: some View {
        ZStack {
            Color.red
            Text("Your content")
        }
    }
}
```

- **Make partial screen red:** Color.red.frame(width: 200, height: 200)



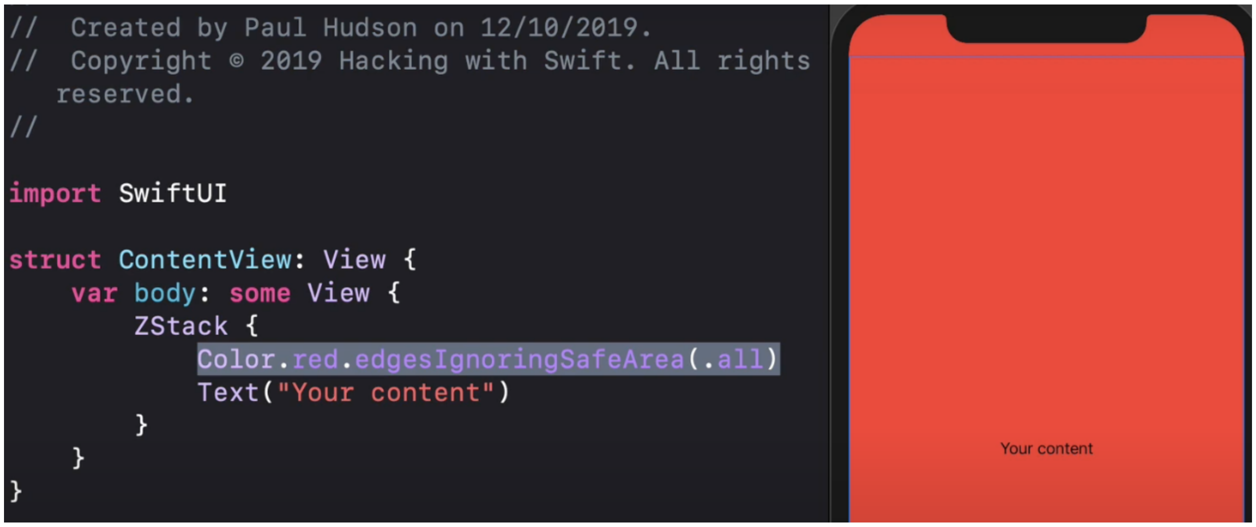
```
struct ContentView: View {
    var body: some View {
        ZStack {
            Color.red.frame(width: 200, height: 200)
            Text("Your content")
        }
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

- **Specify a specific color: passing values in [0, 1]**
  Color(red: 1, green: 0.8, blue: 0)



```
struct ContentView: View {
    var body: some View {
        ZStack {
            Color(red: 1, green: 0.8, blue: 0)
            Text("Your content")
        }
    }
}
```

- **Go outside the safe area**
  .edgesIgnoringSafeArea(.all)

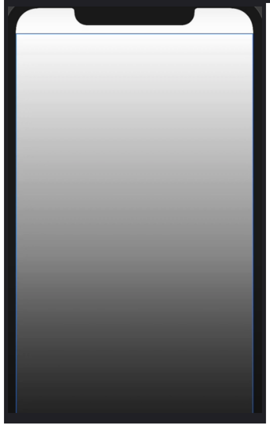


```
//  Created by Paul Hudson on 12/10/2019.
//  Copyright © 2019 Hacking with Swift. All rights
   reserved.
//

import SwiftUI

struct ContentView: View {
    var body: some View {
        ZStack {
            Color.red.edgesIgnoringSafeArea(.all)
            Text("Your content")
        }
    }
}
```

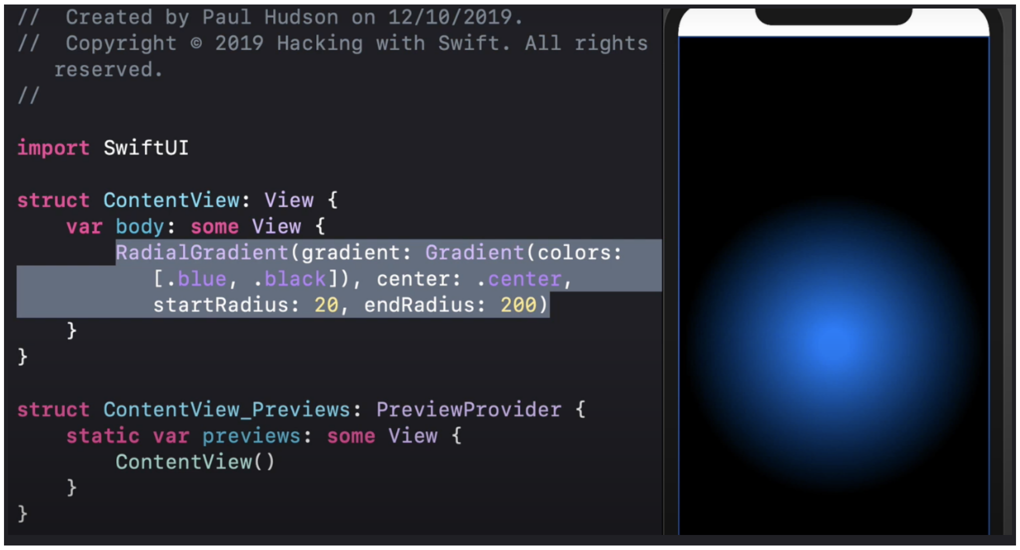
**3/. Gradient**

- **Linear Gradient**

```
LinearGradient(gradient: Gradient(colors: [.white,.black]),
                  startPoint: .top,
                  endPoint: .bottom)
```

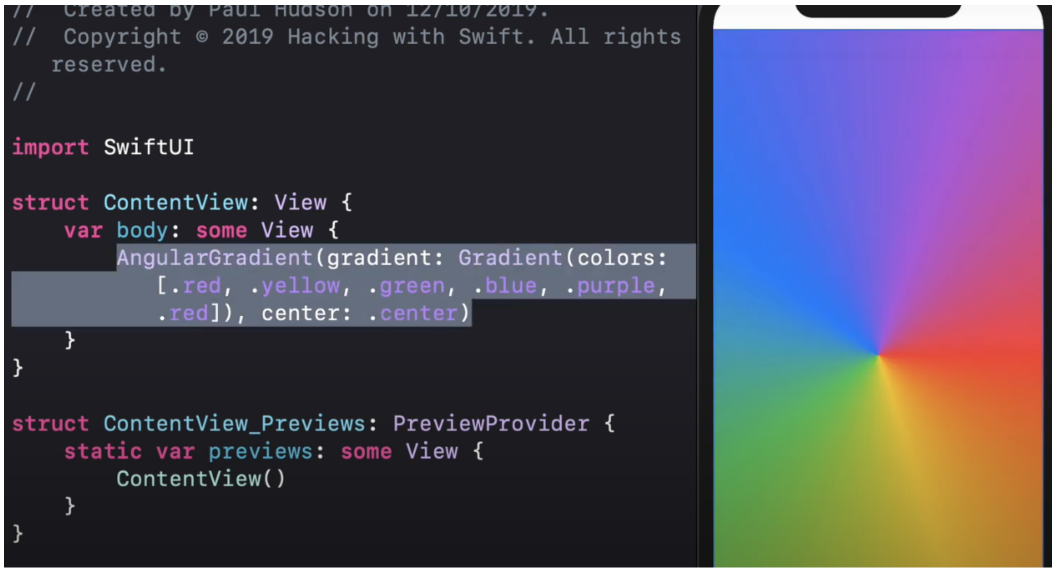


- **RadialGradient**



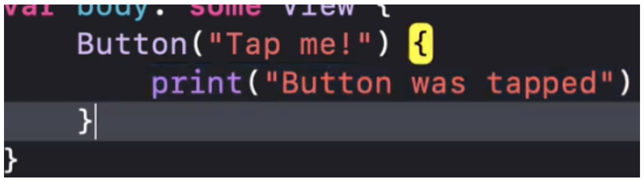
```swift
//  Created by Paul Hudson on 12/10/2019.
//  Copyright © 2019 Hacking with Swift. All rights
   reserved.
//

import SwiftUI

struct ContentView: View {
    var body: some View {
        RadialGradient(gradient: Gradient(colors:
           [.blue, .black]), center: .center,
           startRadius: 20, endRadius: 200)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

- **AngularGradient (Conic gradient)**

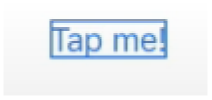


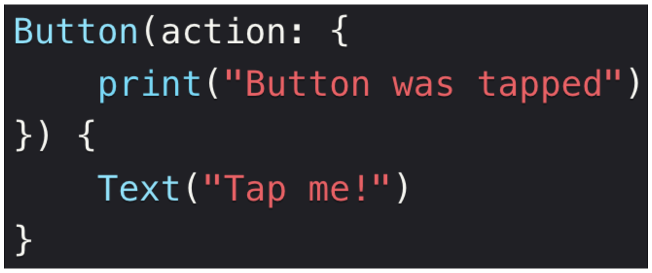
```swift
//  Created by Paul Hudson on 12/10/2019.
//  Copyright © 2019 Hacking with Swift. All rights
   reserved.
//

import SwiftUI

struct ContentView: View {
    var body: some View {
        AngularGradient(gradient: Gradient(colors:
           [.red, .yellow, .green, .blue, .purple,
           .red]), center: .center)
    }
}

struct ContentView_Previews: PreviewProvider {
    static var previews: some View {
        ContentView()
    }
}
```

**4/. Button**



```swift
var body: some View {
    Button("Tap me!") {
        print("Button was tapped")
    }
}
```

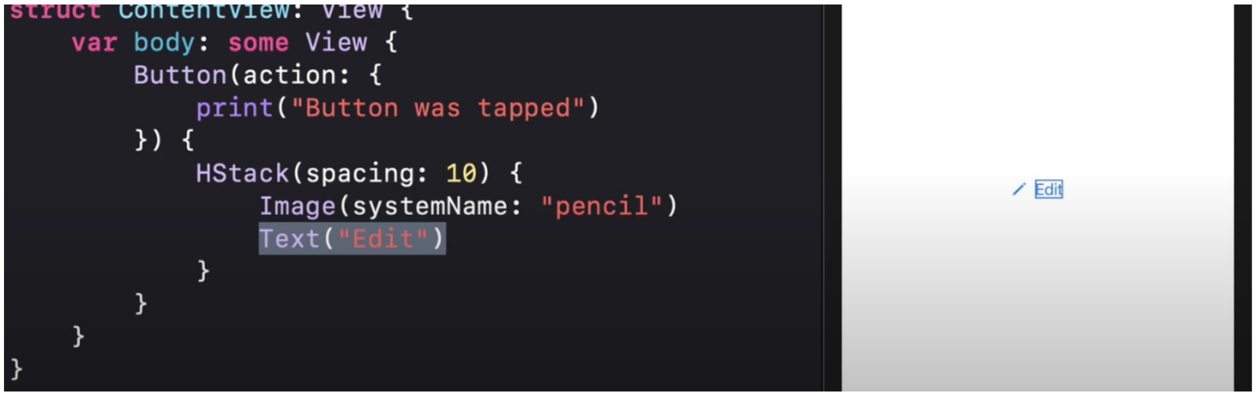




```
Button(action: {
    print("Button was tapped")
}) {
    Text("Tap me!")
}
```

**<<< img / combination of views**

- **Image:** for handling pictures in the app
    - **Image("pencil")** // load a "pencil" image
    - **Image(decorative: "pencil")** // load the same image as above, but will not read "pencil" if users use the screen reader
    - **Image(systemName: "pencil")** // load a pencil icon built into iOS app

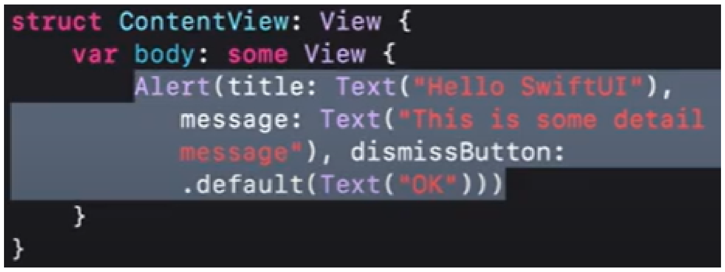


```
struct ContentView: View {
    var body: some View {
        Button(action: {
            print("Button was tapped")
        }) {
            HStack(spacing: 10) {
                Image(systemName: "pencil")
                Text("Edit")
            }
        }
    }
}
```

- Note: use **renderingMode(.original)**
    - Force SwiftUI to show original images (i.e. non-color pencil) instead of the recolored images (i.e. blue-colored pencil)

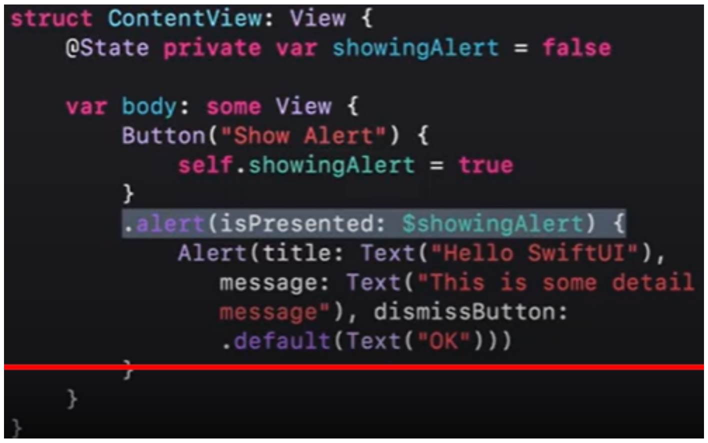
**5/. Alert**:

- Basics:
    - Title
    - Message
    - Dismiss button



```
struct ContentView: View {
    var body: some View {
        Alert(title: Text("Hello SwiftUI"),
            message: Text("This is some detail
            message"), dismissButton:
            .default(Text("OK")))
    }
}
```

- When to show alert:



```
struct ContentView: View {
    @State private var showingAlert = false

    var body: some View {
        Button("Show Alert") {
            self.showingAlert = true
        }
        .alert(isPresented: $showingAlert) {
            Alert(title: Text("Hello SwiftUI"),
               message: Text("This is some detail
               message"), dismissButton:
               .default(Text("OK")))
        }
    }
}
```

**6/. Pick a random number**: var x = Int.random(in: 0...10) // random num in [0, 10]

**7/. Adjust the flag image:**

- .clipShape(shape())
    - .clipShape(Rectangle())
    - Rounded rectangle
    - Circle
    - Capsule
- Draw boundary around the image
    - .overlay(Capsule().stroke(Color.black, lineWidth: 1))
- Add shadow around the image
    - .shadow(color: .black, radius: 1)