Day 16:

ContentView is the struct that gets recreated by default in a new SwiftUI application. ContentView conforms to View Protocol. View protocol required a body parameter that is of type ***some View***

Any element can contain a maximum upto 10 children. For example, a Form view can hold 10 children elements. Due to this limitation, we can wrap the children into Group/Section view in order to hold more data.

ContentView_Previews is used to create a Preview on the canvas and is not part of the final xcode build.

NavigationView can be used to add a navigation bar.

```
struct ContentView: View {
    var body: some View {
        NavigationView {
            Form {
                Section {
                    Text("Hello, world!")
                }
            }
            .navigationTitle("SwiftUI")
            .navigationBarTitleDisplayMode(.inline)
        }
    }
}

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

State variables can be used inside the code by mentioning the @State keyword. This is because the ContentView is of type Struct and hence the variables cannot be updated unless they are marked with @State. This make Swift UI store these state variables in a different location.

```
struct ContentView: View {
    @State private var tapCount = 0

    var body: some View {
        Button("Tap Count: \(tapCount)") {
            tapCount += 1
        }
    }
}
```

We can similarly use state variables for textfields to store the value that gets entered into the field. We add $ before the variable name denoting that two way binding is in place i.e, value is updated and also returned

```
struct ContentView: View {
    @State private var name = ""

    var body: some View {
        Form {
            TextField("Enter your name", text: $name)
            Text("Your name is \(name)")
        }

    }
}
```