SWIFTUI cheat sheet

Adriana Belinski

Basic Views

VStack stacks views vertically, top to bottom

```
VStack {
   Text("Welcome")
   Image(systemName: "star.fill")
   Text("to SwiftUI!")
}
```

HStack stacks views horizontally, left to right

```
HStack {
   Text("Hello, world!"))
   Image(systemName: "star.fill")
      .foregroundColor(.yellow)
   Text("This is a VStack.")
}
```

ZStack stacks views "on top of eachother" on z-axis

```
ZStack {
   Color.purple first in the code & will be beneath the next two views
      .ignoresSafeArea() // Fill the entire screen
   Image(systemName: "star.fill") second
      .resizable()
      .frame(width: 200, height: 200)
      .foregroundColor(.yellow)
      .position(x: 150, y: 200) // Position the star
   Text("Welcome!") third & lies on top of the stack
      .font(.largeTitle)
      .foregroundColor(.white)
}
```

Text

```
Text("Hello, world!")
```

Image

```
Image("logo_1")
```

Button

```
Button("Tap me") {
   print("Tapped!")
}
```

NavigationStack

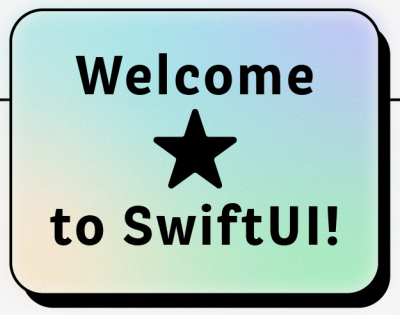
```
struct ContentView: View {
   var body: some View {
      NavigationStack {
         List {
            Text("Item 1")
            NavigationLink(destination: DetailView()) {
               Text("Item 2")
            }
         }
         .navigationTitle("Main List")
      }
   }
}

struct DetailView: View {
    var body: some View {
        Text("Detailed information about Item 2")
    }
}
```

TabView

```
struct ContentView: View {
   var body: some View {
      TabView {
         PageOne()
            .tabItem {
               Label(
                  "Category 1",
                  systemImage: "list.dash")
            }

         PageTwo()
            .tabItem {
               Label(
                  "Category 2",
                  systemImage: "heart.fill")
            }
      }
   }
}
```



View Modifiers

.frame

```
Text("Hello, world!")
   .frame(width: 200, height: 50) fixed size
   .frame(maxWidth: .infinity) full width
```

.font

```
Text("Hello, world!")
   .font(.largeTitle) or
   .font(.system(
      size: 24,
      weight: .bold,
      design: .serif))
```

.opacity

```
Image("logo")
   .opacity(0.5) half transparent
```

.background

```
Text("Hello")
   .background(Color.blue) or
   .background(Image("pattern"))
```

.rotationEffect

```
Image(systemName: "arrow.right")
   .rotationEffect(.degrees(45))
```

.blur

```
Image("background")
   .blur(radius: 5)
```

.scaleEffect

```
Button("Tap me")
   .scaleEffect(1.2) slightly larger
```

.padding

```
Text("Important message")
   .padding(20) all sides
   .padding(.horizontal, 40) horizontal only
```

.foregroundColor

```
Text("Important message")
   .foregroundColor(.red) or
   .foregroundColor(Color(
         red: 0.5,
         green: 0.8,
         blue: 1.0))
```

.tint

```
Button("Tap me")
   .tint(.orange) tints the button's text,
                  image, or other elements
```

Trailing Closure Syntax

Example of TCS in a modifier

```
view
   .modifierName { closure arguments in
      // Closure body
}
```

Example of TCS in a for-each view

```
ForEach(items) { item in
   Text(item.name)
}
```

Data Binding

@State

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

   var body: some View {
      TextField("Enter your name", text: $name)
         .padding()
      Text("Hello, \(name)!")
   }
}
```

ObservedObject

```
class UserSettings: ObservableObject {
   @Published var username = "defaultUser"
}

struct ProfileView: View {
   @ObservedObject var userSettings = UserSettings()

   var body: some View {
      Text("Username: \(userSettings.username)")
      TextField(
         "Update username",
         text: $userSettings.username)
   }
}
```

@Binding

```
struct SettingsView: View {
   @Binding var brightness: Double

   var body: some View {
      Slider(value: $brightness, in: 0 ... 1)
   }
}

struct ContentView: View {
   @State private var brightness = 0.5

   var body: some View {
      Text("Brightness: \(brightness)")
      SettingsView(brightness: $brightness)
   }
}
```