Controls cheat sheet

In this lesson you will explore different kinds of controls that are available to you in SwiftUI and how to build them. SwiftUI provides many ways for the user to interact with the layout of a user interface. The most basic means of doing so is through controls. Controls form the building blocks for interactions with users and become more complex as the requirements of your application increase. Some of the most commonly used controls include `Button`, `Toggle`, `Stepper`, `Slider`, `ProgressView`, `DatePicker` and `Label`.

When writing a control you must keep in mind that the control should enable some sort of specific user interaction needed by your app. Each control is composed of a UI and a programmatic setup which allows for further customization of behavior down the line for your app. Keep in mind that each respective control has methods to enable customization, such as coloring, but this is beyond the scope of this course.

Button

Requires a closure to execute actions and needs its appearance specified.

```
struct ContentView: View {

    var body: some View {

        Button(role: .destructive) {
            print("do something!")
        } label: {
            HStack {
                Image(systemName: "trash")
                Text("")
            }
        }
    }
}
```

Toggle

Toggles help switch between states of true and false.

```
struct ContentView: View {

    @State var isShowing = true // toggle state – A toggle won't work unless you've give a truth stat

    var body: some View {

        Toggle(isOn: $isShowing) {
            Text("Hello World")
        }
    }
}
```

Steppers

Steppers require some sort of state like a toggle. Because steppers work in increments, rather than assigning a `bool`, you give an `int`. The output? A method for the user to increment or decrement a value, such as selecting the number of guests to reserve a table for.

```
struct ContentVIew: View {

@State private var value = 1

var body: some View {
        NavigationView {
            VStack {
                Text("Current value: \(value)")
                Stepper("Number of guests", value: $value, in:1...20)
            }
            .padding()
        .navigationTitle("Reservation form")
        }
    }
}
```

Sliders

The usual interaction sliders are responsible for are selecting between a range of numbers

```
struct ContentView: View {

    @State private var size: CGFloat = 0.1
```

```
    @State private var size: CGFloat = 0.1

    var body: some View {
        VStack {
            Text("Little Lemon").font(.system(size: size * 50))
            Slider(value: $size)

        }
        .padding()
    }
}
```

DatePicker

If your app requires selecting a date, for example, choosing a date on a calendar, `DatePicker` is a useful control. Under `Form` or `List`, it appears as a single list row.

```
struct ContentView: View {

    @State var selectedDate = Date()
    var dateClosedRange: ClosedRange<Date> {
        let min = Calendar.current.date(byAdding: .day, value: -1, to: Date())!
        let max = Calendar.current.date(byAdding: .day, value: 1, to: Date())!
        return min...max
    }

    var body: some View {


        NavigationView {
            Form {
                Section {
                    DatePicker(
                        selection: $selectedDate,
                        in: dateClosedRange,
                        displayedComponents: .date,
                        label: { Text("Due Date") }
                    )
                }
            }
        }
    }
}
```

Label

Label is a convenient view that presents an image and text alongside each other. This is suitable for a menu item or your settings.

```
struct SwiftUIView: View {

    var body: some View {

        Label("Menu", systemImage: "fork.knife")
            .labelStyle(.titleAndIcon)
    }
}
```

Conclusion

In this lesson you learned how to build basic controls including buttons, toggles, steppers, sliders, DatePickers, and a label through provided code samples for different controls.

Mark as completed

Like Dislike Report an issue