Building Apps with SwiftUI
CMSC 436
Decluttering the UI with Tabs

```swift
var body: some View {
    TabView {
        HStack {
            Text("max:").frame(width: 50, height: 20) // .padding()
            TextField("", text: $mvText) {
                _ in if let i = Int(mvText) { vals.setMax(i) }
            }
            .frame(width: 50, height: 20, alignment: .center)
            .textFieldStyle(RoundedBorderTextFieldStyle())
        }.tabItem { Text("Set Length") }
        MyScroll(vals.maxVal).padding().tabItem { Text("View List") }
    }
}
```

Embed tabs in a TabView

tabItem() method lets us provide a label
Tabbed Version of the UI
struct ButtonView: View {
    @State private var buttonText = "Push me!"

    var body: some View {
        Button(action: f) {
            Text(buttonText)
        }
    }

    func f() {
        buttonText = "You pushed the button!"
    }
}
Testing `ButtonView`

1. Push me!
   - Displays "You pushed the button!"
struct AlertingButtonView: View {
    @State private var showAlert = false

    var body: some View {
        Button("Push me!") {
            showAlert = true
        }.alert(isPresented: $showAlert) {
            Alert(  
                title: Text("This is an alert!"),
                message: Text("You pushed the button, didn't you?")
            )
        }
    }
}

All Views have an alert() method

Alert based on app state (errors, scheduled announcements, etc.)
Testing AlertingButtonView

Push me!

This is an alert!
You pushed the button, didn’t you?
OK
struct CustomAlertingButtonView: View {
    @State private var showAlert = false
    @State private var buttonText = "Push me!"

    var body: some View {
        Button(buttonText) {
            showAlert = true
        }.alert(isPresented: $showAlert) {
            Alert(
                title: Text("This is an alert!")
                message: Text("You pushed the button, didn't you?")
                primaryButton: .default(Text("Foo"),
                    action: {
                        buttonText = "Foo"
                    } ),
                secondaryButton: .destructive(Text("Bar"),
                    action: {
                        buttonText = "Bar"
                    } )
            )
        }
    }
}
Testing CustomAlertingButtonView

This is an alert!
You pushed the button, didn't you?

Foo    Bar
Action Sheets

Modal (like Alerts)
Based on user actions

struct ActionSheetView: View {
    @State private var showActionSheet = false
    @State private var buttonText = "Push me!"

    var body: some View {
        Button(buttonText) {
            showActionSheet = true
        }.actionSheet(isPresented: $showActionSheet) {
            ActionSheet(
                title: Text("This is an action sheet"),
                message: Text("You pushed the button, didn't you?"),
                buttons: [
                    .cancel(),
                    .default(Text("Foo"),
                        action: { buttonText = "Foo" } ),
                    .destructive(Text("Bar"),
                        action: { buttonText = "Bar" } )
                ]
            )
        }
    }
}
Testing ActionSheetView

- Appears at bottom
- System might re-order the buttons
- As many buttons as you need
- Button actions can be methods, functions, or closures (like with Alerts and Buttons)
Toggles

Sometimes we need a boolean

```swift
struct ToggleView: View {
    @State private var selected = false
    private var yes = "It is!"
    private var no = "It is not!"

    var body: some View {
        VStack{
            Toggle(isOn: $selected) {
                Text("Is it selected?")
            }.padding()
            Text("\(selected ? yes : no )")
        }
    }
}
```
Testing ToggleView

Is it selected?

It is not!

Is it selected?

It is!
This shows two of the ways of creating a Menu.
Testing MenuView

Note that the order of the options is reversed!
Gestures

The stuff you’re used to:

- **Tap** (TapGesture)
- **Long press** (LongPressGesture)
- **Drag** (DragGesture)
- **Magnify** (MagnificationGesture)
- **Rotate** (RotationGesture)

All of these have:

- `onEnded()`
- `updating()`

Most of them (excluding Tap) have:

- `onChanged()`
Tap Gestures

Anything can be a button!

\[ \text{init()} \text{ takes an optional count} \]
\[ \Rightarrow \text{You have to tap this many times to activate it} \]

\[ \text{onEnded()} \text{ takes a closure } () \rightarrow \text{Void} \]
\[ \Rightarrow \text{This is executed when activated} \]