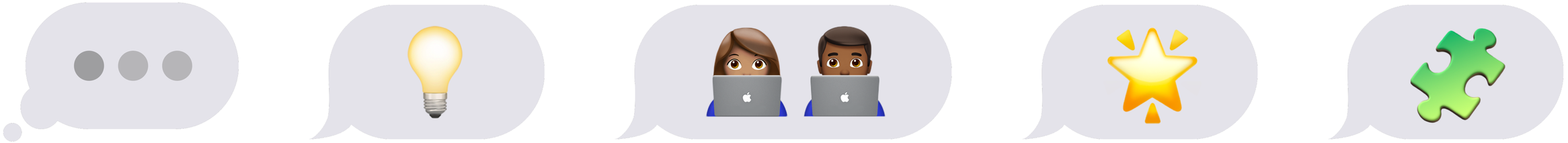
移动平台开发

实验课三：SwiftUI进阶



2025春夏学期

SwiftUI中的交互的视图组建

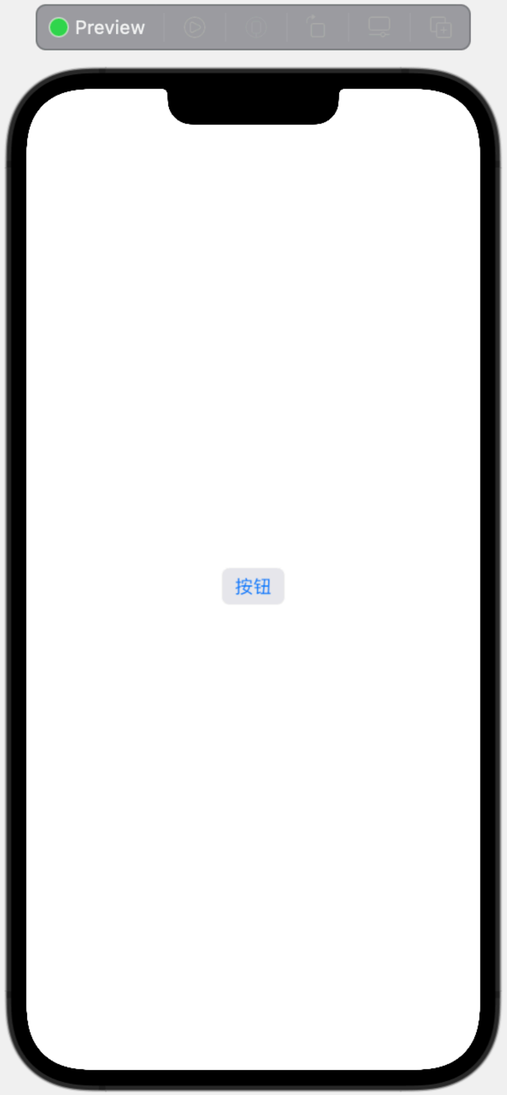
Interacting elements in SwiftUI

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Button("按钮", action: {})
    }
}
```

SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

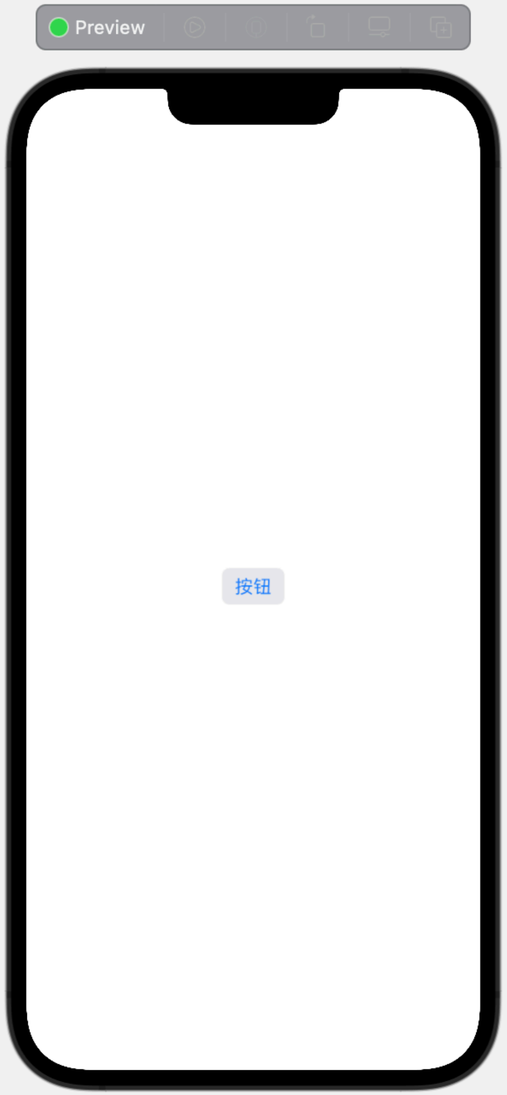
```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Button("按钮", action: {})
            .buttonStyle(.bordered)
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

```
import SwiftUI

struct ContentView: View {
    var pressed = false

    var body: some View {
        Button("按钮", action: {})
                    .buttonStyle(.bordered)

    }
}
```

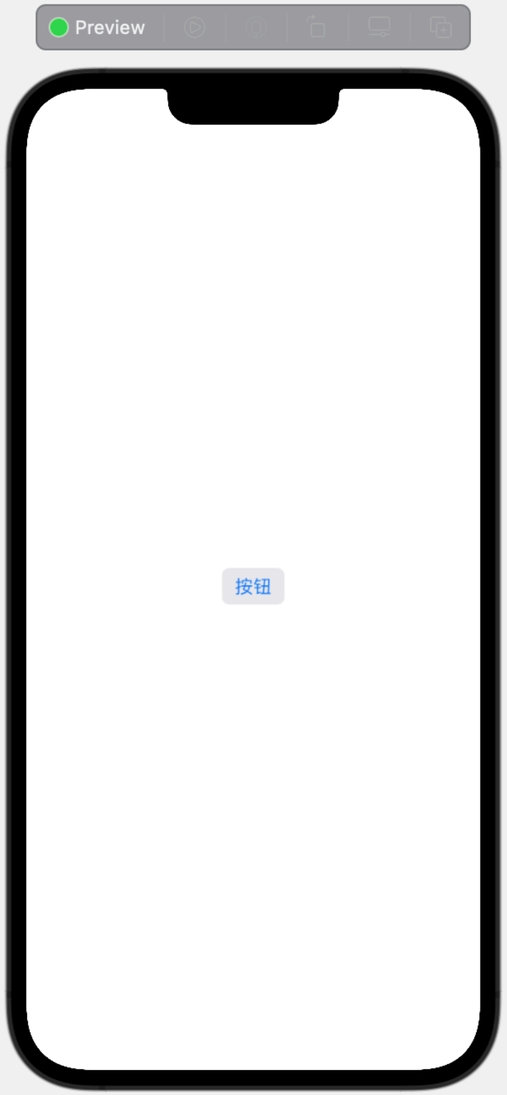


SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

```
import SwiftUI

struct ContentView: View {
    var pressed = false

    var body: some View {
        Button("按钮", action: {})
                .buttonStyle(.bordered)
    }
}
```

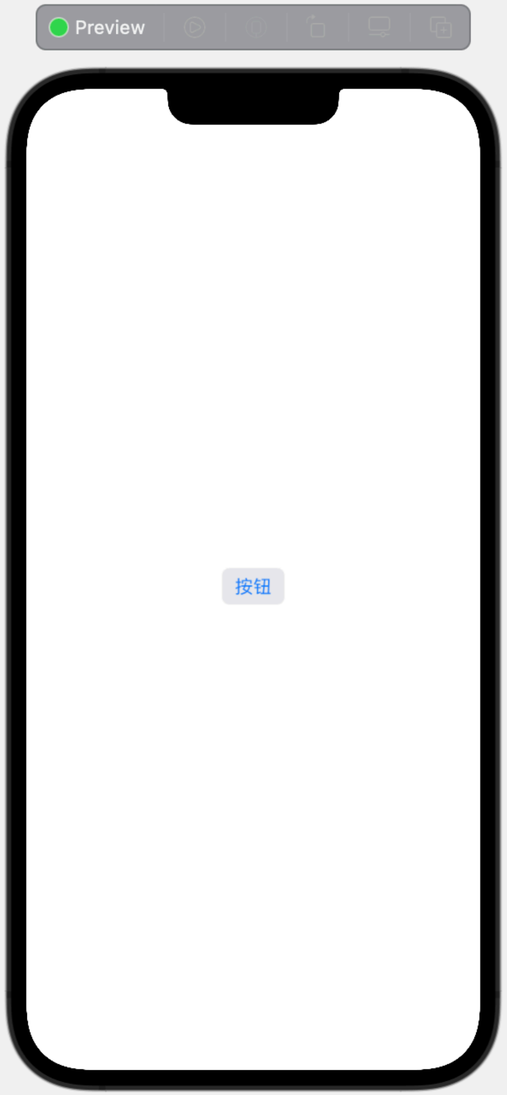
Cannot assign to property: 'self' is immutable



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

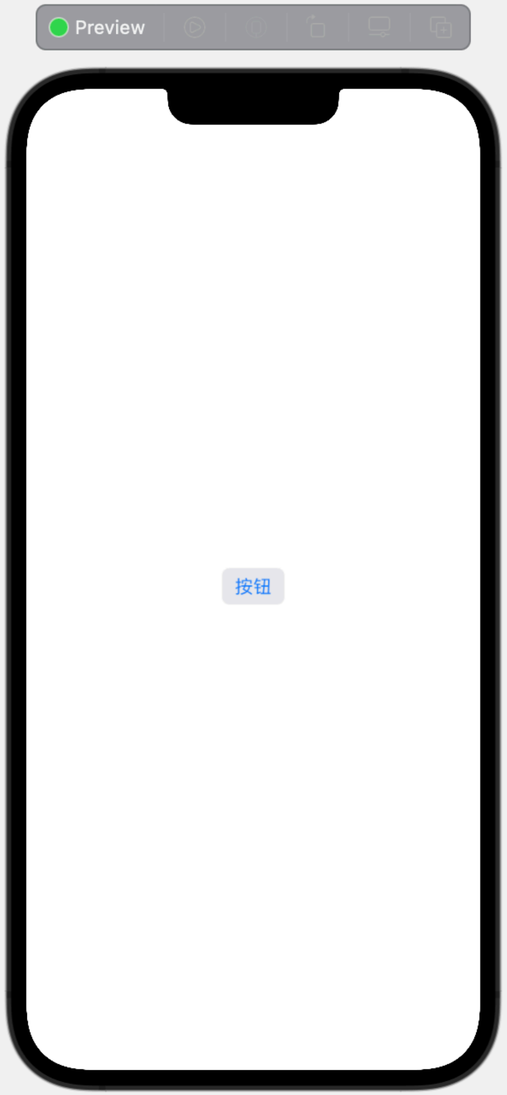
```
import SwiftUI

struct ContentView: View {
  @State var pressed = false
    var body: some View {
          Button("按钮", action: {
              pressed = true
          })
            .buttonStyle(.bordered)
      }
  }
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

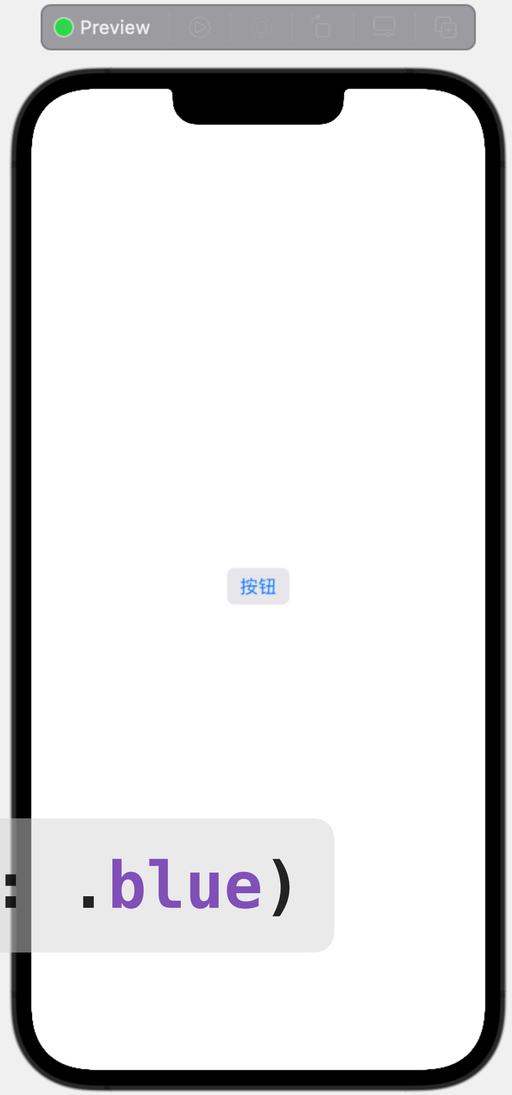
```
import SwiftUI

struct ContentView: View {
    @State var pressed = false
    var body: some View {
        Button("按钮", action: {
            pressed.toggle()
        })
        .buttonStyle(.bordered)
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

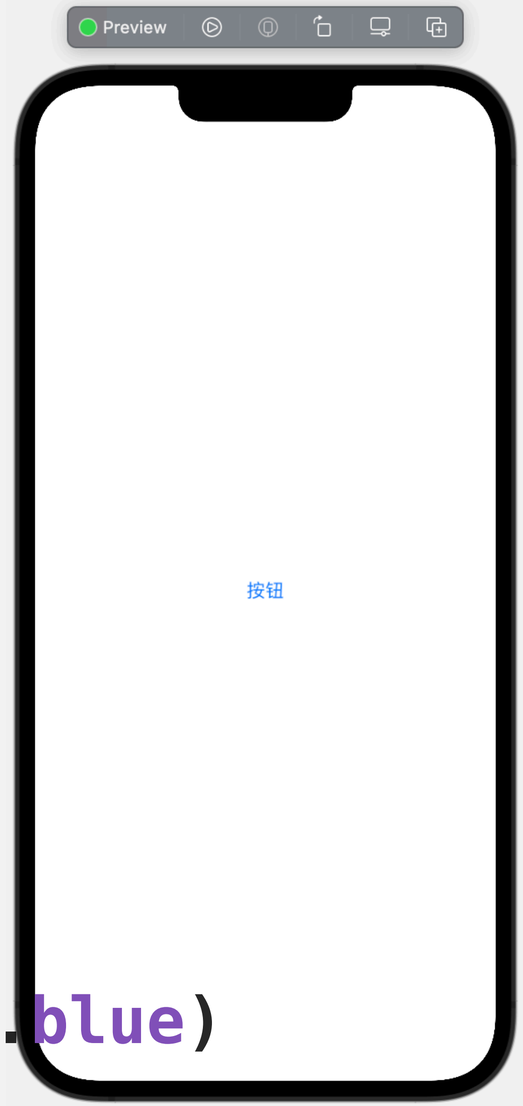
```
import SwiftUI

struct ContentView: View {
    @State var pressed = false
    var body: some View {
        Button("按钮", action: {
            pressed.toggle()
        })
        .foregroundColor(pressed ? .green : .blue)
        .buttonStyle(.bordered)
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

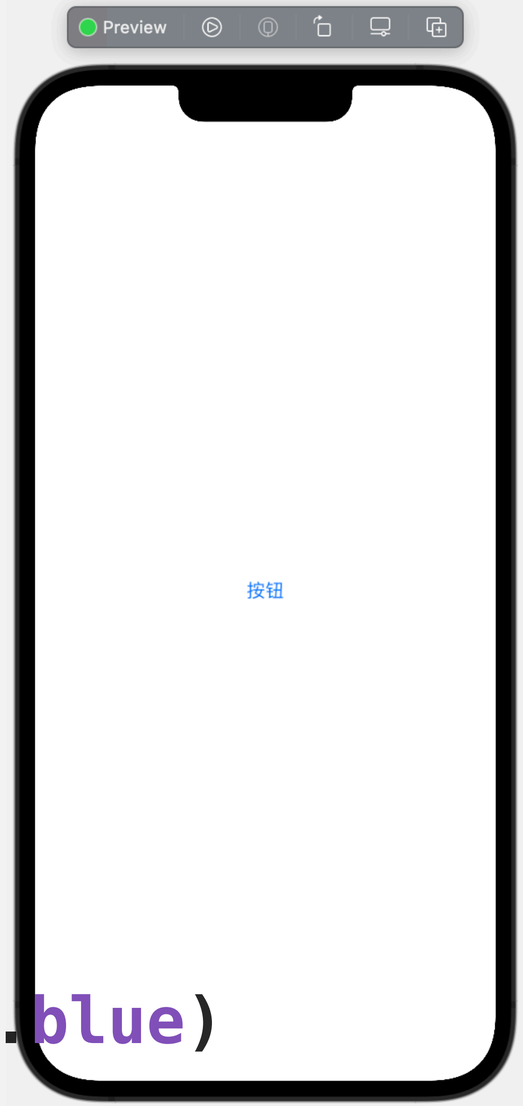
```
import SwiftUI
struct ContentView: View {

    @State var pressed = false
    var body: some View {
        Button(action: {
            pressed.toggle()
        },label: {
            Text("按钮")
        })
        .foregroundColor(pressed ? .green : .blue)
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

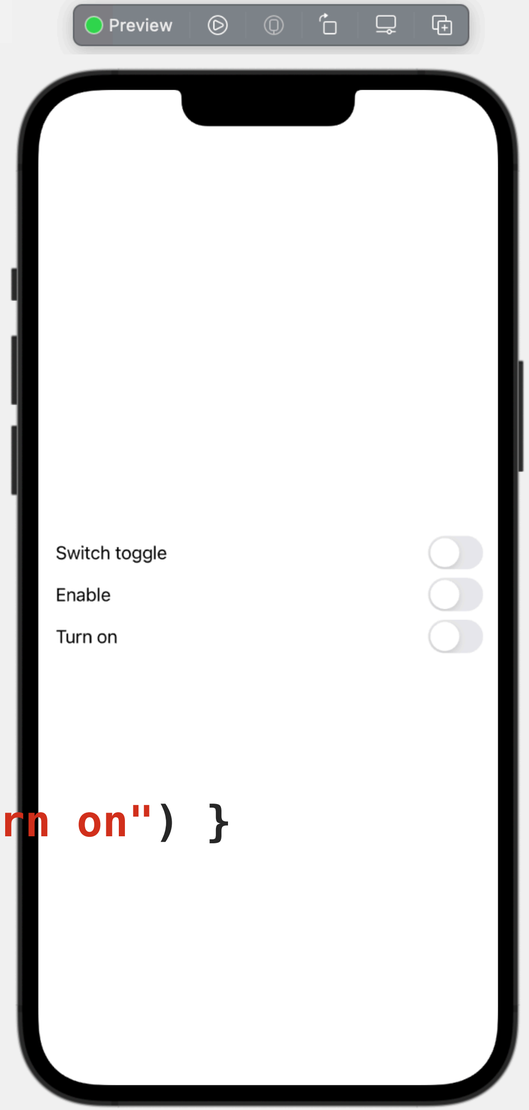
```
import SwiftUI
struct ContentView: View {

    @State var pressed = false
    var body: some View {
        Button(action: {
            pressed.toggle()
        },label: {
            HStack{ Text(...) Image(...) }
        })
        .foregroundColor(pressed ? .green : .blue)
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

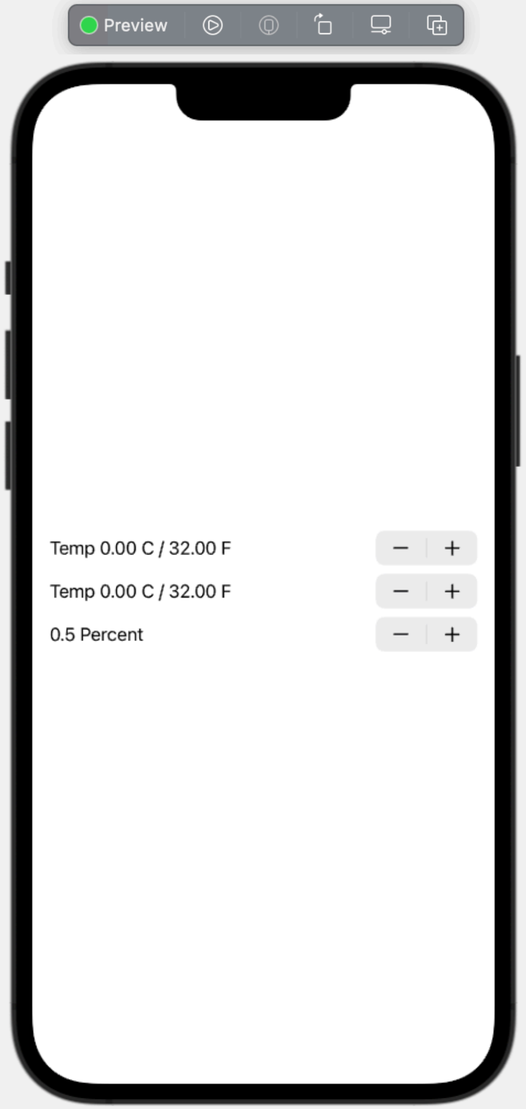
```
import SwiftUI
struct ContentView: View {

    @State private var flag1 = false
    @State private var flag2 = false
    @State private var flag3 = false

    var body: some View {
        VStack {
            Toggle("Switch toggle", isOn: $flag1)

            Toggle(isOn: $flag2, label: { Text("Enable") })

            Toggle(isOn: $flag3){ Text(flag3 ? "Turn off" : "Turn on") }
        }
        .padding()
    }

}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

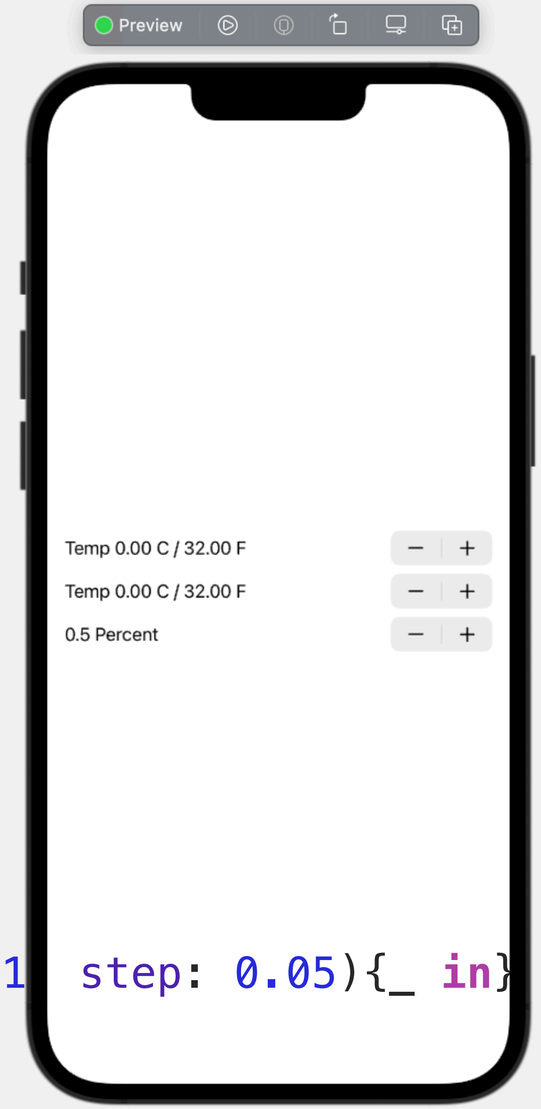
```
@State private var c: Double = 0
@State private var f: Double = 32
@State var value: Double = 0.5

var body: some View {
    VStack {
        Stepper {
            Text("Temp \(formatVal(c)) C / \(formatVal(f)) F")
        } onIncrement: {
            self.c += 1
            self.f = self.c * (9/5) + 32
        } onDecrement: {
            self.c -= 1
            self.f = self.c * (9/5) + 32
        }
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

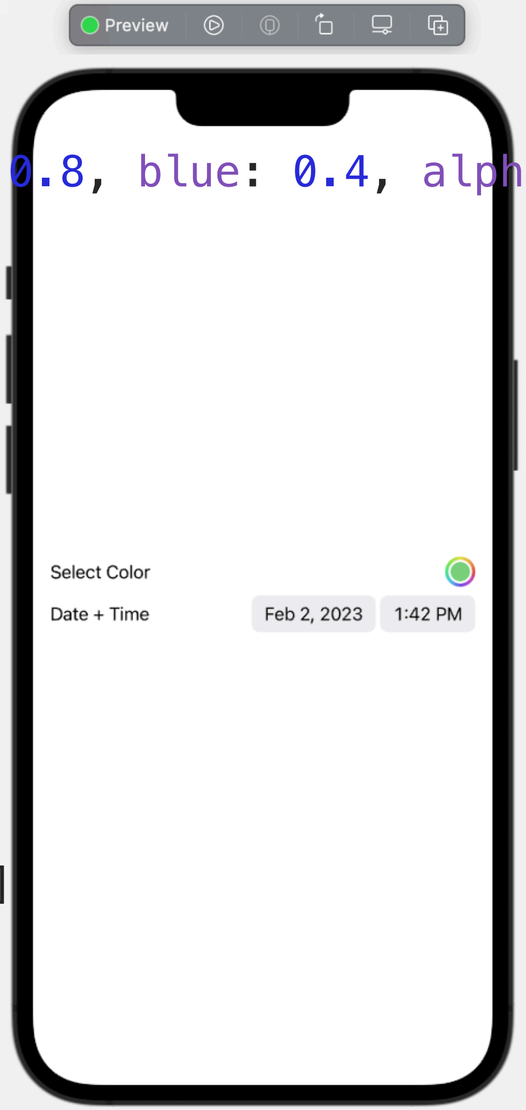
```
@State private var c: Double = 0
@State private var f: Double = 32
@State var value: Double = 0.5

Stepper(onIncrement: {
     self.c += 1
     self.f = self.c * (9/5) + 32
 }, onDecrement: {
     self.c -= 1
     self.f = self.c * (9/5) + 32
 }, onEditingChanged: {
     print("\($0)")
 }, label: {
     Text("Temp \(formatVal(c)) C / \(formatVal(f)) F")
 })

 Stepper("\(value) Percent" as String, value: $value, in: 0...1  step: 0.05){_ in}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

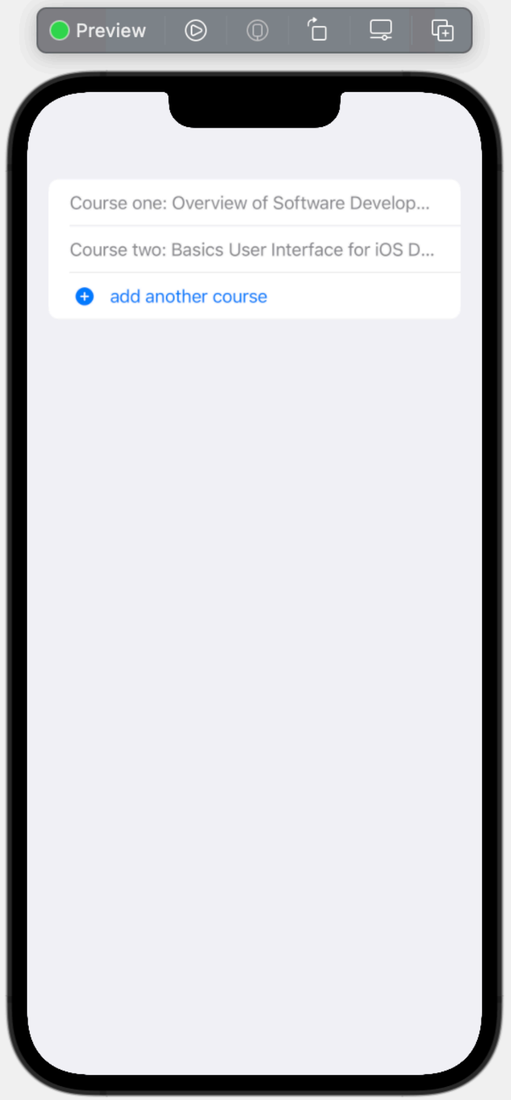
```
@State private var cgColor: CGColor = CGColor(red: 0.4, green: 0.8, blue: 0.4, alpha: 1.0)
@State private var selectedDate = Date()
let colors: [Color] = [.green, .yellow, .orange, .red]

let title = "Select Color"

var body: some View{

    VStack {
        ColorPicker(title, selection: $cgColor,
                    supportsOpacity: true)

        DatePicker("Date + Time",
                   selection: self.$selectedDate,
                   displayedComponents: [.hourAndMinute, .date])
            // .datePickerStyle(WheelDatePickerStyle())
    }
    .padding()
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

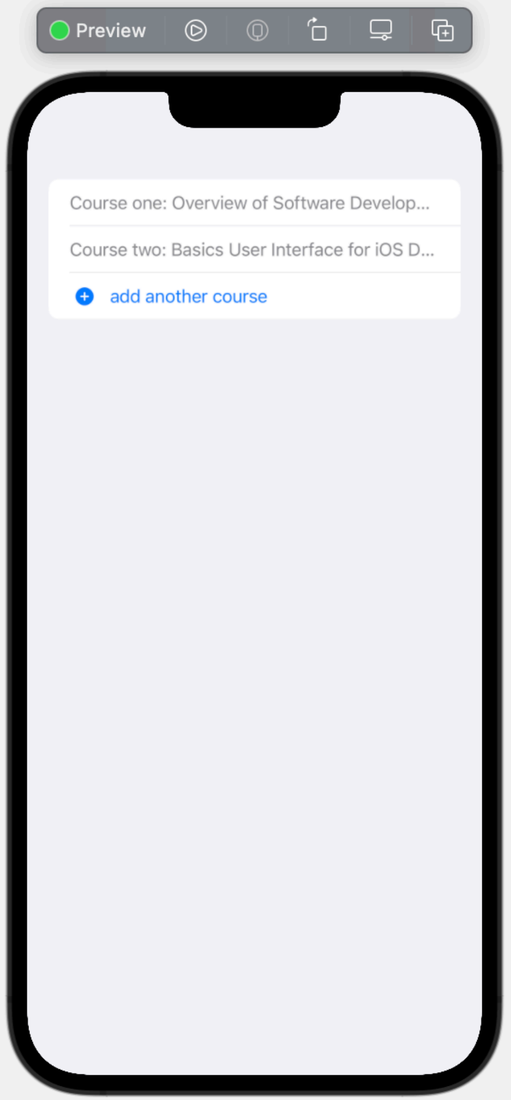
```
import SwiftUI
struct ContentView: View {
    var body: some View {
        List{
            Text(...)
            Text(...)
            Button(...)
        }
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

```
import SwiftUI
struct ContentView: View {
    var body: some View {
        Form{
            Text(...)
            Text(...)
            Button(...)
        }
    }
}
```

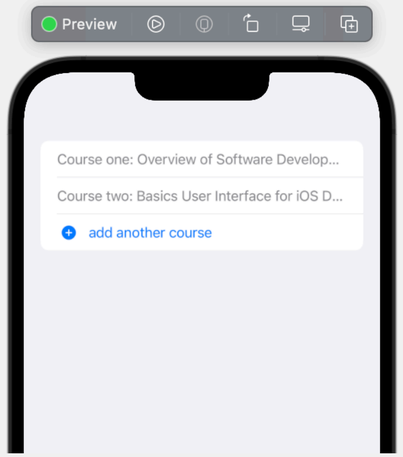


SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

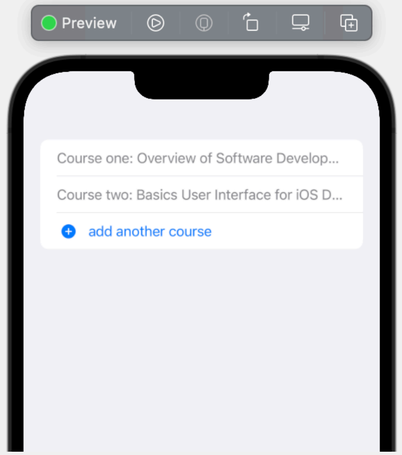
浙江大学 ZHEJIANG UNIVERSITY

List



(Default)

Form



(Default)

SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

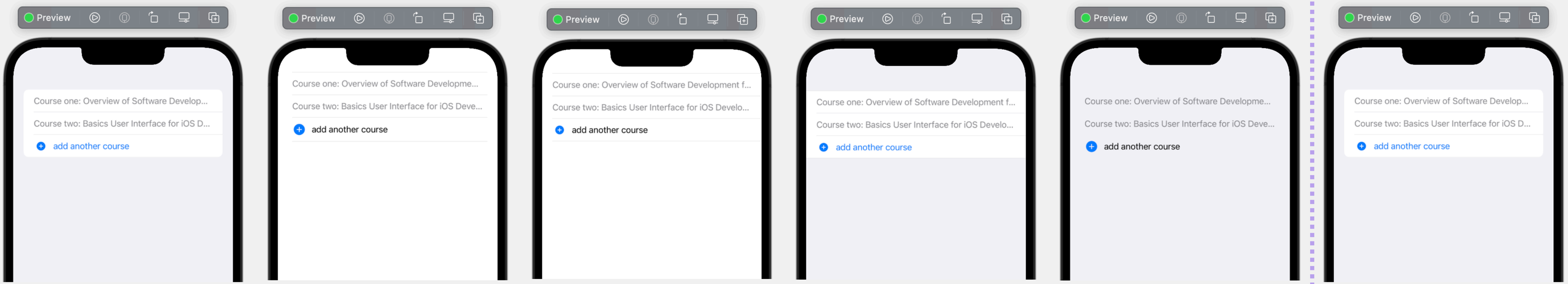
浙江大学 ZHEJIANG UNIVERSITY

List

.listStyle(.inset)

.listStyle(.grouped)

Form



(Default)

.listStyle(.plain)

.listStyle(.sidebar)

(Default)

SwiftUI中的交互的视图组建

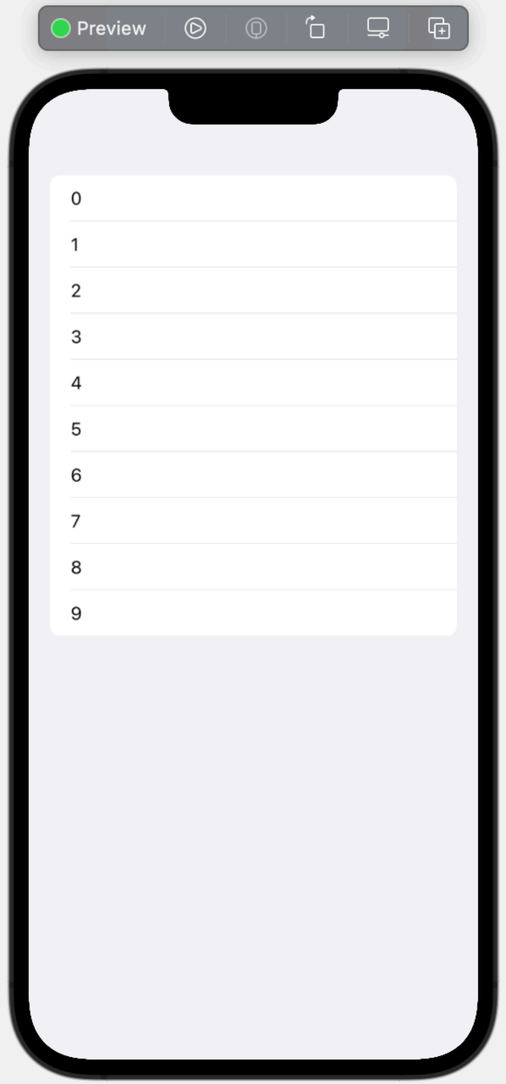
Interacting elements in SwiftUI

```
import SwiftUI
struct ContentView: View {
    var body: some View {
        List{
          Text("1")
             . . .
          Text(“10")
        }
    }
}
```

SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

```
import SwiftUI
struct ContentView: View {

    var body: some View {
        List{
            ForEach(0...9 , id: \.self){ i in
                Text("\(i)")
            }

        }
    }
}
```



SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

```
import SwiftUI
struct ContentView: View {
    var body: some View {
        List{
            Section{
                Text("1")
                Text("10")
            }
        }
    }
}
```

SwiftUI中的交互的视图组建

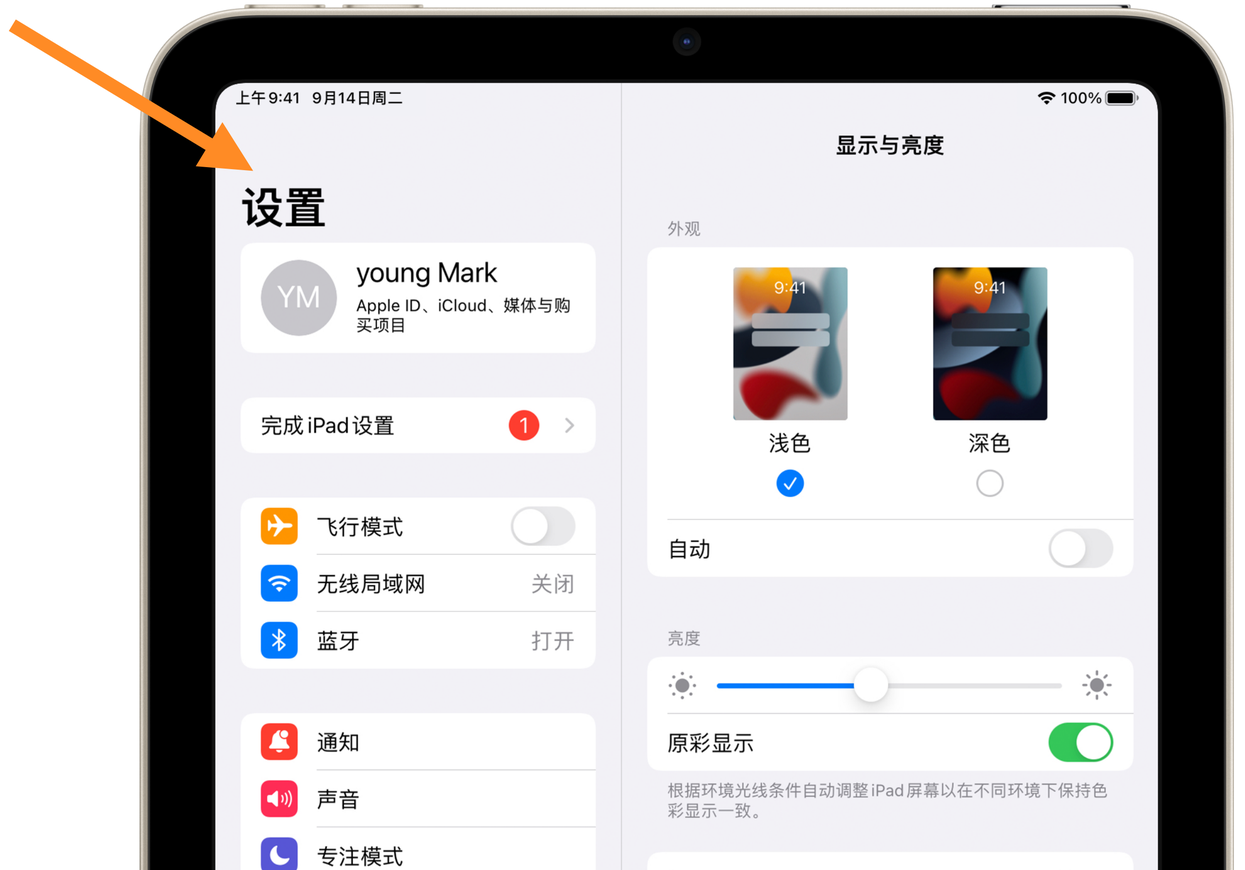
Interacting elements in SwiftUI

```
import SwiftUI
struct ContentView: View {

    var body: some View {
        List{
            Section(header: Text("!!")) {
                Text("1")
                    .
                    .
                    .
                Text("10")
            }
        }
    }
}
```

SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

浙江大学 ZHEJIANG UNIVERSITY



SwiftUI中的交互的视图组建

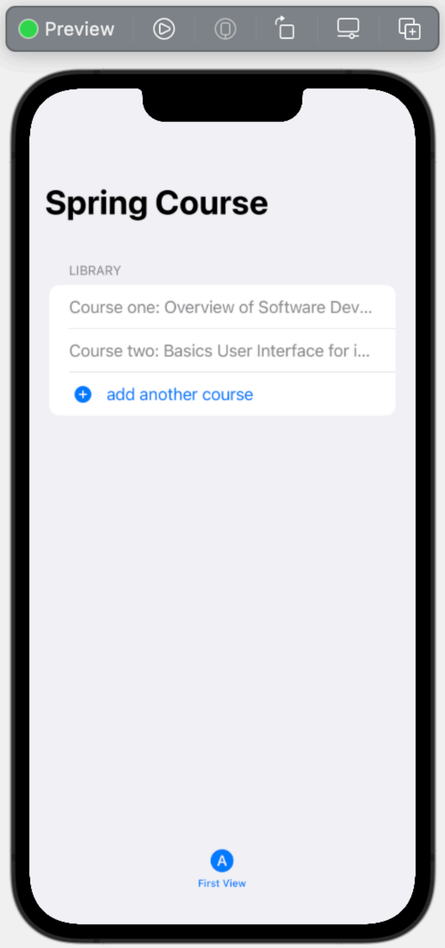
Interacting elements in SwiftUI

```
import SwiftUI
struct ContentView: View {
    var body: some View {
        NavigationView{
            List{
                . . .
            }
        .navigationTitle("设置")
        }
    }
}
```

SwiftUI中的交互的视图组建

Interacting elements in SwiftUI

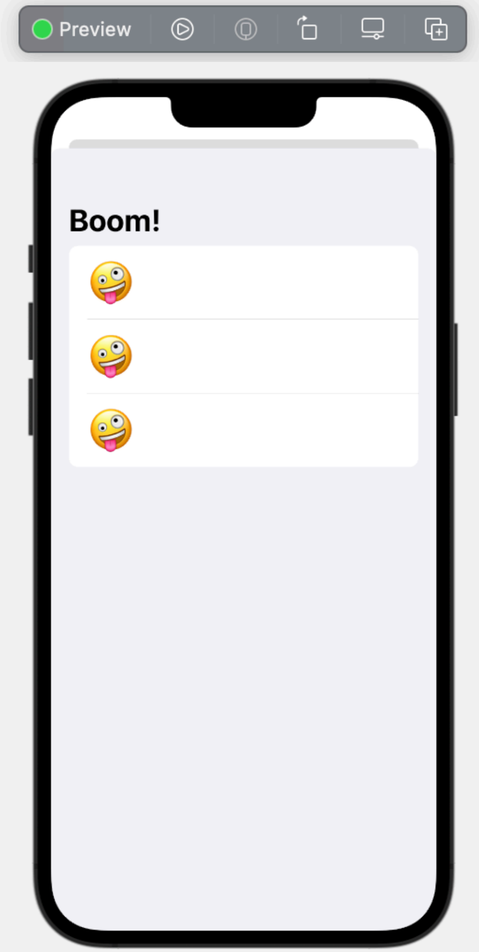
```
import SwiftUI
struct ContentView: View {
    var body: some View {

        TabView{

            homePage
                .tabItem {
                    Label("First View",
                          systemImage: "a.circle")
                }
        }

    }
```



SwiftUI中的交互的视图组建

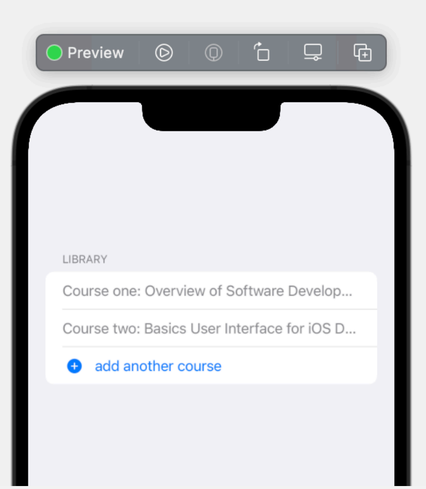
Interacting elements in SwiftUI

```
import SwiftUI

struct ContentView: View {
    @State var pressed = false
    var body: some View{
        Button("Trick or Treat", action: {pressed.toggle()})
            .buttonStyle(.bordered)
            .sheet(isPresented: $pressed){
                MySheet
            }
            .sheet(isPresented: $pressed,
                   onDismiss: { print("finished!") },
                   content: {
                    MySheet
                    .presentationDetents([.medium, .large])
                    .presentationDragIndicator(.visible)
            })
    }
}
```

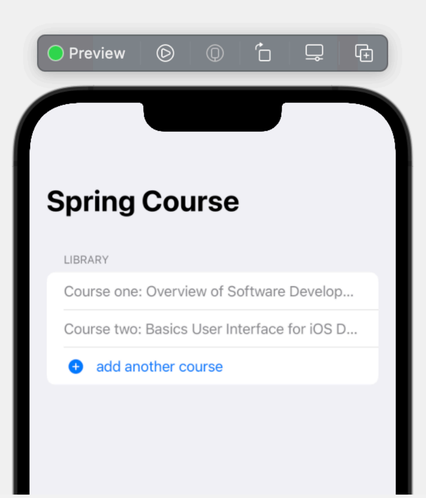


SwiftUI中的交互的视图组建

Interacting elements in SwiftUI



Form / List / Section



NavigationStack

TabView

SheetView

SwiftUI中的动画

The Animation in SwiftUI

浙江大学 ZHEJIANG UNIVERSITY

存储属性 `var playerName = "Yoo"`

- 是存储在`class`或`struct`中的属性，可以是常量或者变量
- 初始化：在定义时进行初始化，或者在`init()`函数中进行初始化

计算属性 `var playerBlood { ... }`

- 计算属性没有存储值，而是提供了一个`Getter`和一个可选的`Setter`，在引用这个属性时再进行计算

SwiftUI中的动画

The Animation in SwiftUI

浙江大学 ZHEJIANG UNIVERSITY

属性包装器 `@wrapperName`

- 属性包装器可以对属性的操作进行统一过滤

  `@State` 是SwiftUI用于管理存储属性的包装器，当被`@State`包装的属性发生更改时，当前的视图直接废止，并重新计算`body`的值

  一般来说：只在当前视图内部访问被`@State`包装的属性，
  防止外部干扰，所以被`@State`包装的属性一般声明为`private`

SwiftUI中的动画

The Animation in SwiftUI

存储属性　`var playerName = "Yoo"`

计算属性　`var playerBlood { ... }`

属性包装器　`@wrapperName`

@State装饰的属性改变时，body计算属性重新计算

```
struct PlayView: View{
    @State private var isPlaying: Bool = false

    var body: some View{
        HStack{
            Text("Start Play")
                .font(.title)
                .padding()
            Button(action:{
                self.isPlaying.toggle()
            }){
                Image(systemName: isPlaying ?
                    “pause.circle":"play.circle")
            }
        }
    }
}
```

bool值取非

SwiftUI中的动画

The Animation in SwiftUI

存储属性　`var playerName = "Yoo"`

计算属性　`var playerBlood { ... }`

属性包装器　`@wrapperName`

引起UI刷新之源头

```
struct PlayView: View{
    @State private var isPlaying: Bool = false

    var body: some View{
        HStack{
            Text("Start Play")
                .font(.title)
                .padding()
            Button(action:{
                self.isPlaying.toggle()

            }){
                Image(systemName: isPlaying ?
                   “pause.circle":"play.circle")
            }
        }
    }
}
```

SwiftUI中的动画

The Animation in SwiftUI

存储属性　var playerName = "Yoo"

计算属性　var playerBlood { ... }

属性包装器　@wrapperName

```
struct PlayView: View{
    @State private var isPlaying: Bool = false

    var body: some View{
        HStack{
            Text("Start Play")
                .font(.title)
                .padding()
            Button(action:{
                withAnimation(.spring()){
                    self.isPlaying.toggle()
                }
            }){
                Image(systemName: isPlaying ?
                    “pause.circle":"play.circle")
                .resizable()
                .frame(width: isPlaying ? 50 : 20,
                       height: isPlaying ? 50 : 20)
            }
        }
    }
}
```

Animation

SwiftUI中的动画

The Animation in SwiftUI

浙江大学
ZHEJIANG UNIVERSITY

TapGesture

LongPressGesture

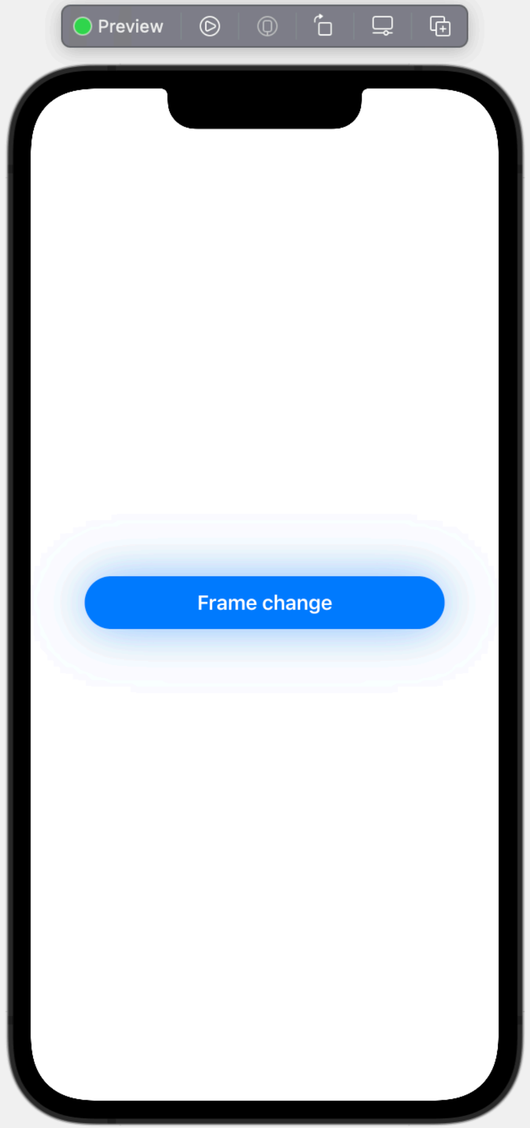
DragGesture

CustomGesture

SwiftUI中的动画

The Animation in SwiftUI

```
var ButtonA: some View{
    Text("Frame change")
        .bold()
        .foregroundColor(.white)
        .frame(width: changeFrame ? 320 : 300,
               height: changeFrame ? 100 : 44)
        .background(.blue)
        .cornerRadius(30)
        .shadow(color: Color.blue.opacity(0.5), radius: 20)
}
```

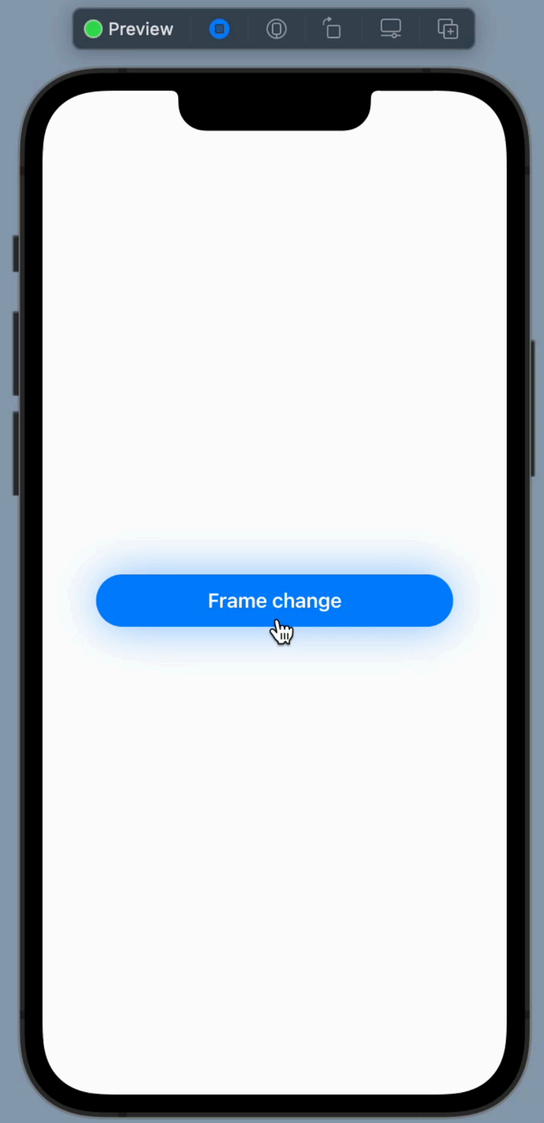


SwiftUI中的动画

The Animation in SwiftUI

```
@State var changeFrame = false

var body: some View {
    ButtonA
        .onTapGesture{
            withAnimation(.spring()){
                changeFrame.toggle()
            }
        }
}
```

== Button

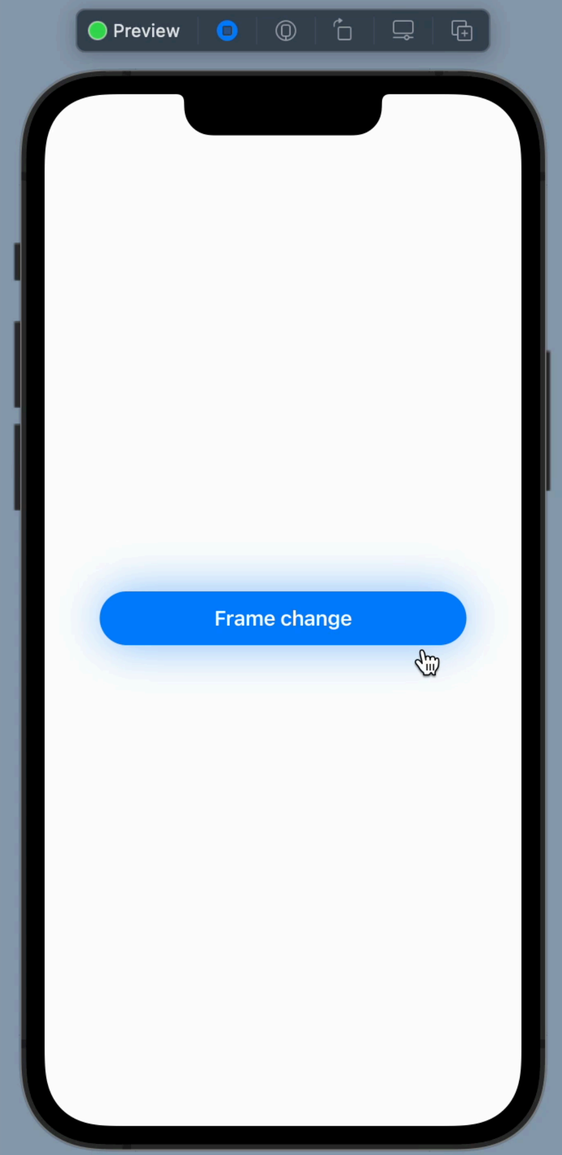


SwiftUI中的动画

The Animation in SwiftUI

```
@State var changeFrame = false

var body: some View {
    ButtonA
        .onTapGesture(count: 2){
            withAnimation(.spring()){
                changeFrame.toggle()
            }
        }
}
```

== Double tap



SwiftUI中使用MapKit

MapKit in SwiftUI

```
var body: some View {
    let parking = CLLocationCoordinate2D(
        latitude: 42.354528, longitude: -71.068369
      )

    Map {
          Annotation("Parking", coordinate: parking){
                ZStack {
                    Image(systemName: "car").padding(5)
                }
            }
        }
}
```