

SwiftUI 的 State property , data binding & 動畫

彼得潘

SwiftUI 的按鈕 button

[連結](#)

判斷 SwiftUI 的 function 型別參數 是否是 ViewBuilder

[連結](#)

用狀態設計 SwiftUI 畫面

認識 @State property

連結

講到練習: 骰子 App

簡化 SwiftUI 程式的 ?: ternary conditional operator



搭配 ?: 設定內容或呼叫 modifier，
讓 SwiftUI 程式變得更精簡易懂

```
if isStarFill {  
    Image(systemName: "star.fill")  
        .foregroundColor(.red)  
}  
else {  
    Image(systemName: "star")  
        .foregroundColor(.gray)  
}
```



```
Image(systemName: isStarFill ? "star.fill" : "star")  
    .foregroundColor(isStarFill ? .red : .gray)
```

連結

SwiftUI view 的型別定義裡，
可改變的 property 要用
@State var 宣告，少了 @State 的 var 是不
能改變的



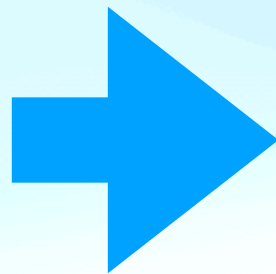
```
struct ContentView: View {  
    @State private var isRain = true
```



```
struct ContentView: View {  
    var isRain = true
```



@State 變數



var body: some View { }

當狀態改變時，畫面將馬上更新，
執行 body 的程式，
重新生成 body 裡受 state property 影響的元件

亂數: random

```
var isRain = Bool.random()  
var number1 = Int.random(in: 1...100)  
var number2 = Double.random(in: 0...1)
```


練習

骰子亂數 App

```
struct ContentView: View {  
    var body: some View {  
        Image(systemName: "die.face.1.fill")  
            .resizable()  
            .scaledToFit()  
            .frame(width: 200, height: 200)  
            .onTapGesture {  
                }  
    }  
}
```

點選時執行 {} 的程式

其它亂數 App 範例

連結 1

連結 2

連結 3

參考資料來源的 Binding

連結

練習 Binding

綁定資料的 Binding 元件

練習

滑動選值的 Slider

連結

練習 slider

連結

選時間的歲月神偷 DatePicker

[連結](#)

輸入文字的 TextField，SecureField， TextEditor & 猜數字 App

連結

alert



連結

選擇項目的 Picker



連結

利用 sheet 切換頁面

```
struct ContentView: View {  
    @State private var isShowingSecondView = false  
  
    var body: some View {  
        Button("show second view") {  
            isShowingSecondView = true  
        }  
        .sheet(isPresented: $isShowingSecondView) {  
            SecondView()  
        }  
    }  
}
```

```
struct SecondView: View {  
    var body: some View {  
        Text("Second View")  
    }  
}
```

```
M sheet(isPresented:content:)  
  sheet(isPresented:onDismiss:content:)  
M sheet(item:content:)  
  sheet(item:onDismiss:content:)  
M actionSheet(isPresented:content:) ⚠  
M actionSheet(item:content:) ⚠  
  
sheet(isPresented: Binding<Bool>, onDismiss: (() -> Void)?, content: () -> View)  
  -> View  
Presents a sheet when a binding to a Boolean value that you provide is true.
```

綁定的資料為 true 時顯示下一頁

Binding<Bool>
綁定型別為 Bool 的資料

自訂 button 返回前一頁

```
struct SecondView: View {
    @Binding var isShowingSecondView: Bool

    var body: some View {
        Button("close") {
            isShowingSecondView = false
        }
    }
}

#Preview {
    SecondView(isShowingSecondView: .constant(true))
}
```

利用 `.constant` 產生 Binding

自訂 button 返回前一頁

```
struct ContentView: View {  
    @State private var isShowingSecondView = false  
  
    var body: some View {  
        Button("show second view") {  
            isShowingSecondView = true  
        }  
        .sheet(isPresented: $isShowingSecondView) {  
            SecondView(isShowingSecondView: $isShowingSecondView)  
        }  
    }  
}
```

將 isShowingSecondView 的 Binding 傳到下一頁

animation 動畫

連結

練習 animation 動畫

設定 opacity 呈現蒙娜麗莎淡入動畫

```
struct ContentView: View {  
    @State private var opacity: Double = 0  
  
    var body: some View {  
        Image(.peter)  
            .opacity(opacity)  
            .animation(.easeInOut(duration: 5), value: opacity)  
            .onAppear {  
                opacity = 1  
            }  
    }  
}
```

ps: 畫面出現時開始動畫的效果在 preview 可能有問題，可改從模擬器測試

利用 transition 設定元件 出現 / 消失的動畫效果

transition: 轉變

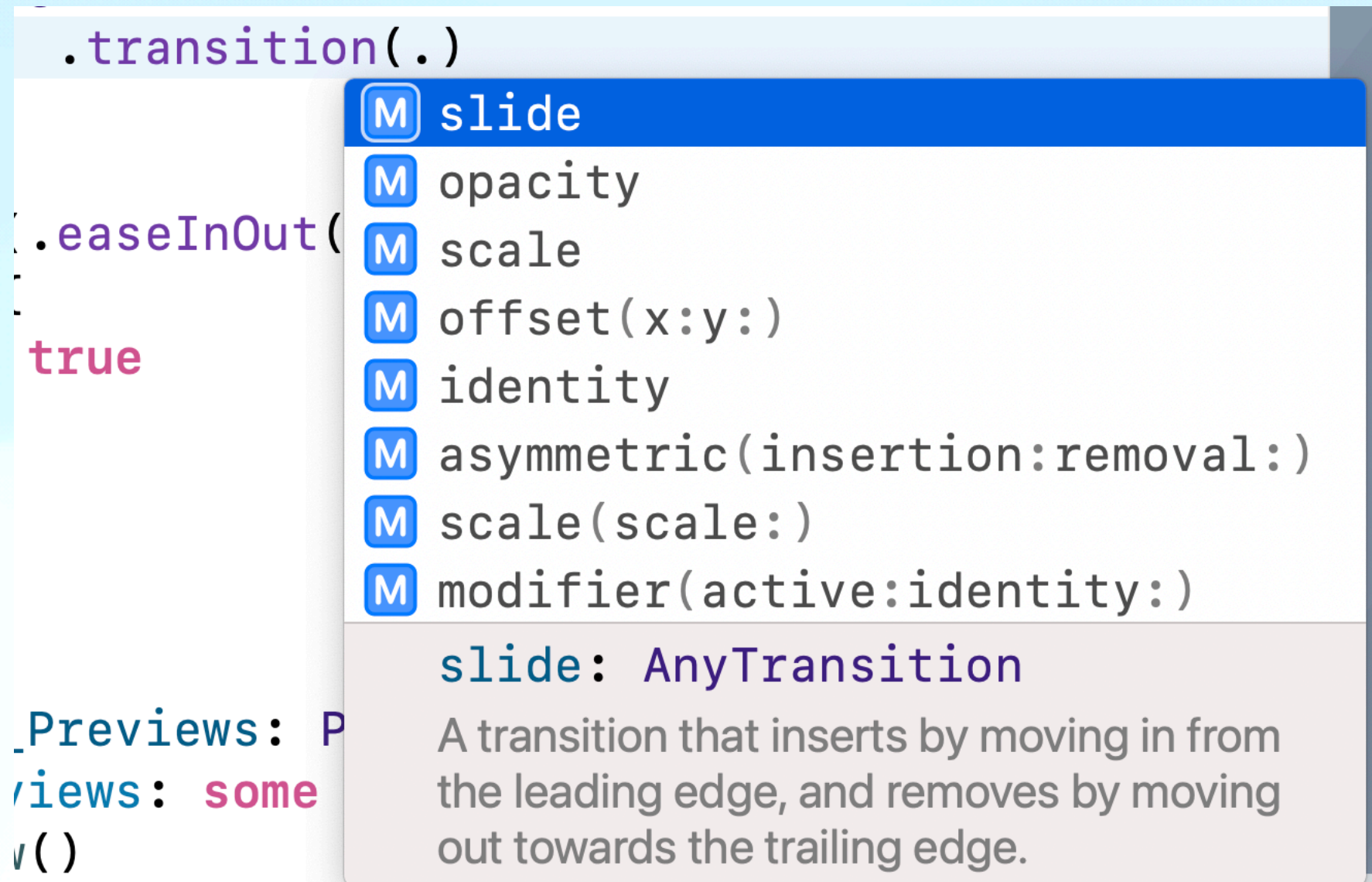
利用 transition 設定元件出現/消失的動畫

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            if show {
                Image(.peter)
                    .transition(.opacity)
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
        .onAppear {
            show = true
        }
    }
}
```

- transition: 轉場動畫，加在想設定出現/消失動畫的元件上
- 要搭配 animation，transition 才會有效果
- 舊版 Xcode 的 animation 沒有參數 value

多種內建的 transition



opacity: 淡入淡出的動畫

```
struct ContentView: View {
    @State private var show = false


    var body: some View {
        VStack {
            if show {
                Image(.peter)
                    .transition(.opacity)
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
        .onAppear {
            show = true
        }
    }
}
```

以淡入動畫慢慢出現

animation 要加在包含 Image 的 VStack 才有效果，加在 Image 沒有效果

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            if show {
                Image(.peter)
                    .transition(.opacity)
                    .animation(.easeInOut(duration: 5), value: show)
            }
        }
        .onAppear {
            show = true
        }
    }
}
```



scale: 放大縮小的動畫

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            if show {
                Image(.peter)
                    .transition(.scale)
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
        .onAppear {
            show = true
        }
    }
}
```

搭配參數的 transition

```
struct ContentView: View {  
    @State private var show = false  
  
    var body: some View {  
        VStack {  
            if show {  
                Image(.peter)  
                    .transition(.scale(scale: 3))  
            }  
        }  
        .animation(.easeInOut(duration: 5), value: show)  
        .onAppear {  
            show = true  
        }  
    }  
}
```

出現時由 3 倍大慢慢縮小到 1 倍

利用 combined 結合多種 transition

新版寫法

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            if show {
                Image(.peter)
                    .transition(.scale(scale:
3).combined(with: .opacity))
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
        .onAppear {
            show = true
        }
    }
}
```

出現時由 3 倍大慢慢縮小到 1 倍，搭配淡入動畫慢慢出現

宣告 computed property 簡化程式

```
struct ContentView: View {  
    @State private var show = false  
    var scaleAndOpacityTransition: AnyTransition {  
        .scale(scale: 3).combined(with: .opacity)  
    }  
  
    var body: some View {  
        VStack {  
            if show {  
                Image(.peter)  
                    .transition(scaleAndOpacityTransition)  
            }  
        }  
        .animation(.easeInOut(duration: 5), value: show)  
        .onAppear {  
            show = true  
        }  
    }  
}
```

SwiftUI 精簡程式的 5 個方法

連結

練習 transition

出現 & 消失的動畫

點選 button 出現 / 消失

```
struct ContentView: View {  
    @State private var show = false  
  
    var body: some View {  
        VStack {  
            Button(show ? "hide" : "show") {  
                show.toggle()  
            }  
            if show {  
                Image(.peter)  
                    .transition(.opacity)  
            }  
        }  
        .animation(.easeInOut(duration: 5), value: show)  
    }  
}
```

出現 & 消失的動畫效果預設將會相反，因此圖片出現時將淡入，圖片消失時將淡出

問題: 文字移動

點選 button 出現 / 消失

```
struct ContentView: View {  
    @State private var show = false  
  
    var body: some View {  
        VStack {  
            Button(show ? "hide" : "show") {  
                show.toggle()  
            }  
            if show {  
                Image(.peter)  
                    .transition(.opacity)  
            } else {  
                Image(.peter)  
                    .hidden()  
            }  
        }  
        .animation(.easeInOut(duration: 5), value: show)  
    }  
}
```

圖片不顯示時依然佔著空間，圖片出現/消失時，文字不會再跟著移動

讓文字不要有 animation

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            Button(show ? "hide" : "show") {
                show.toggle()
            }
            .animation(nil, value: show)

            if show {
                Image(.peter)
                    .transition(.opacity)
            } else {
                Image(.peter)
                    .hidden()
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
    }
}
```

出現 & 消失設定不同的動畫效果 asymmetric (不對稱)

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            Button(show ? "hide" : "show") {
                show.toggle()
            }
            .animation(nil, value: show)

            if show {
                Image(.peter)
                    .transition(.asymmetric(insertion: .scale,
removal: .slide))
            } else {
                Image(.peter)
                    .hidden()
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
    }
}
```

出現時由小到大，消失時水平移動

自訂 transition

```
extension AnyTransition {  
    static var customTransition: AnyTransition {  
        let insertion = AnyTransition.move(edge: .trailing)  
            .combined(with: .opacity)  
        let removal = AnyTransition.offset(x: 200, y: 200)  
            .combined(with: .opacity)  
        return .asymmetric(insertion: insertion, removal: removal)  
    }  
}
```


自訂 transition

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            Button(show ? "hide" : "show") {
                show.toggle()
            }
            .animation(nil, value: show)

            if show {
                Image(.peter)
                .transition(.customTransition)
            } else {
                Image(.peter)
                .hidden()
            }
        }
        .animation(.easeInOut(duration: 5), value: show)
    }
}
```


練習

另一種產生動畫的寫法

呼叫 withAnimation

```
struct ContentView: View {
    @State private var show = false

    var body: some View {
        VStack {
            if show {
                Image(.peter)
                    .transition(.opacity)
            }
        }
        .onAppear {
            withAnimation(.easeInOut(duration: 5)) {
                show = true
            }
        }
    }
}
```

SF Symbol 動畫

連結

動畫範例

連結