SwiftUI

Ryan Chung

課程大綱

- 顯示使用者名稱
  - 跨頁永續存取 AppStorage
- 自由自在的繪圖世界
  - 矩形 Path
  - 圓餅 / 進度條 / 救生圈
- 精選課程呈現
  - 客製化清單 List
- 手勢
  - 長按 LongTapGesture
  - 點擊 TapGesture
  - 拖拉 DragGesture
- 使用者輸入驗證
  - 獨立資料處理模型 UserRegistrationModel
  - 依據使用者輸入進行變化 Ternary Operator
- @家族
  - 內部 @State
  - 上層 @Binding
  - 客製 @ObservedObject
  - 環境 @EnvironmentObject
  - 發行 @Published

依據使用者名稱顯示

- 使用前一個HelloTab專案
- 在設定中使用者可輸入名稱
- 回到歡迎頁可以看到自己的名稱
  - 如果沒有輸入則不會顯示

SettingView

- @AppStorage
- TextField

```
import SwiftUI

struct SettingView: View {
    private var displayFontType = [".default",
                ".rounded",".monospaced",".serif"]
    @State var displayFontSelected = 0
    @State var IsDeepScheme = false
    @State var colorArray:Array = [255.0,255.0,255.0]
    @State var stepperValue = 0
    @State var sliderValue = 0.0

    @AppStorage("UserName") var UserName:String = ""

    var body: some View {
        Form{
            Section(header: Text("使用者名稱")) {
                TextField("請輸入您的名字", text: $UserName)
            }
            Section(header: Text("字型設定")){
```

WelcomeView

- AppStorage取用、有值沒值的處理

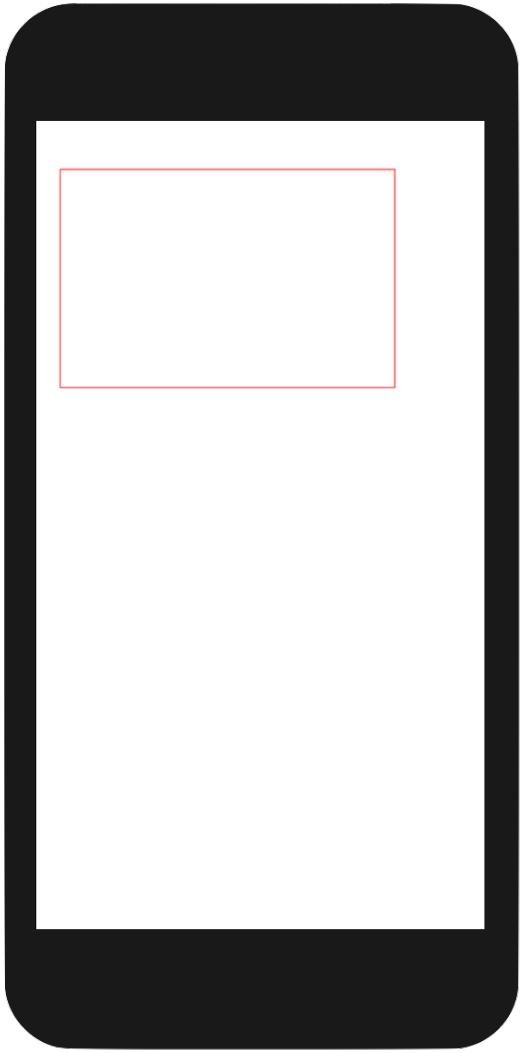
```
struct WelcomeView: View {

    @AppStorage("UserName") var UserName:String = ""

    var body: some View {
        VStack{
            Image("mobiledevtw")
                .resizable()
                .aspectRatio(contentMode: .fit)
                .overlay(
                    Text(UserName.isEmpty ? "" : "歡迎 \(UserName)")
                        .font(.title)
                        .foregroundColor(.purple)
                        .offset(x: -80, y: -150)
                )
            Text("資訊技術培訓\nWeb/APP/AI應用開發")
```

使用模擬器進行測試

- 修改會更新、無值不顯示、重開還會在

自由自在的繪圖世界

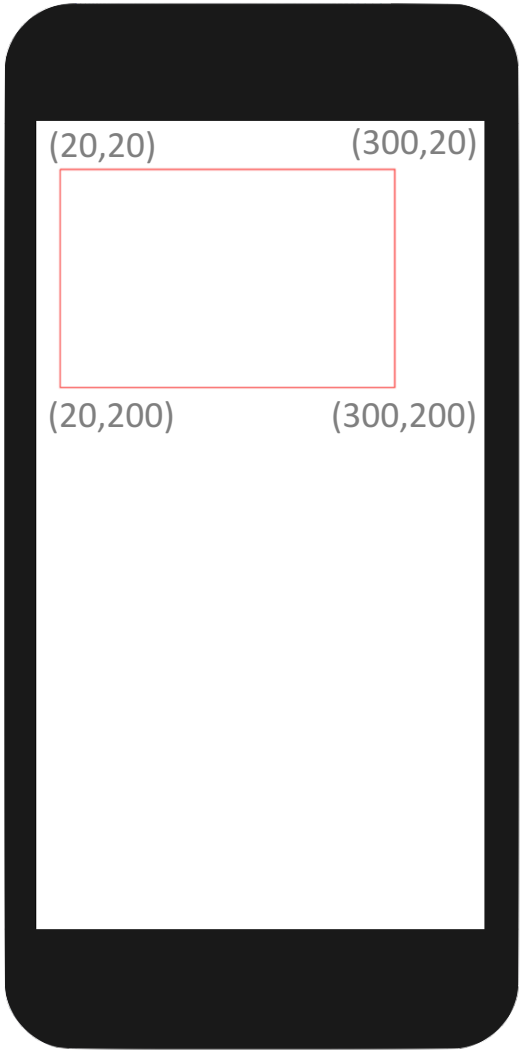
- Path( )



紅色框框

- Path( )

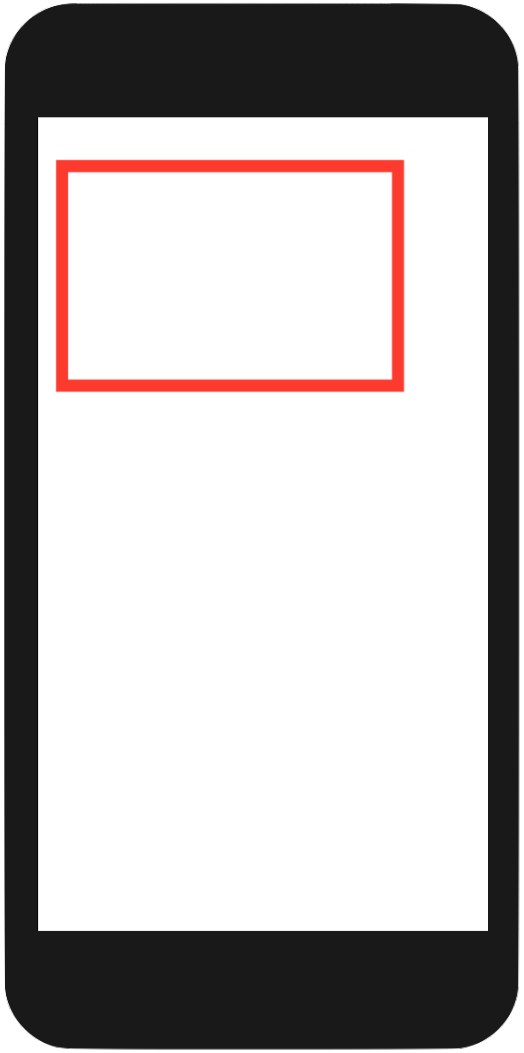
```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Path(){ myPath in
            myPath.move(to: CGPoint(x: 20, y: 20))
            myPath.addLine(to: CGPoint(x: 300, y: 20))
            myPath.addLine(to: CGPoint(x: 300, y: 200))
            myPath.addLine(to: CGPoint(x: 20, y: 200))
            myPath.addLine(to: CGPoint(x: 20, y: 20))
        }
        .stroke(Color.red)
    }
}
```



自動連結回起點

- lineWidth設定線條粗細

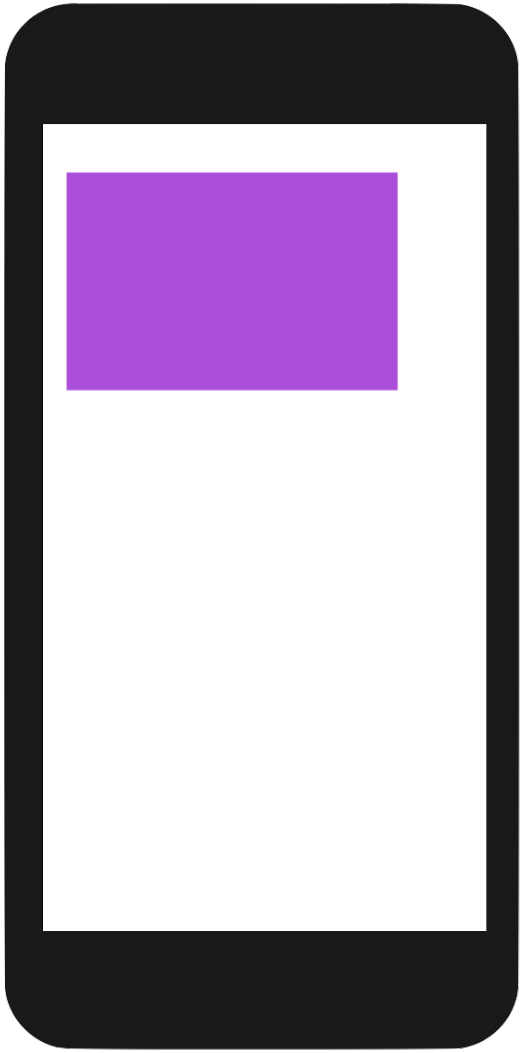
```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Path(){ myPath in
            myPath.move(to: CGPoint(x: 20, y: 20))
            myPath.addLine(to: CGPoint(x: 300, y: 20))
            myPath.addLine(to: CGPoint(x: 300, y: 200))
            myPath.addLine(to: CGPoint(x: 20, y: 200))
            myPath.closeSubpath()
        }
        .stroke(Color.red, lineWidth: 10)
    }
}
```



紫色實心矩形

- Path( )

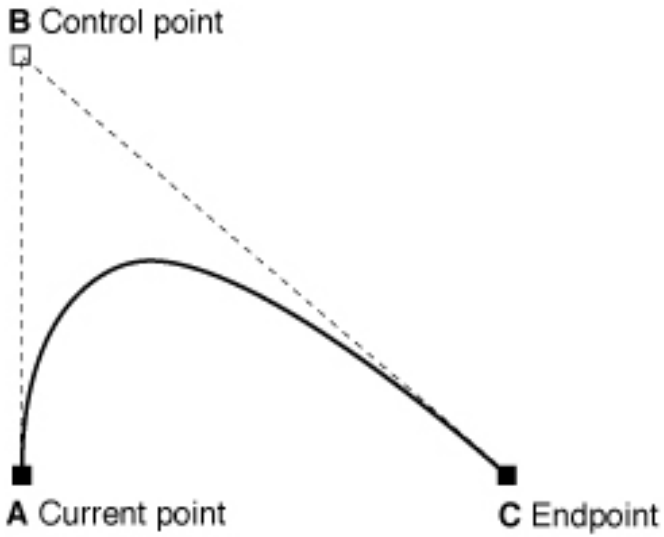
```
import SwiftUI

struct ContentView: View {
    var body: some View {
        Path(){ myPath in
            myPath.move(to: CGPoint(x: 20, y: 20))
            myPath.addLine(to: CGPoint(x: 300, y: 20))
            myPath.addLine(to: CGPoint(x: 300, y: 200))
            myPath.addLine(to: CGPoint(x: 20, y: 200))
            myPath.addLine(to: CGPoint(x: 20, y: 20))
        }
        .fill(Color.purple)
    }
}
```

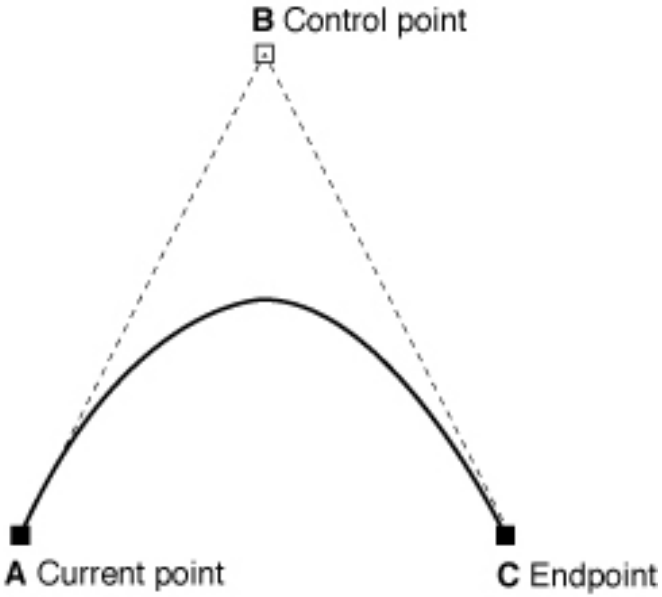


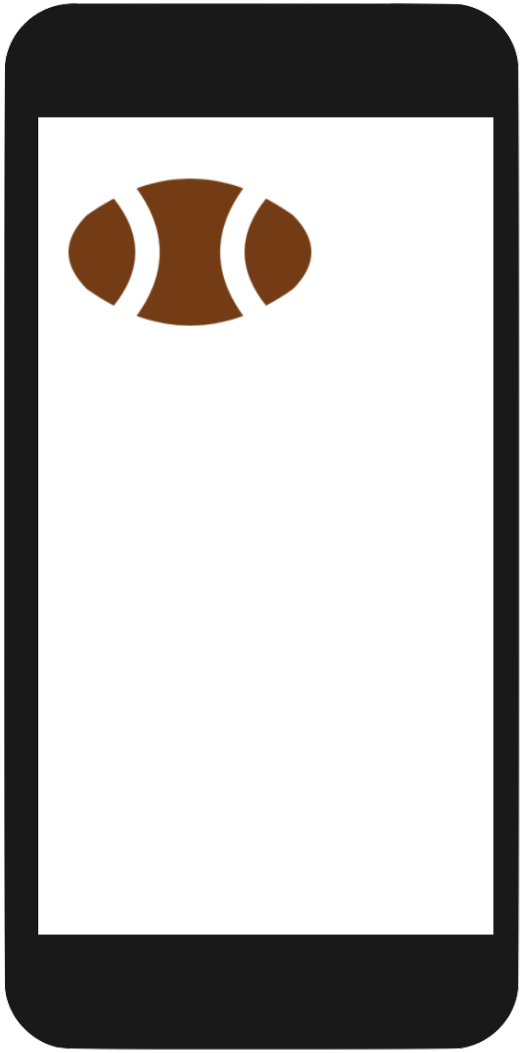
教練，我想打橄欖球

- addQuadCurve(to:control:)

```
myPath.move(to: CGPoint(x: 40, y: 60))
myPath.addQuadCurve(to: CGPoint(x: 210, y: 60),
                    control: CGPoint(x: 125, y: 0))
```





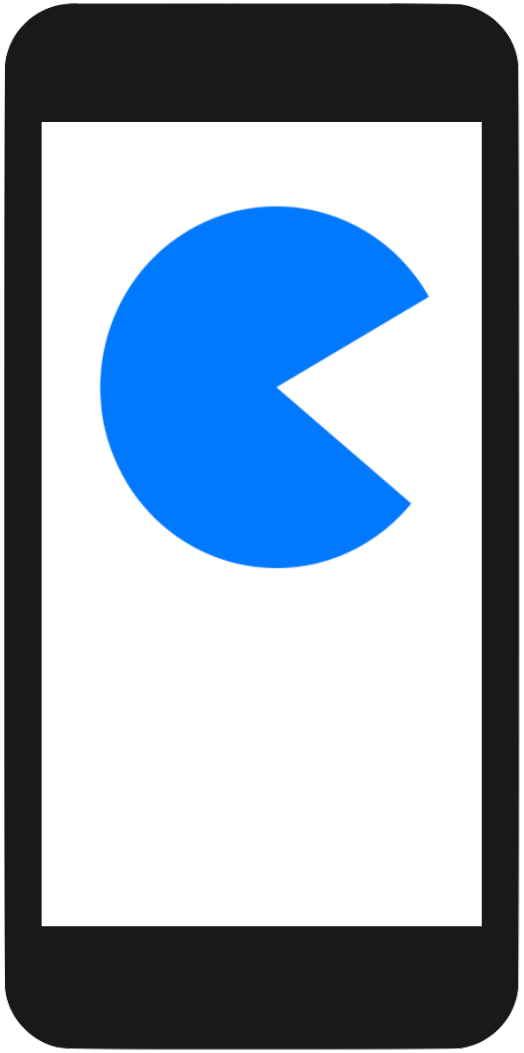


Zstack(球體+左白線+右白線)

```
struct ContentView: View {
    var body: some View {
        ZStack{
            Path(){ myPath in
                myPath.move(to: CGPoint(x: 40, y: 60))
                myPath.addQuadCurve(to: CGPoint(x: 210, y: 60), control: CGPoint(x: 125, y: 0))
                myPath.addQuadCurve(to: CGPoint(x: 210, y: 120), control: CGPoint(x: 240, y: 90))
                myPath.addQuadCurve(to: CGPoint(x: 40, y: 120), control: CGPoint(x: 125, y: 180))
                myPath.addQuadCurve(to: CGPoint(x: 40, y: 60), control: CGPoint(x: 10, y: 90))
            }
            .fill(Color(red: 117/255, green: 63/255, blue: 22/255))
            Path(){ myPath in
                myPath.move(to: CGPoint(x: 70, y: 40))
                myPath.addQuadCurve(to: CGPoint(x: 70, y: 140), control: CGPoint(x: 110, y: 90))
            }
            .stroke(Color.white, lineWidth: 20)
            Path(){ myPath in
                myPath.move(to: CGPoint(x: 180, y: 40))
                myPath.addQuadCurve(to: CGPoint(x: 180, y: 140), control: CGPoint(x: 140, y: 90))
            }
            .stroke(Color.white, lineWidth: 20)
        }
    }
}
```

圓餅圖

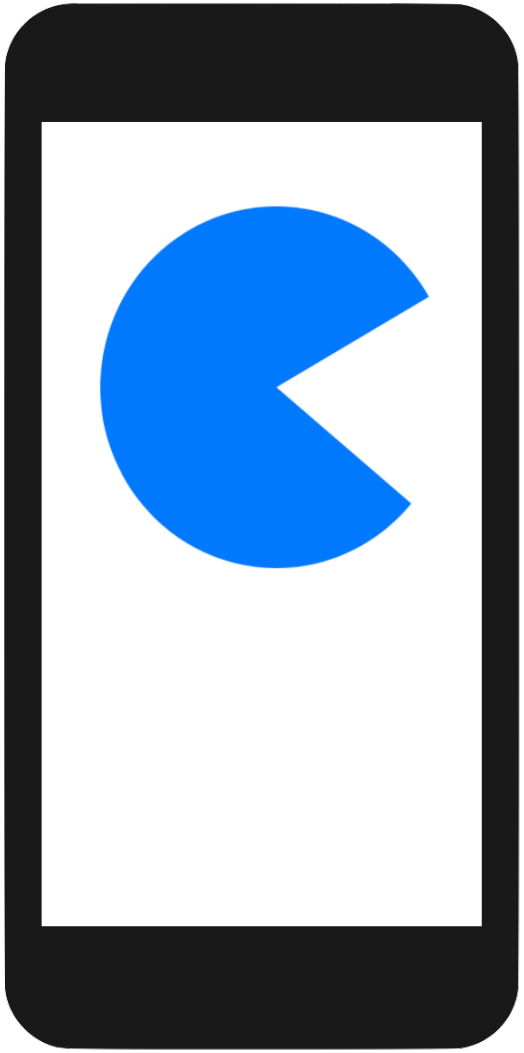
- 先移動至圓心
- 利用addArc( )畫出弧
- 塗色



小精靈

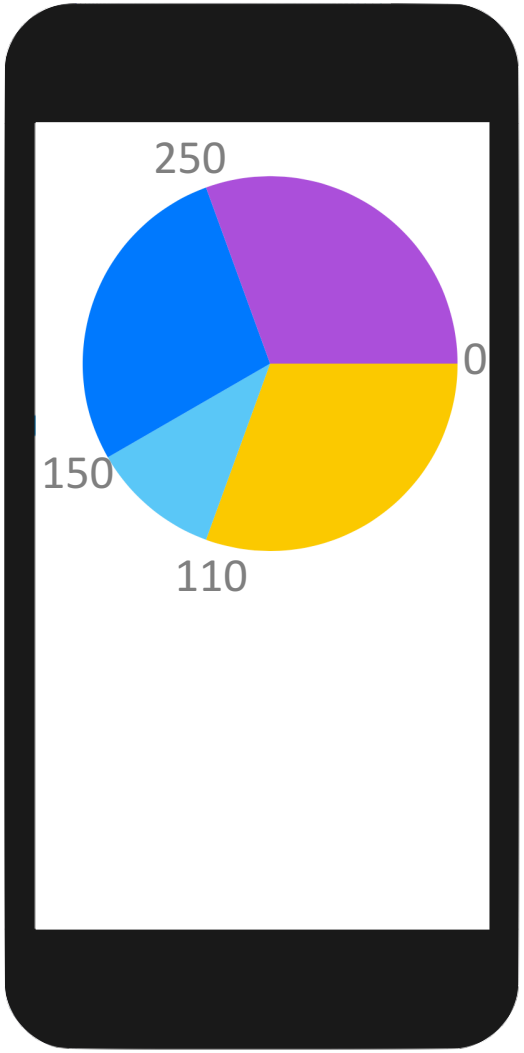
- addArc( )

```
struct ContentView: View {
    var body: some View {
        Path{ path in
            path.move(to: CGPoint(x: 200, y: 200))
            path.addArc(center: CGPoint(x: 200, y: 200),
                        radius: 150,
                        startAngle: Angle(degrees: 40),
                        endAngle: Angle(degrees: 330),
                        clockwise: false)
        }.fill(Color.blue)
    }
}
```



圓餅圖

```
struct ContentView: View {
    var body: some View {
        ZStack {
            Path { path in
                path.move(to: CGPoint(x: 200, y: 200))
                path.addArc(center: .init(x: 200, y: 200), radius: 150,
                    startAngle: Angle(degrees: 0.0), endAngle: Angle(degrees: 110),
                    clockwise: false)
            }
            .fill(Color(.systemYellow))
            Path { path in
                path.move(to: CGPoint(x: 200, y: 200))
                path.addArc(center: .init(x: 200, y: 200), radius: 150,
                startAngle: Angle(degrees: 110), endAngle: Angle(degrees: 150),
                clockwise: false)
            }
            .fill(Color(.systemTeal))
            Path { path in
                path.move(to: CGPoint(x: 200, y: 200))
                path.addArc(center: .init(x: 200, y: 200), radius: 150,
                startAngle: Angle(degrees: 150), endAngle: Angle(degrees: 250),
                clockwise: false)
            }
            .fill(Color(.systemBlue))
            Path { path in
                path.move(to: CGPoint(x: 200, y: 200))
                path.addArc(center: .init(x: 200, y: 200), radius: 150,
                startAngle: Angle(degrees: 250), endAngle: Angle(degrees: 360),
                clockwise: false)
            }
            .fill(Color(.systemPurple))
        }
    }
}
```



圓餅圖強化 – 分離

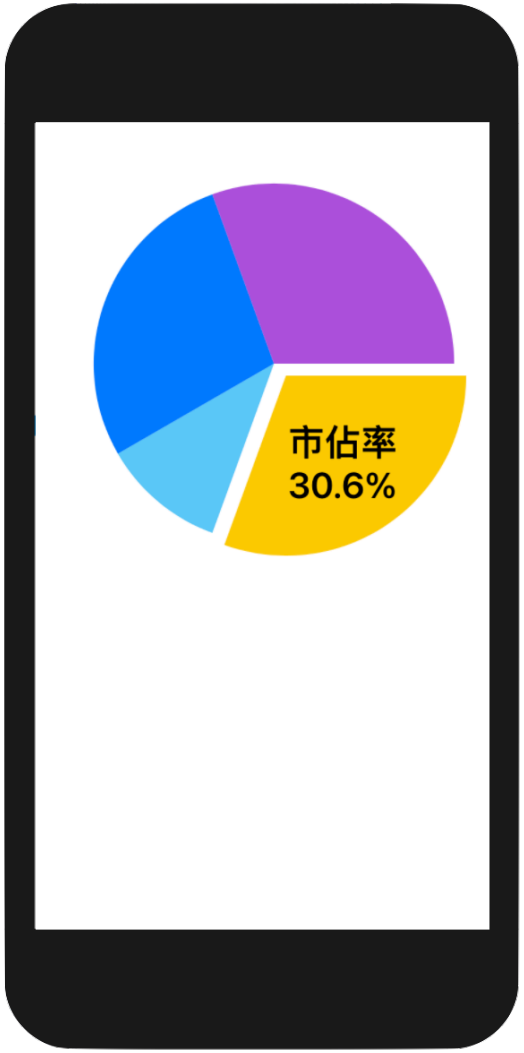
- offset

```
import SwiftUI


struct ContentView: View {
    var body: some View {
        ZStack {
            Path { path in
                path.move(to: CGPoint(x: 200, y: 200))
                path.addArc(center: .init(x: 200, y: 200),
                radius: 150,
                startAngle: Angle(degrees: 0.0),
                endAngle: Angle(degrees: 110),
                clockwise: false)
            }
            .fill(Color(.systemYellow))
            .offset(x: 10.0, y: 10.0)
```

圓餅圖強化 – 疊字

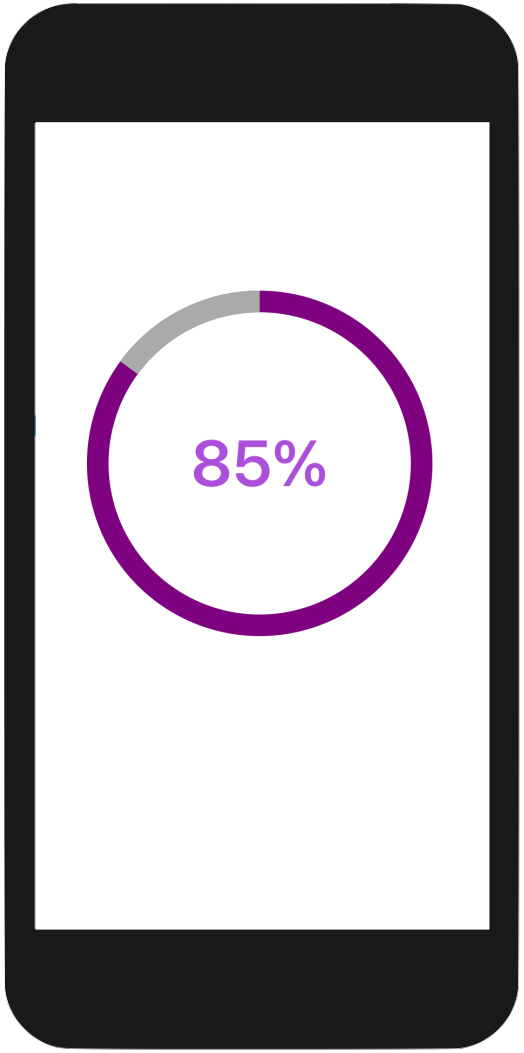
- overlay

```
import SwiftUI

struct ContentView: View {
    var body: some View {
        ZStack {
            Path { path in
                path.move(to: CGPoint(x: 200, y: 200))
                path.addArc(center: .init(x: 200, y: 200),
                radius: 150,
                startAngle: Angle(degrees: 0.0),
                endAngle: Angle(degrees: 110),
                clockwise: false)
            }
            .fill(Color(.systemYellow))
            .offset(x: 10.0, y: 10.0)
            .overlay(
                Text("市佔率\n30.6%")
                    .font(.system(size: 30))
                    .bold()
                    .offset(x: 70, y: -40)
            )
```



圓形進度條

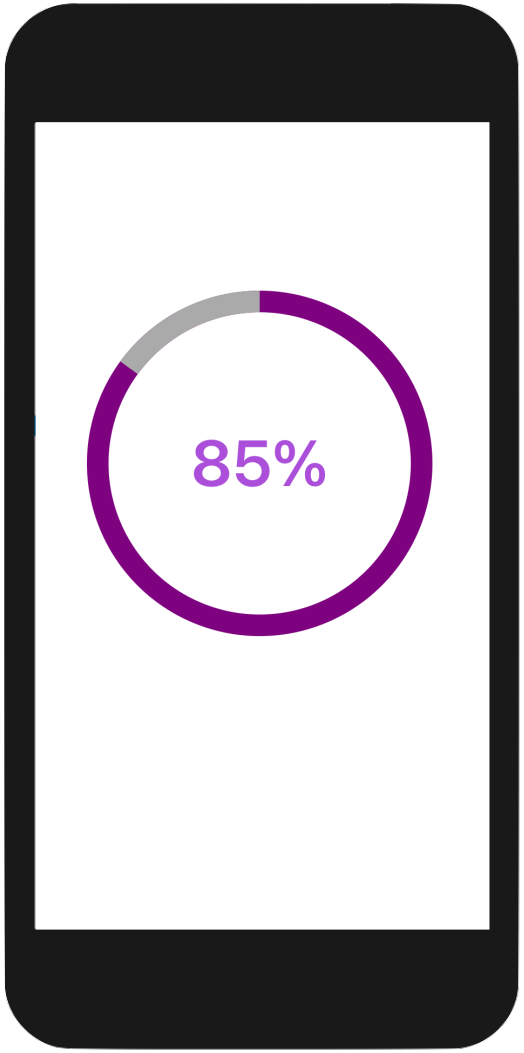
- 用圓形線條呈現進度
- 中間顯示數字



圓形進度條

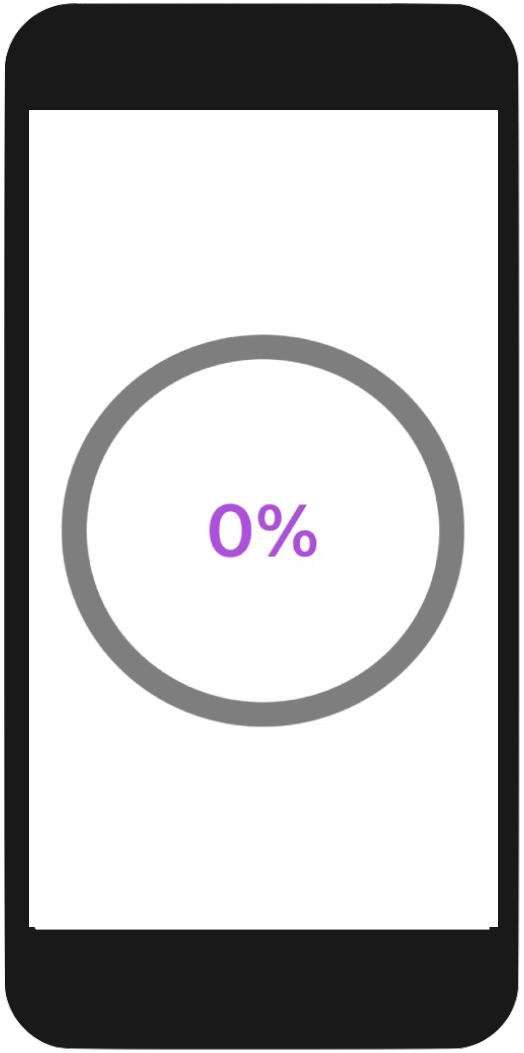
- 內建Circle( )
- ZStack + overlay( )

```
struct ContentView: View {
    var body: some View {
        ZStack{
            Circle()
                .stroke(Color(.purple), lineWidth: 20)
                .frame(width: 300, height: 300, alignment: .center)
            Circle()
                .trim(from:0.60, to:0.75)
                .stroke(Color(.lightGray), lineWidth: 20)
                .frame(width: 300, height: 300, alignment: .center)
                .overlay(
                    Text("85%")
                        .font(.system(size: 60))
                        .bold()
                        .foregroundColor(.purple)
                )
        }
    }
}
```



真的圓形進度條

- 會隨著時間或真實進度前進
  - 每1秒完成10%
  - 每0.5秒更新一次進度
- 問題
  - 圓圈的0度是從右邊中間開始的
  - 如何產生計時器

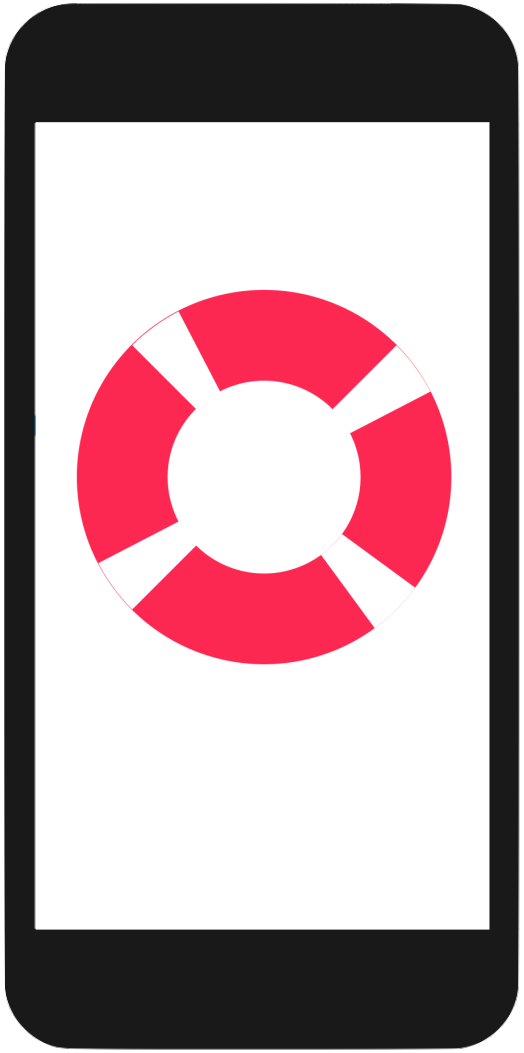


真的圓形進度條

```
struct ContentView: View {
    @State var progress:CGFloat = 0.0
    var body: some View {
        ZStack{
            Circle()
                .stroke(Color(.gray), lineWidth: 20)
                .frame(width: 300, height: 300, alignment: .center)
            Circle()
                .trim(from:0, to:progress)
                .stroke(Color(.purple), lineWidth: 20)
                .frame(width: 300, height: 300, alignment: .center)
                .rotationEffect(Angle(degrees: -90))
                .overlay(
                    Text("\(Int(progress*100))%")
                        .font(.system(size: 60))
                        .bold()
                        .foregroundColor(.purple)
                )
        }
        .onAppear {
            Timer.scheduledTimer(withTimeInterval: 0.5, repeats: true){ timer in
                self.progress+=0.05
                if self.progress>=1.0{
                    timer.invalidate()
                }
            }
        }
    }
}
```

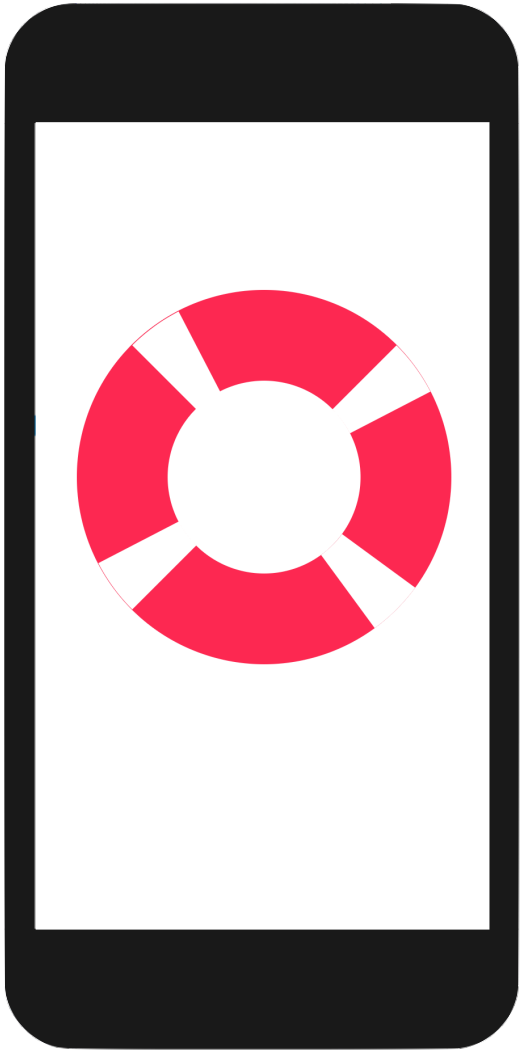
來個救生圈

- Circle( )
- 搭配trim( )



來個救生圈

```
struct ContentView: View {
    var body: some View {
        ZStack{
            Circle()
                .stroke(Color(.systemPink), lineWidth: 80)
                .frame(width: 250, height: 250, alignment: .center)
            Circle()
                .trim(from:0.1, to:0.15)
                .stroke(Color(.white), lineWidth: 80)
                .frame(width: 250, height: 250, alignment: .center)
            Circle()
                .trim(from:0.375, to:0.425)
                .stroke(Color(.white), lineWidth: 80)
                .frame(width: 250, height: 250, alignment: .center)
            Circle()
                .trim(from:0.625, to:0.675)
                .stroke(Color(.white), lineWidth: 80)
                .frame(width: 250, height: 250, alignment: .center)
            Circle()
                .trim(from:0.875, to:0.925)
                .stroke(Color(.white), lineWidth: 80)
                .frame(width: 250, height: 250, alignment: .center)
        }
    }
}
```



精選課程呈現

- 客製化清單
  - 增加是否為精選課程屬性
  - 是的話用另外一種格式呈現
- 使用先前的HelloTab專案修改

FullImageRow

- 在BasicImageRow下方再增加這個struct

```
struct FullImageRow: View {
        var thisCourse:Course
        var body: some View {
            ZStack{
                Image(thisCourse.image)
                    .resizable()
                    .aspectRatio(contentMode: .fill)
                    .frame(height: 180)
                    .cornerRadius(10)
                    .overlay(
                        Rectangle()
                            .foregroundColor(.black)
                            .cornerRadius(10)
                            .opacity(0.2)
                    )
                Text(thisCourse.name)
                    .font(.system(.title))
                    .fontWeight(.black)
                    .foregroundColor(.white)
                    .offset(x: 0.0, y: 50.0)
            }
        }
    }
```

修改課程Struct並更新資料

```
struct Course: Identifiable{
        var id = UUID()
        var name:String
        var image:String
        var description:String
        var isFeature:Bool
    }


var courses = [
      Course(name: "Intro to AI", image: "ai", description: "人工智慧介紹",
             isFeature:true),
      Course(name: "Intro to Python", image: "python", description: "Python介紹",
             isFeature:true),
      Course(name: "Intro to Data Science", image: "data-science", description: "資料科學介紹",
            isFeature:false),
      Course(name: "Intro to Azure", image: "azure", description: "微軟Azure介紹",
            isFeature:false),
      Course(name: "Intro to AWS", image: "aws", description: "亞馬遜AWS介紹",
            isFeature:false),
   ]
```

增加條件判斷

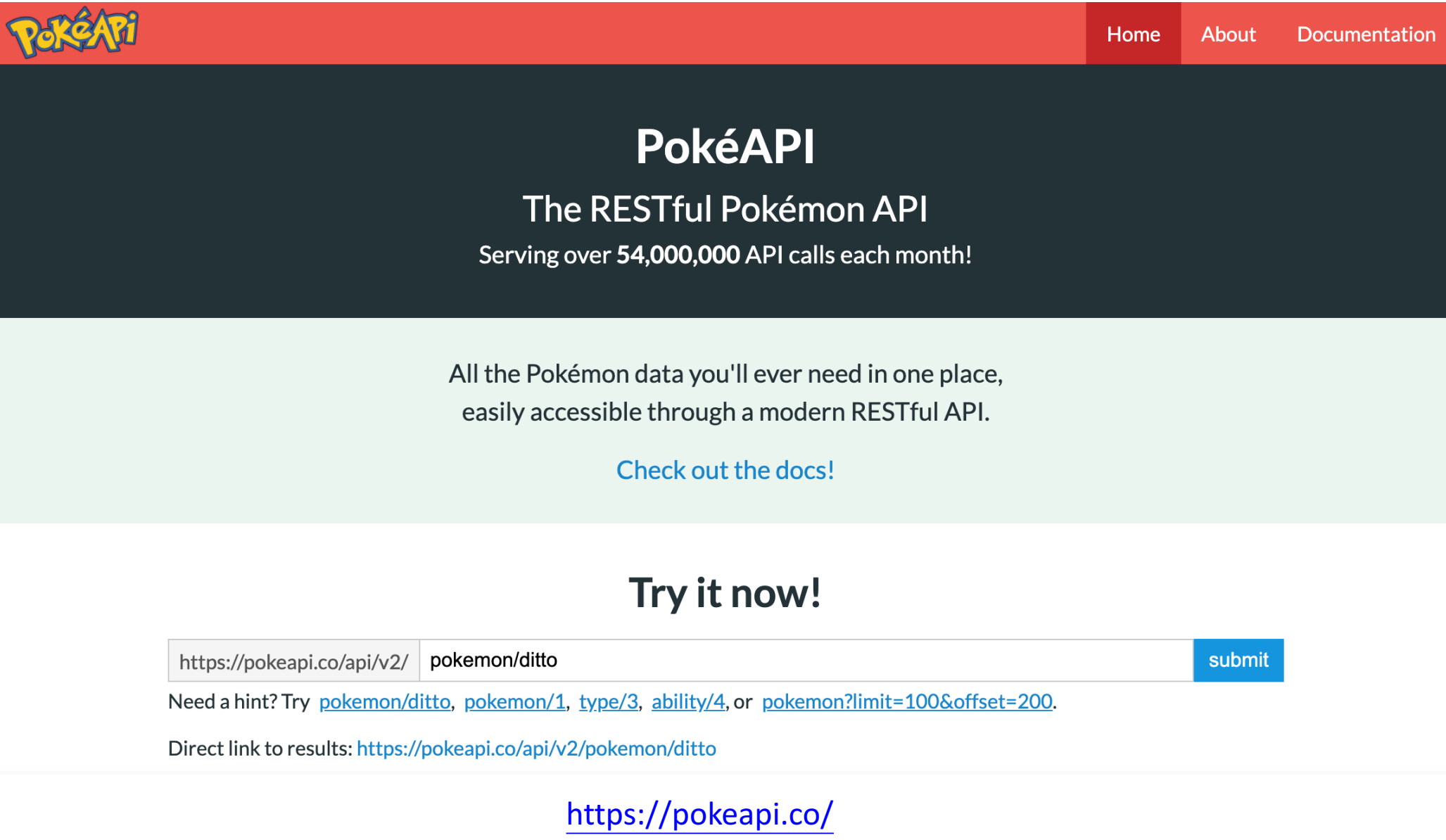
- courseItem的isFeature是否為真

```
var body: some View {
        NavigationView{
            List(courses){ courseItem in
                if courseItem.isFeature{
                    FullImageRow(thisCourse: courseItem)
                        .onTapGesture {
                            self.showDetailView = true
                            self.selectedCourse = courseItem
                        }
                }else{
                    BasicImageRow(thisCourse: courseItem)
                        .onTapGesture {
                            self.showDetailView = true
                            self.selectedCourse = courseItem
                        }
                }

            }
            .sheet(item: self.$selectedCourse){ thisCourse in
                CourseDetailView(course: thisCourse)
            }
            .navigationBarTitle("課程列表")
        }
    }
```

手勢

- 按住可移動
- 點擊跳一下轉身
- 長按換顏色

準備圖片



https://pokeapi.co/

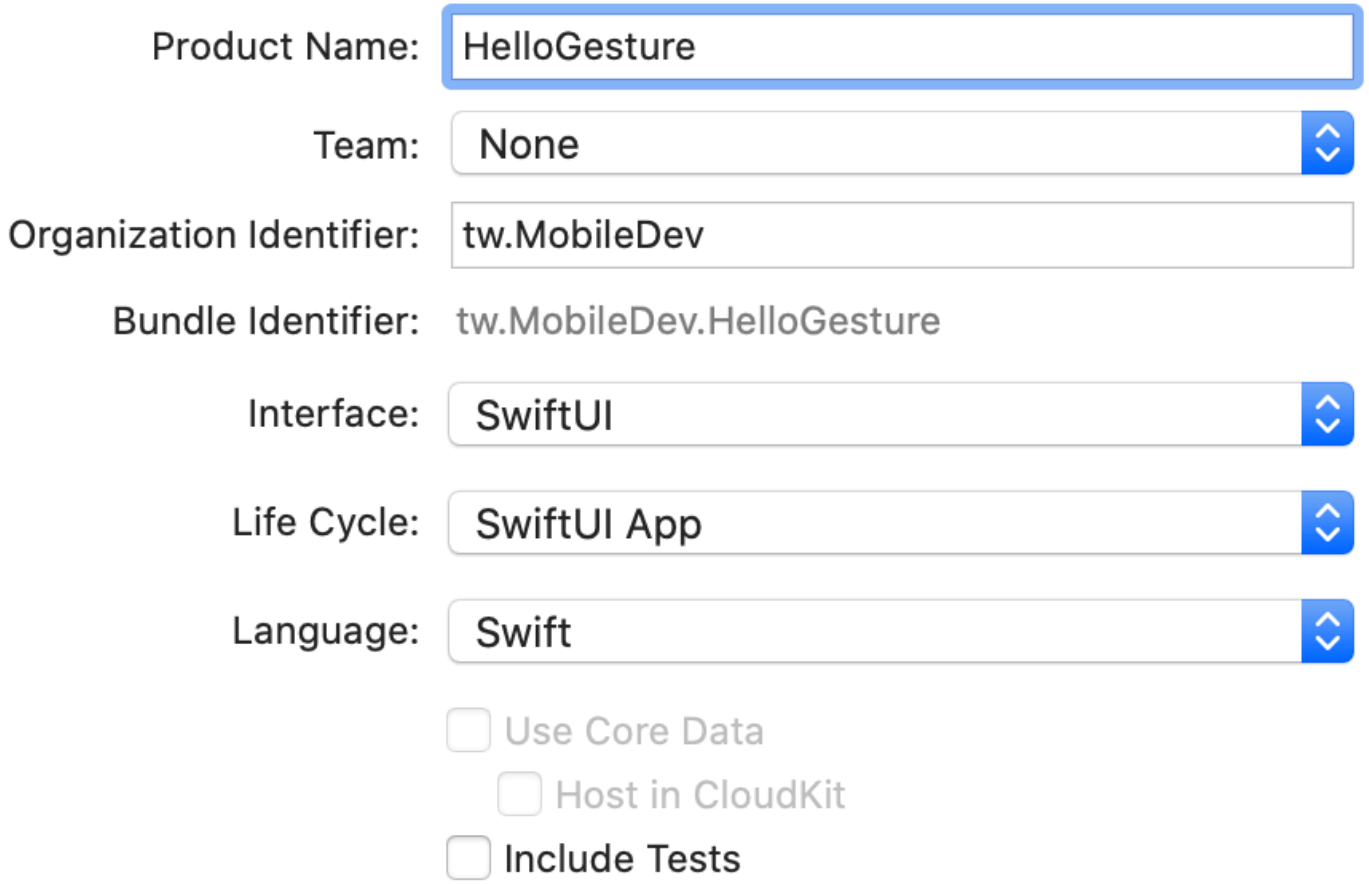
下載圖片

- 挑一隻喜歡的，下載牠的四張圖片
  - Front
  - Front-shiny
  - Back
  - Back-shiny

https://pokeapi.co/api/v2/pokemon-form/25/

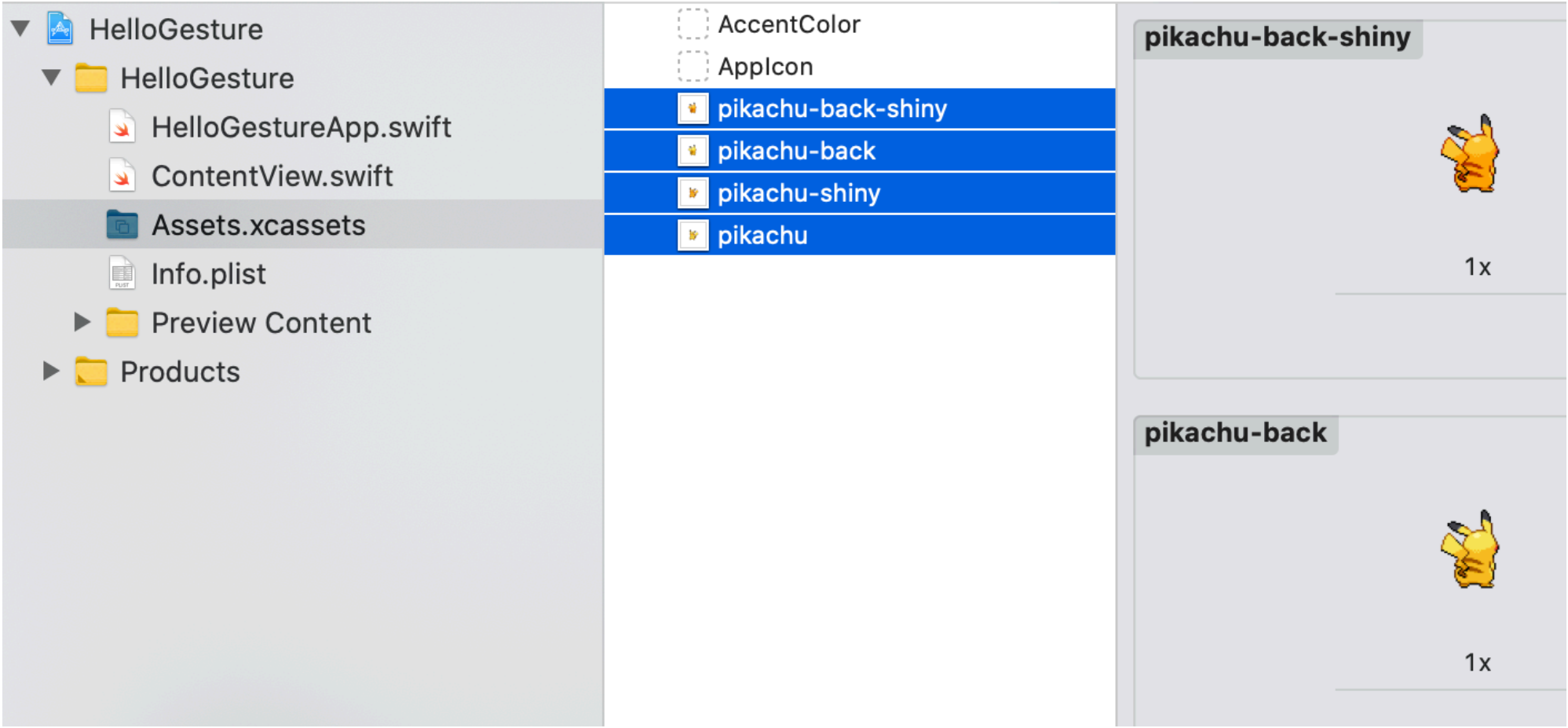
新增專案 - HelloGesture

- iOS -> App



將圖片放入專案中

- 導覽區 -> Assets.xcassets



長按換顏色

- onLongTapGesture()

```
import SwiftUI

struct ContentView: View {

    var imageNames:Array = [
        "pikachu",
        "pikachu-shiny",
        "pikachu-back",
        "pikachu-back-shiny"
    ]
    @State var currentImage:Int=0

    var body: some View {
        Image(imageNames[currentImage])
            .resizable()
            .frame(width: 400, height: 400, alignment: .center)
            .onLongPressGesture {
                if currentImage<=1{
                    currentImage = 1 - currentImage
                }else{
                    currentImage = 5 - currentImage
                }
            }
    }
}
```

點擊跳一下轉身

- onTapGesture(), scaleEffect()

```
struct ContentView: View {
    var imageNames:Array = [
        "pikachu",
        "pikachu-shiny",
        "pikachu-back",
        "pikachu-back-shiny"
    ]
    @State var currentImage:Int=0
    @State var isPressed = false
    var body: some View {
        Image(imageNames[currentImage])
            .resizable()
            .frame(width: 400, height: 400, alignment: .center)
            .scaleEffect(isPressed ? 0.5:2.0)
            .animation(.easeInOut)
            .scaleEffect(isPressed ? 2.0:0.5)
            .animation(.easeInOut)
            .onTapGesture {
                self.isPressed.toggle()
                if currentImage==0 || currentImage==2{
                    currentImage = 2 - currentImage
                }else{
                    currentImage = 4 - currentImage
                }
            }
    }
}
```

拖著跑

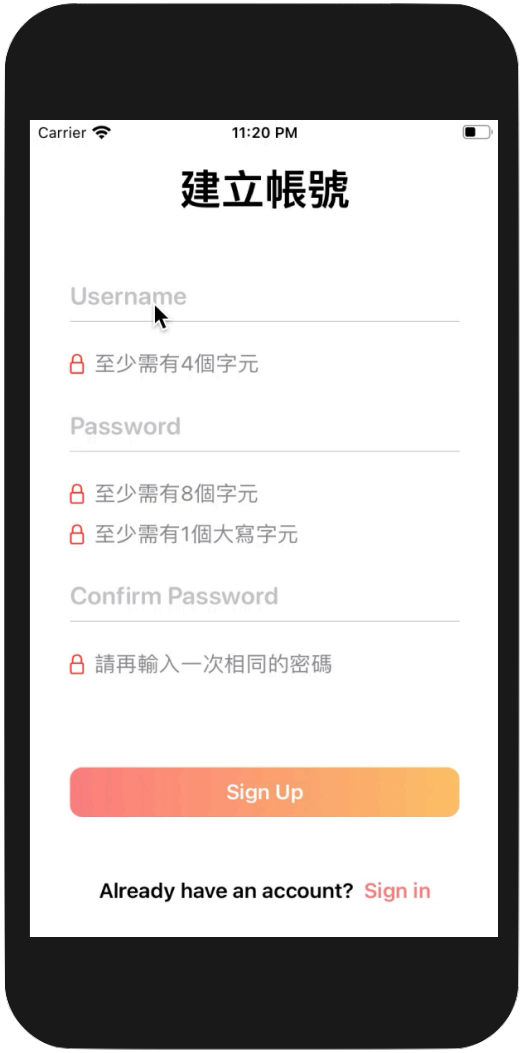
- DragGesture()

拖著跑

```
@State private var currentPosition: CGSize = .zero
@State private var newPosition: CGSize = .zero

    var body: some View {
            Image(imageNames[currentImage])
                .resizable()
                .frame(width: 400, height: 400, alignment: .center)
                .scaleEffect(isPressed ? 0.5:2.0)
                .animation(.easeInOut)
                .scaleEffect(isPressed ? 2.0:0.5)
                .animation(.easeInOut)
                .offset(x: self.newPosition.width, y: self.newPosition.height)
                .onTapGesture {
                    self.isPressed.toggle()
                }
                .onLongPressGesture {
                    currentImage = 1 - currentImage
                }
                .gesture(DragGesture()
                            .onChanged { value in
                                self.newPosition = CGSize(
                                    width: value.translation.width + self.currentPosition.width,
                                    height: value.translation.height + self.currentPosition.height)
                                    }
                            .onEnded { value in
                                self.newPosition = CGSize(
                                    width: value.translation.width + self.currentPosition.width,
                                    height: value.translation.height + self.currentPosition.height)
                                self.currentPosition = self.newPosition
                            }
                )
    }
```

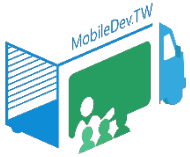
使用者輸入驗證

- 確認使用者輸入是否合乎該欄位要求
  - 使用者帳號超過4個字元
  - 密碼超過8個字元
  - 密碼裡面至少要有1個大寫英文字
  - 再輸入一次相同的密碼進行確認



FormFieldView.swift

```
import SwiftUI

struct FormFieldView: View {
    var fieldName = ""
    @Binding var fieldValue:String
    var isSecure = false
    var body: some View {
        VStack{
            if isSecure{
                SecureField(fieldName, text:$fieldValue)
                    .font(.system(size: 20, weight:.semibold, design:.rounded))
                    .padding(.horizontal)
            }else{
                TextField(fieldName, text:$fieldValue)
                    .font(.system(size: 20, weight:.semibold, design:.rounded))
                    .padding(.horizontal)
            }
            Divider()
                .frame(height: 1)
                .background(Color.init(red: 240/255, green: 240/255, blue: 240/255))
                .padding(.horizontal)
        }
    }
}
```



RequirementTextView.swift

```
import SwiftUI

struct RequirementTextView: View {

    var iconName = "xmark.square"
    var iconColor = Color(red: 251/255, green: 128/255, blue: 128/255)

    var text = ""
    var isStrikeThrough = false

    var body: some View {
        HStack {
            Image(systemName: iconName)
                .foregroundColor(iconColor)
            Text(text)
                .font(.system(.body, design: .rounded))
                .foregroundColor(.secondary)
                .strikethrough(isStrikeThrough)
            Spacer()
        }
    }
}
```

複製檔案

- UserRegistrationModel.swift
  - 負責因應使用者輸入進行判斷

https://gist.github.com/ryanchung403/22e4f0d9973821d7e228f3cc56e88f29

ContentView

- 改採用UserRegistrationViewModel方式

```
import SwiftUI

struct ContentView: View {

    /*
    @State private var userName = ""
    @State private var password = ""
    @State private var passwordConfirm = ""
    */

    @ObservedObject private var userRegistrationViewModel =
                                UserRegistrationViewModel()
```

ContentView

- 標題、FormFieldView、RequirementText

```
var body: some View {
        VStack{
            Text("建立帳號")
                .font(.largeTitle)
                .bold()
                .padding(.bottom, 30)
            FormFieldView(fieldName: "Username",
                          fieldValue: $userRegistrationViewModel.username,
                          isSecure: false)
            RequirementTextView(
                iconName: userRegistrationViewModel.isUsernameLengthValid ? "lock.open":"lock",
                iconColor: userRegistrationViewModel.isUsernameLengthValid ? Color.green : Color.red,
                text: "至少需有4個字元",
                isStrikeThrough: userRegistrationViewModel.isUsernameLengthValid
            )
                .padding()
```

ContentView

- 標題、FormFieldView、RequirementText

```
FormFieldView(fieldName: "Password", fieldValue: $userRegistrationViewModel.password, isSecure: true)
            VStack{
                RequirementTextView(
                    iconName: userRegistrationViewModel.isPasswordLengthValid ?"lock.open":"lock",
                    iconColor: userRegistrationViewModel.isPasswordLengthValid ? Color.green : Color.red,
                    text: "至少需有8個字元",
                    isStrikeThrough: userRegistrationViewModel.isPasswordLengthValid
                )
                RequirementTextView(
                    iconName: userRegistrationViewModel.isPasswordCapitalLetter ?"lock.open":"lock",
                    iconColor: userRegistrationViewModel.isPasswordCapitalLetter ? Color.green :Color.red,
                    text: "至少需有1個大寫字元",
                    isStrikeThrough: userRegistrationViewModel.isPasswordCapitalLetter
                )
            }
            .padding()
```

ContentView

- 標題、FormFieldView、RequirementText

```
FormFieldView(fieldName: "Confirm Password", fieldValue: $userRegistrationViewModel.passwordConfirm,
isSecure: true)
            RequirementTextView(
                iconName: userRegistrationViewModel.isPasswordConfirmValid ?"lock.open":"lock",
                iconColor:userRegistrationViewModel.isPasswordConfirmValid ?Color.green :Color.red,
                text: "請再輸入一次相同的密碼",
                isStrikeThrough: userRegistrationViewModel.isPasswordConfirmValid)
                .padding()
                .padding(.bottom, 50)
```

ContentView

- Button, LinearGradient

```
Button(action:{}){
                Text("Sign Up")
                    .font(.body)
                    .foregroundColor(.white)
                    .bold()
                    .padding(.all, 10)
                    .frame(minWidth: 0, maxWidth: .infinity)
                    .background(
                        LinearGradient(gradient: Gradient(colors:
                          [Color(red: 251/255, green: 128/255, blue: 128/255),
                           Color(red: 253/255, green: 193/255, blue: 104/255)]),
                                       startPoint: .leading, endPoint: .trailing)
                        .cornerRadius(10)
                        .padding(.horizontal)
                    )
            }
```

ContentView

- Button, LinearGradient

```
HStack{
   Text("Already have an account?")
   .font(.body)
   .bold()
   Button(action:{}){
      Text("Sign in")
      .font(.body)
      .bold()
      .foregroundColor(Color(red: 251/255, green: 128/255, blue: 128/255))
    }
}.padding(.top, 50)
Spacer()


        }.padding() //VStack
    } //body
} //ContentView
```

@家族

- @State
  - 在這個View中的各項狀態值
  - 如果在子類別取用需加上$
- @Binding
  - 宣告一個取用父類別變數的變數
- @ObservedObject
  - 可客製化資料型態
  - 可與其他View分享
- @EnvironmentObject
  - 應用程式層級，與所有View分享
- @Published
  - 可與@ObservedObject搭配使用