

iOS 從零開始

Swift基礎語法

蔡智強 Denny Tsai

denny@hpd.io

<https://iosdev.hpd.io>



特色

- Objective C without the C
- 不需要分號
- 減少括號的使用
- Optional 型別接收nil
- 型別安全 (type safe)
- 編譯語言，速度快

開始前的提醒

- 本單元可能有點難啃
- 試著用口語化的方式唸每一段程式碼
- Again, practice makes perfect!

資料型別 (Data Type)

- 整數 Int

1, 4, -15, 20, 0

- 浮點數 Double, Float

0.23534234, -30.34234,
3.1415

- 布林值 Bool

true, false

- 字串 String

"iOS從零開始", "今天天氣真好"

- 陣列 Array

[1, 2, 10, 25, 31, -2]

- 字典 Dictionary

["apple": "蘋果",
"banana": "香蕉"]

變數 (Variable)

- 把記憶體中的一塊取個名字，存放資料使用
- 可以更改的變數使用 var 宣告，不能更改的變數 (或叫常數 Constant) 使用 let 宣告
- 宣告沒有帶初始值或者無法判斷初始值的型別的變數或常數時，需特別標注型別 (type annotation)

變數 (Variable)

```
let a = 10
let b = "Hello"
var x = true
```

```
var currentCondition = ["weather": "不太好", "temperature": "有點冷"]
let fixedDictionary = ["apple": "蘋果", "ios": "很有趣"]
```

```
var gg: Int
var anotherDict: [String: Int]
var aListOfNumbers: [Double]
```

```
let q: Bool
q = false
q = true
```

運算子 (Operator)

- 指派 (=)
- 數值運算 (+, -, *, /, %)
- 複合指派 (+=, -=, *=, /=)
- 比較 (==, !=, >, <, >=, <=)
- 邏輯 (!, ||, %%)
- 區間 (...<, ...)

運算子 (Operator)

```
var a = 5  
a = 6  
a = a + 3
```

```
var c = 10  
c + 15 // 25  
c % 3 // 1
```

```
var x = 10  
x = x + 2 // 12
```

```
var y = 10  
y += 2 // 12
```

```
var k = 15  
k > 0 // true  
k == 10 // false
```

```
var truth = true  
!truth // false  
k > 0 && y < 0 // false  
k > 0 || y == 0 // true
```


整數 (Int) 和浮點數 (Double)

- 用Double而不是Float
- 不同型別的數字不能做運算

- 整數轉換浮點數

```
let x = 10  
let xDouble = Double(x)
```

- 浮點數轉換整數 (小數部分會直接捨棄)

```
let y = 3.1415  
let yInt = Int(y)
```

布林值 (Bool)

- 是非題的概念 true / false
- 使用條件語句的時候就是用布林值來做判斷

```
let a = true
if a {
    print("Hello")
}
```

```
let b = 5
if b > 3 {
    print("b大於3")
}
```

字串 (String)

- 用來儲存文字類的內容
- 使用 + 運算子來連接兩段字串
- 字串插值 (interpolation): 使用 `\(expression)` 來將 `expression` 的值轉換成字串放入

```
var hello = "Hello"  
var world = "World"  
hello + " " + world // "Hello World"
```

```
var hi = "Hi"  
hi += " Denny" // "Hi Denny"
```

```
var number = 10  
"The number is \($number)." // "The number is 10."
```

陣列 (Array)

- 陣列內的元素 (element) 需要是同一個型別的
- 在型別的兩側加上 [] 來宣告陣列型別
- 使用 [index] 來存取元素
- 可以使用常用的運算子來修改陣列
- 使用 count 屬性 (property) 來讀取陣列元素數
- 使用 isEmpty 屬性來檢查陣列是否為空陣列
- 其他陣列的方法 (method): insert(atIndex:), removeAtIndex()

陣列 (Array)

```
var numbers = [1, 2, 3, 3, 10, -20, 8, 4]
var third = 2
numbers[0] // 1
numbers[5] // -20
numbers[third] // 3
numbers.count // 8
numbers[3..  
numbers.count] // [3, 10, -20, 8, 4]

var nothing = [String]()
nothing.isEmpty // true
nothing += ["Hello", "World"] // ["Hello", "World"]
nothing.count // 2
nothing.insert("My", atIndex: 1) // ["Hello", "My", "World"]
nothing.removeAtIndex(0) // ["My", "World"]
```

字典 (Dictionary)

- 一個 key 對一個 value，又叫做 key-value pair
- Key 需要為實作 Hashable 這個協定的型別
- 一般而言，很常使用 String 來當成 Dictionary 的 key
- 使用 [key] 來讀取字典裡面的內容，可以想像成查真的字典
- 字典內的元素並沒有特別的順序，不像陣列有順序

字典 (Dictionary)

```
var dict = [  
    "apple": "蘋果",  
    "banana": "香蕉",  
    "car": "汽車",  
    "dog": "狗",  
]  
  
dict["dog"] // "狗"  
dict.count // 4  
dict.removeValueForKey("dog")  
dict.count // 3  
dict["dog"] // nil  
  
var emptyDict = [String: Int]()  
emptyDict["One"] = 15  
emptyDict["Five"] = "5" // error
```

條件語句 (Conditional Statement)

- if
- if ... else ...
- switch ... case

條件語句 (Conditional Statement)

```
let x = 15
```

```
if x == 0 {  
    print("Number is 0")  
}
```

```
if x > 0 {  
    print("Positive number")  
} else {  
    print("Negative number or 0")  
}
```

```
if x > 0 {  
    print("Positive number")  
} else if x == 0 {  
    print("Number is 0")  
} else {  
    print("Negative number")  
}
```

條件語句 (Conditional Statement)

```
var animal = "dog"

switch animal {
  case "dog":
    print("汪汪")

  case "cat":
    print("喵喵")

  default:
    print("我不知道你怎麼叫")
}
```

```
var thing = "dog"

switch thing {
  case "dog", "cat", "fish":
    print("動物")

  case "denny":
    print("人")

  default:
    print("不明")
}
```

For-in 迴圈

- 適合用來對一個陣列的每一個元素做重複的事情
- 重複n次

```
for number in 1...10 {  
    print("\(number)")  
}
```

```
var names = ["Cindy", "Denny", "Bob", "Michael"]
```

```
for name in names {  
    print(name)  
}
```

```
// old style loop  
for i in 0..  
    print(names[i])  
}
```

While 迴圈

- 一直重複執行一直到條件不被滿足為止
- 注意，容易形成**無限迴圈**

While 迴圈

```
while i < 10 {  
    print("\(i)")  
    i += 1  
}
```

```
var found = false  
var nameToSearch = "Tom"  
var names = ["Jessica", "Jimmy", "Joseph", "Tom", "Alex"]  
var currentIndex = 0
```

```
while !found {  
    if names[currentIndex] == nameToSearch {  
        found = true  
    } else {  
        currentIndex += 1  
    }  
}
```

```
currentIndex
```

迴圈中的指令

- `continue`: 不管迴圈中從這個指令之後的程式碼，直接進行下一個輪迴
- `break`: 在這個點之後直接強制結束迴圈，不管迴圈結束條件有沒有達到

函式 (Function)

- 用來把一段指令做成一個可以重複使用的指令
- 通常會把會一直需要用到功能做成function才不用一直打重複的程式碼
- Function的呼叫方式為 `funcName()`

```
func testFunc() {  
    print("Hello Function!")  
}
```

```
testFunc()
```

函式 (Function)

- 函式可以回傳值，通常會用在計算或者處理資料的函式
- 函式也可以接受輸入的值，叫做parameter
- 宣告方式為

```
func name(param1: Type, param2: Type, ...) -> ReturnType
```

```
func addNumbers(num1: Int, num2: Int) -> Int {  
    return num1 + num2  
}
```

```
addNumbers(1, num2: 2) // 3
```


函式 (Function)

- Parameter可以有預設值，有預設值的parameter在呼叫function的時候就可以不用指定

```
func multiply(num: Int, factor: Int = 2) -> Int {  
    return num * factor  
}
```

```
multiply(3) // 6  
multiply(4, factor: 3) // 12
```

類別 (Class)

- Swift是物件導向的語言，物件是從Class生產出來的
- Class就像是一個工廠，不同的Class負責生產不同的物件 (object)
- Class可以繼承其他的Class，就像是車子工廠可以生產不同型號的車一樣
- 物件有屬性 (property) 跟方法 (method)，屬性就像是物件自己的變數，方法就是函式

類別 (Class)

```
class Car {  
    var wheels = 4  
    var doors = 4  
  
    func drive() {  
        print("driving!")  
    }  
}
```

```
var car = Car()  
car.wheels // 4  
car.doors // 4  
car.drive() // "driving!"
```

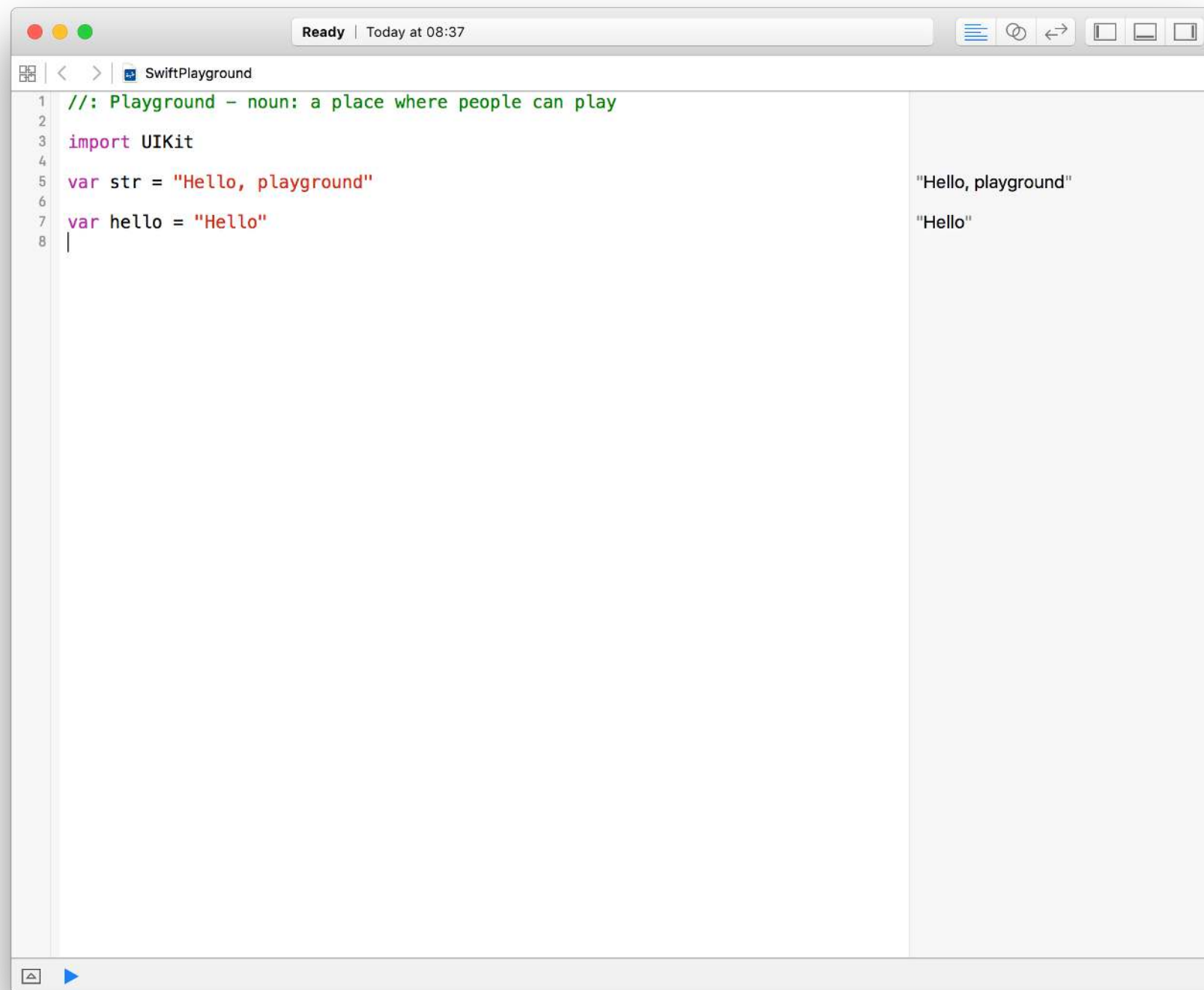
```
class SuperCar: Car {  
}
```

```
var supercar = SuperCar()  
supercar.wheels // 4  
supercar.drive() // "driving!"
```

Swift相關參考資料

- [The Swift Programming Language \(Swift 2.1\)](#)
- [The Swift Programming Language 簡體中文翻譯](#)

Xcode Playground



Xcode Playground

- 提供一個可以直接打程式碼跟看結果的環境
- 適合做一些快速的小測試跟概念確認
- 計算機 (??)
- 適合學Swift !