Swift开发浅析

俞佳兴 计算机科学与技术学院

18888911405

3180103614@zju.edu.cn

Swift 基本语法

常量 变量 函数 数组 字典 控制流 循环 可选类型 Optional 解包 _ Switch 函数与闭包

简单的 App 开发

Piano

Swift 基本语法

常量

```
var weight = 160.4
var age = 18
```

变量

```
let birthday ="2020.02"
let numberOfEyes = 2
```

常见类型

```
Int Double String
```

```
var luckyNumber: Int = 9
```

Swift 基本语法

函数

```
func <#name#>(<#parameters#>) -> <#return type#> {
    <#function body#>
}

func add(a: Int, and b: Int) -> Int {
    return a + b
}

add(a: 12, and: 12)
```

Swift 基本语法

数组

```
var scores: [Int] = [98, 100, 78, 94]
var scores: [String] = ["A", "B", "A", "D"]
var scores: [String] = []
```

增加元素

```
scores += [102]  scores.append(102)
scores.insert(102, at: 0)
```

删除元素

```
scores.remove(at: 0)     scores.removeAll()
scores.removeLast()      scores.removeFirst()
```

Swift 基本语法

字典

```
var identifiers: [Int : String] = [
    1 : "djowmdwom112",
    2 : "somqcnowe293",
]

identifiers.keys : identifiers.values
```

增加元素

```
identifiers[3] = “dpdd2mcnckwc"
```

删除元素

```
identifiers.removeValue(forKey: 1)
```

Swift 基本语法

For 循环

```
for <#item#> in <#items#> {
    <#code#>
}

for item in 0...10 {
    print("\(item)")
}
```

Swift 基本语法

Range

```
0...10

0..<10

0.5...2.5
```

Swift 基本语法

CountableRange

```
0...10      Range✓   CountableRange✓

0..<10      Range✓   CountableRange✓

0.5...2.5   Range✓   CountableRange✖
```

Swift 基本语法

CountableRange

```
for i in stride(from: 0.5, through: 15.25, by: 0.3) {

}
```

through: 闭区间 to: 开区间

Swift 基本语法

While循环

```
while <#condition#> {
    <#code#>
}
```

```
while i > 0 {
    i = i - 1
}
```

Swift 基本语法

Optional

Optional类型的本质是枚举 (enumeration)

```
enum Optional<T> {  // a generic type
    case none
    case some(<T>)
}
```

Swift 基本语法

Optional

none情况: **nil**

? 用来定义Optional类型。

```
var indexOfOneAndOnlyFaceUpCard: Int? ( = nil)
```

! 用来解包Optional类型。

```
let index = cardButtons.index(of: button)!
```

Swift 基本语法

Optional

if 用于解包Optional类型。

```
if let index = cardButtons.index(of: button) { ... }
```

用!定义的Optional类型，当被访问时不需要解包

```
var flipCountIndex: UILabel! ( = nil)
flipCountIndex.text = "Hello"

var flipCountIndex: UILabel? ( = nil)
flipCountIndex!.text = "Hello"
```

Swift 基本语法

Optional

?? 用于给出默认值

```
return emoji[card.identifier] ?? “?"
```

Swift 基本语法

Optional

```
enum Optional<T> {  // a generic type
    case none
    case some(<T>)
}

var hello: String? ( = nil)
var hello: Optional<String> = .none

var hello: String? = “Hello"
var hello: Optional<String> = .some("hello")
```

Swift 基本语法

_ 下划线

表示省略不重要的内容

```
func add(_ a: Int, and b: Int) -> Int {
    return a + b
}

add(12, and: 12)
```

Swift 基本语法

Function 类型

```
var operation: (Double) -> Double

func sqrt(_ a: Double) -> Double {
    return a.squareRoot()
}

operation = sqrt  // 可以正常赋值

let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

```
func changeSign(op: Double) -> Double { return -op }

var operation: (Double) -> Double
operation = changeSign
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

1. 把函数的定义移下来

```
var operation: (Double) -> Double
operation = (op: Double) -> Double { return -op }
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

2. 将 { 提前，并添加in

```
var operation: (Double) -> Double
operation = { (op: Double) -> Double in return -op }
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

3. 系统可以推断返回的类型

```
var operation: (Double) -> Double
operation = { (op: Double) in return -op }
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

4. 省略传入类型

```
var operation: (Double) -> Double
operation = { (op) in return -op }
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

5. 省略返回标记

```
var operation: (Double) -> Double
operation = { (op) in -op }
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

想要函数这样的类型，但是有不想定义一个变量储存。

```
5. 可以用$0,$1,$2...来替代传入参数
var operation: (Double) -> Double
operation = { -$0 }
let result = operation(4.0)
```

Swift 基本语法

Closure 类型

常见的Closure

Array中的map会将Closure应用到数组中的每一个元素

```
let primes = [2.0, 3.0, 5.0, 7.0, 11.0]
let negativePrimes = primes.map({ -$0 }) // [-2.0, -3.0, -5.0, -7.0, -11.0]
let invertedPrimes = primes.map() { 1.0/$0 } // [0.5, 0.333, 0.2, etc.]
let primeStrings = primes.map { String($0) } // [“2.0”,”3.0”,”5.0”,”7.0”,”11.0”]
```

简单的 App 开发

Piano