
无限互联是国内唯一一家**专注**于iPhone和iPad软件开发培训机构，到目前为止为各大公司输送了一大批优秀的iOS高级软件研发人才。随着iOS7系统的发布，我们也在陆续发布**国内首套完整**的iOS开发的视频教程，手把手教您写代码，从入门到熟练再到精通。

**高薪就业**是检验一家培训机构质量的唯一标准，我们的学员高薪就业是对我们最好的肯定，也是我们前进的最强烈的动力，我们感谢同学们的努力，感谢你们对我们的支持！我们也将免费为你们提供最好的就业后的技术支持！

亲爱的同学们，你们的**高薪就业**才是我们最大的成功！

http://www.iphonetrain.com


# 函数与闭包

主讲：汪鸿俊

http://www.iphonetrain.com

無限互联
MOBILE STAR

# 本节内容

- 函数定义与调用
- 函数参数与返回值
- 函数参数名称
- 函数类型
- 函数嵌套
- 闭包表达式
- 尾随闭包
- 值捕获

無限互联 MOBILE STAR

# 函数（Functions）

- 函数（Functions）简介
  - 函数是用来完成特定任务的独立代码块。给一个函数起一个合适的名字，用来标示函数做什么，当函数需要执行的时候，调用这个函数的名字就行。
  - Swift 中的函数语法灵活，可以用来表示任何函数，包括从最简单的没有参数名字的 C 风格函数，到复杂的带局部和外部参数名的 Objective-C 风格函数。参数可以提供默认值，以简化函数调用。参数既可以当做传入参数，也当做传出参数，也就是说，一旦函数执行结束，传入的参数值可以被修改
  - 在 Swift 中，每个函数都有一种类型，包括函数的参数值类型和返回值类型。你可以把 函数类型当做 任何其他普通变量类型 一样处理，这样就可以更简单地把函数当做别的函数的参数，也可以从其他函数中返回函数。函数的定义可以写在其他函数定义中，这样可以在嵌套函数范围内实现功能封装
- 函数的定义与调用
  - 函数定义一般格式为：func 函数名(形参: 形参类型) -> 返回类型
  - 函数调用一般格式为：函数名(实参)

# 函数参数与返回值

- 函数参数与返回值
  - 多重输入参数。函数可以有多个输入参数，写在圆括号中，用逗号分隔
    - func 函数名(形参1: 形参类型, 形参2: 形参类型) -> 返回类型
  - 无参函数。函数可以没有参数，但不能缺少后面的圆括号
    - func 函数名() -> 返回类型
  - 无返回值函数。函数可以没有返回值
    - func 函数名()
  - 多重返回值函数。你可以用元组让多个值作为一个复合值从函数中返回
    - func 函数名(形参: 形参类型) -> (元组值对象: 值类型, 元组值对象: 值类型)

無限互联 MOBILE STAR

# 函数参数名称

- 函数参数名称
  - Swift 的函数参数名称分两种，局部参数名 与 外部参数名
- 局部参数名
  - 局部参数名 是指参数名称只能在函数体中使用，不能在函数调用时使用
    - func 函数名(局部参数名: 形参类型)
- 外部参数名
  - 外部参数名 是指可以在函数调用的时候使用 定义函数的时候 定义的参数名称，可以给每个参数除了局部参数名外 再定义一个外部参数名，作用是可以指出各个实参的用途是什么
    - func 函数名(外部参数名 局部参数名: 形参类型)
  - 如果不想写两次参数名，可以使用井号（#）作为参数名前缀，这就告诉 Swift 使用这个参数名作为局部和外部参数名
    - func 函数名(#形参名称: 形参类型)

無限互联 MOBILE STAR

# 默认参数值

- 默认参数值
  - 可以在函数体中为每个参数定义默认值。当默认值被定义后，调用这个函数时可以省略这个参数
    - func 函数名(#形参名称: 形参类型 = 默认值)
- 默认值参数的外部参数名
  - 为了使定义外部参数名更加简单，当未给 带默认值的参数提供外部参数名时，Swift 会自动提供外部名字。此时外部参数名与局部名字是一样的，就像你已经在局部参数名前写了井号（#）一样
    - func 函数名(形参名称: 形参类型 = 默认值)
    - func 函数名(#形参名称: 形参类型 = 默认值)

# 可变参数

- 可变参数
  - 一个可变参数 可以接受一个或多个值。
  - 函数调用时，你可以用可变参数来传入不确定数量的输入参数
  - 通过在变量类型名后面加入（...）的方式来定义可变参数

```
//传入可变参数的值在函数体内当做这个类型的一个数组。例如，一个叫做 numbers 的 Double... 型可变参数，在函数体内可以当做一个叫 numbers 的 Double[] 型的数组常量。
func arithmeticMean(numbers: Double...) -> Double {
    var total: Double = 0
    for number in numbers {
        total += number
    }
    return total / Double(numbers.count)
}
println(arithmeticMean(1, 2, 3, 4, 5))
```

# 常量参数和变量参数

- 常量参数和变量参数
  - 函数参数默认是常量。但是可以通过指定一个或多个参数为变量参数，从而避免自己在函数中定义新的变量。变量参数不是常量，你可以在函数中把它当做新的可修改的副本来使用
  - 通过在参数名前加关键字 var 来定义变量参数

```
//通过在参数名前加关键字 var 来定义变量参数
func alignRight(var string: String, count: Int, pad: Character) -> String {
    let amountToPad = count - countElements(string)
    for _ in 1...amountToPad {
        string = pad + string
    }
    return string
}
let originalString = "hello"
let paddedString = alignRight(originalString, 10, "-")
println("originalString:" + originalString)
println("paddedString:" + paddedString)
```

無限互联 MOBILE STAR

# 输入输出参数（In-Out Parameters）

- 输入输出参数（In-Out Parameters）
  - 变量参数，仅仅能在函数体内被更改。如果想要一个函数可以修改参数的值，并且想要 这些修改在函数调用结束后仍然存在，那么就应该把这个参数定义为输入输出参数（In-Out Parameters）
  - 定义一个输入输出参数时，在参数定义前加 inout 关键字
  - 输入输出参数不能有默认值，而且可变参数不能用 inout 标记。如果你用 inout 标记一个参数，这个参数不能被 var 或者 let 标记。

```
func swapTwoInts(inout a: Int, inout b: Int) {
    let temporaryA = a
    a = b
    b = temporaryA
}
//只能传入一个变量作为输入输出参数
var someInt = 3
var anotherInt = 107
//当传入的参数作为输入输出参数时，需要在参数前加&符，表示这个值可以被函数修改
swapTwoInts(&someInt, &anotherInt)
println("someInt is now \(someInt), and anotherInt is now \(anotherInt)")
```






无限互联网站：　http://www.iphonetrain.com
老师E-mail：　junewhj@qq.com
老师Blog：　http://blog.csdn.net/jaywon

视频讲解过程中如有不妥之处，欢迎大家将信息反馈到我的Email中，我们会努力完善！谢谢各位的支持。

视频持续更新中... 敬请期待！


无限互联是国内唯一一家**专注**于iPhone和iPad软件开发培训机构，到目前为止为各大公司输送了一大批优秀的iOS高级软件研发人才。随着iOS7系统的发布，我们也在陆续发布**国内首套完整**的iOS开发的视频教程，手把手教您写代码，从入门到熟练再到精通。

**高薪就业**是检验一家培训机构质量的唯一标准，我们的学员高薪就业是对我们最好的肯定，也是我们前进的最强烈的动力，我们感谢同学们的努力，感谢你们对我们的支持！我们也将免费为你们提供最好的就业后的技术支持！

亲爱的同学们，你们的**高薪就业**才是我们最大的成功！

http://www.iphonetrain.com

無限互联 MOBILE STAR

# 函数类型（Function Types）

- 函数类型（Function Types）
  - 函数类型（Function Types）是一种数据类型，类似 C语言函数指针、OC语言的 Block
  - 1、定义函数；2、声明函数类型变量或常量；3、给函数类型变量赋值

```
//1、定义函数
func addTwoInts(a: Int, b: Int) -> Int {
    return a + b
}
//2、声明一个叫 mathFunction 的变量，类型是'一个有两个 Int 型的参数并返回一个 Int 型的值的函数'
var mathFunction: (Int, Int) -> Int
//3、给函数类型变量赋值
mathFunction = addTwoInts

//2、3步可以合并
//var mathFunction: (Int, Int) -> Int = addTwoInts    //可以理解成 var a: Int = 10

//类型推导，可以让 Swift 来推测 mathFunction 函数的类型
//var mathFunction = addTwoInts

//4、使用
println("Result: \(mathFunction(2, 3))")
```

# 函数类型

- 函数类型作为参数类型

```
//参数类型为 (Int, Int)
func printMathResult(mathFun: (Int, Int) -> Int, a: Int, b: Int) {
    println("Result: \(mathFun(a, b))")
}
printMathResult(addTwoInts, 3, 5)
```

- 函数类型作为返回类型

```
//这两个函数的类型都是 (Int) -> Int
func stepForward(input: Int) -> Int {
    return input + 1
}
func stepBackward(input: Int) -> Int {
    println("input: \(input)")
    return input - 1
}

//好，有没有晕？😄晕了就休息一下，再看一遍刚刚讲过的内容；没晕就继续了
func chooseStepFunction(backwards: Bool)   ->    (Int) -> Int {
    return backwards ? stepBackward : stepForward   //返回函数类型
}
```

# 嵌套函数（Nested Functions）

- 嵌套函数（Nested Functions）
  - 可以把函数定义在别的函数体中，称作嵌套函数（nested functions）

```
//默认情况下，嵌套函数是对外界不可见的，但是可以被他的封闭函数（enclosing function）来调用。一个封闭函数也可以返回它的某一个嵌套函数，使得这个函数可以在其他域中被使用
func chooseStepFunction(backwards: Bool) -> (Int) -> Int {
    func stepForward(input: Int) -> Int {
        return input + 1
    }
    func stepBackward(input: Int) -> Int {
        return input - 1
    }
    return backwards ? stepBackward : stepForward
}
var currentValue = -4
let moveNearerToZero = chooseStepFunction(currentValue > 0)
println(moveNearerToZero(10))
```






无限互联网站：　http://www.iphonetrain.com
老师E-mail：　junewhj@qq.com
老师Blog：　http://blog.csdn.net/jaywon

视频讲解过程中如有不妥之处，欢迎大家将信息反馈到我的Email中，我们会努力完善！谢谢各位的支持。

视频持续更新中... 敬请期待！


无限互联是国内唯一一家**专注**于iPhone和iPad软件开发培训机构，到目前为止为各大公司输送了一大批优秀的iOS高级软件研发人才。随着iOS7系统的发布，我们也在陆续发布**国内首套完整**的iOS开发的视频教程，手把手教您写代码，从入门到熟练再到精通。

**高薪就业**是检验一家培训机构质量的唯一标准，我们的学员高薪就业是对我们最好的肯定，也是我们前进的最强烈的动力，我们感谢同学们的努力，感谢你们对我们的支持！我们也将免费为你们提供最好的就业后的技术支持！

亲爱的同学们，你们的**高薪就业**才是我们最大的成功！

http://www.iphonetrain.com

# 闭包（Closures）

- 闭包表达式（Closure Expressions）
  - 闭包表达式语法一般形式如下：

```
{ (parameters) -> returnType in
    statements
}
```

  - 闭包表达式语法可以使用常量、变量和inout类型作为参数，不提供默认值
  - 元组也可以作为参数和返回值

```
let names = ["Chris", "Alex", "Ewa", "Barry", “Daniella”]
//闭包的函数体部分由关键字in引入。 该关键字表示闭包的参数和返回值类型定义已经完成，闭包函数体即将开始。
var reversed = sort(names, { (s1: String, s2: String) -> Bool in
    return s1 > s2
})
//根据上下文推断类型（Inferring Type From Context）
reversed = sort(names, { s1, s2 in return s1 > s2 } )
```

無限互联 MOBILE STAR

# 闭包表达式

- 单表达式闭包隐式返回
  - 如果闭包函数体只包含一个单一的表达式，那么return关键字可以省略

```
//单行表达式闭包可以通过隐藏return关键字来隐式返回结果
reversed = sort(names, { s1, s2 in s1 > s2 } )
```

- 参数名称缩写
  - Swift 自动为内联函数提供了参数名称缩写功能，可以直接通过$0、$1、$2来顺序调用闭包的参数。
  - 如果在闭包表达式中使用参数名称缩写，可以在闭包参数列表中省略对其的定义，in关键字也可以省略

```
//$0和$1表示闭包中第一个和第二个String类型的参数
reversed = sort(names, { $0 > $1 } )
```

- 运算符函数（Operator Functions）

```
//Swift 的String类型定义了关于大于号 (>) 的字符串实现
reversed = sort(names, >)
```

# 尾随闭包（Trailing Closures）

- 尾随闭包（Trailing Closures）
  - 如果需要将一个很长的闭包表达式(以至于不能在一行中进行书写时)作为最后一个参数传递给函数，可以使用尾随闭包来增强函数的可读性
  - 尾随闭包是一个书写在函数括号之后的闭包表达式，函数支持将其作为最后一个参数调用

```
func someFunctionThatTakesAClosure(closure: () -> ()) {
    // 函数体部分
}
//不使用尾随闭包进行函数调用
someFunctionThatTakesAClosure({
    // 闭包主体部分
})
//使用尾随闭包进行函数调用
someFunctionThatTakesAClosure() {
    // 闭包主体部分
}

reversed = sort(names) { $0 > $1 }
```

無限互联 MOBILE STAR

# 捕获值（Capturing Values）

- 捕获值（Capturing Values）
  - 闭包可以在其定义的上下文中捕获常量或变量。即使定义这些常量和变量的原域已经不存在，闭包仍然可以在闭包函数体内引用和修改这些值
  - Swift最简单的闭包形式是嵌套函数，也就是定义在其他函数的函数体内的函数。 嵌套函数可以捕获其外部函数所有的参数以及定义的常量和变量

```
func makeIncrementor(forIncrement amount: Int)  ->    () -> Int {
    var runningTotal = 0

    //incrementor函数并没有获取任何参数，但是incrementor捕获了当前runningTotal变量的引用，捕获一个引用保证了当makeIncrementor结束时候并不会消失，也保证了当下一次执行incrementor函数时，runningTotal可以继续增加。
    func incrementor() -> Int {
        runningTotal += amount
        return runningTotal
    }
    return incrementor
}
//定义了一个叫做incrementByTen的常量，该常量指向一个每次调用会加10的incrementor函数
let incrementByTen = makeIncrementor(forIncrement: 10)
println(incrementByTen())
println(incrementByTen())
```

# 闭包总结

- 闭包是引用类型
  - 无论将函数/闭包赋值给一个常量还是变量，实际上都是将常量/变量的值设置为对应函数/闭包的引用

```
//这也意味着如果将闭包赋值给了两个不同的常量/变量，两个值都会指向同一个闭包
let alsoIncrementByTen = incrementByTen
println(alsoIncrementByTen())
```






无限互联网站：　http://www.iphonetrain.com
老师E-mail：　junewhj@qq.com
老师Blog：　http://blog.csdn.net/jaywon

视频讲解过程中如有不妥之处，欢迎大家将信息反馈到我的Email中，我们会努力完善！谢谢各位的支持。

视频持续更新中... 敬请期待！


无限互联是国内唯一一家**专注**于iPhone和iPad软件开发培训机构，到目前为止为各大公司输送了一大批优秀的iOS高级软件研发人才。随着iOS7系统的发布，我们也在陆续发布**国内首套完整**的iOS开发的视频教程，手把手教您写代码，从入门到熟练再到精通。

**高薪就业**是检验一家培训机构质量的唯一标准，我们的学员高薪就业是对我们最好的肯定，也是我们前进的最强烈的动力，我们感谢同学们的努力，感谢你们对我们的支持！我们也将免费为你们提供最好的就业后的技术支持！

亲爱的同学们，你们的**高薪就业**才是我们最大的成功！

http://www.iphonetrain.com