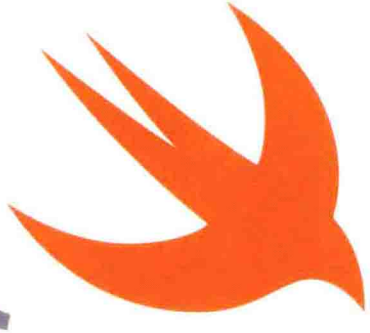
Broadview
www.broadview.com.cn

极客学院
www.jikexueyuan.com



Swift语言

快速入门

极客学院 编著

同步新版官方API文档与语法，确保代码可编译

无缝衔接Objective-C，完美兼容旧代码

语法 | 词法 | 框架覆盖全面，实战案例 | 配套习题丰富

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Swift语言快速入门

极客学院 编著

電子工業出版社
Publishing House of Electronics Industry

内 容 简 介

本书以苹果官方 Swift 英文文档为基础，以其大纲为主线，从易到难全面阐述了 Swift 语言的语言基础、基本运算、字符串操作、集合类型、流程控制、函数与闭包、面向对象、高级运算符操作及语法参考等方方面面。此外，本书内容及 API 已与苹果官方英文文档同步更新，减少了初学者学习旧版语言文档却不能正常编译代码的困惑。

本书涵盖面广，内容全面，不仅适合于 Swift 语言的初学者，有一定 iOS 编程经验的开发者可以把它当作案头工具书。

图书在版编目（CIP）数据

Swift 语言快速入门/极客学院编著.—北京：电子工业出版社，2014.10
ISBN 978-7-121-24328-8

Ⅰ. ①S… Ⅱ. ①极… Ⅲ. ①程序语言—程序设计Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 210620 号

策划编辑：张春雨
责任编辑：刘　舫
印　　刷：北京丰源印刷厂
装　　订：三河市鹏成印业有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　　印张：27　　字数：635 千字
版　　次：2014 年 10 月第 1 版
印　　次：2014 年 10 月第 1 次印刷
定　　价：59.00 元

《Swift 语言快速入门》编委会

主编

陈少佳　　极客学院布道师

李　艺　　木子宁工作室掌门

编委：（排名不分先后）

靳　岩　　梁　杰　　姚尚朗

杨镇林　　帅　印　　张腾飞

Swift 体会

极客学院首席讲师　陈劭家（ime）

我不算是一个果粉，但是我很喜欢苹果的产品，甚至可以说是狂热。2014 年 6 月 2 日晚上我一夜未眠，就是在等苹果的 WWDC，这是开发者的狂欢之夜（或者狂欢之日，因为美国时间是白天）。凌晨 1 点，终于开始了，但是苹果一次又一次的新产品展示让我一次又一次的失望，作为开发者的我表示无感。

我坚持看因为我不相信苹果总是让我失望。最终在 WWDC 的最后一个环节，我真心沸腾了，因为一门新的编程语言——Swift 出现了。要知道在饱受了 Objective-C 语言之苦后看见 Swift 是一种什么心情，你不需要再为引用计数而头疼，不需要再为没有命名空间而不得不添加该死的前缀而烦恼，不需要再为 Objective-C 那难懂的语法而整天晕乎乎的。此外，苹果还为我们提供了 Playground 这个工具，它能够在我们写程序时实时计算出变量的值，甚至还能呈现数字变量的趋势图，让编程富有乐趣及创造性。

由于 Swift 出现得晚，所以它拥有了目前世界上几乎所有编程语言的优点，而没有目前所有编程语言的缺点，Swift 真的太漂亮了。下面我们开始一步一步认识它。

运行速度

从苹果官方给出的数据来看，Objective-C 比 Python 快 2.8 倍，而 Swift 比 Python 快 3.9 倍，可见苹果在 Swift 上下了大量的工夫进行优化。

开发环境

Swift 语言的开发环境是苹果公司提供的集成开发环境 Xcode，可以用来开发 iOS 应用、iOS 游戏、OS X 窗体程序、OS X 游戏、OS X 命令行程序，读者可以直接从 App Store 中搜索并下载。用 Swift 语言可以做到几乎所有 Objective-C 所能做到的事情，所以 Swift 必将取代 Objective-C，

如果你还没有学过 Objective-C 语言，那么恭喜你，不用学了，直接学习 Swift 即可。

运行环境

Swift 被强大的 llvm 编译成机器码，直接运行在系统中。由于 Swift 是苹果的产品，所以目前只支持苹果的系统（OS X 和 iOS）。我们期待会有社区开发出跨平台的 Swift 语言，因为这么好的一门编程语言，应该能够让世界上的每一个人享受到才好，就像 mono 让 C#语言跨平台一样。

语言特性

具有所有现代编程语言的特性，包括：面向对象、类扩展、命名空间、闭包、泛型、强类型、函数有多个返回值等。

这些特性能够大大提高程序员的开发效率，从而为企业节约成本，同时让编程工作充满乐趣。

语法简介

通过前文的介绍，相信读者已经迫不及待地想尝试使用 Swift 了，下面我们就来认识一下 Swift 的语法。

简洁的语法

Swift 抛弃了 Objective-C 那种古板难懂的语法，采用通俗易懂的脚本语言类语法，学过 Python、JavaScript 或者 Lua 语言的读者肯定不会陌生，这大大降低了初学者的学习成本。

变量及常量

如果要定义一个变量 i 等于 1，只需要写 var i = 1，可以看出，不需要指定类型，因为 Swift 会自动做类型推断。

如果要定义一个常量 PI 等于 3.14，只需要写 let PI = 3.14，常量只能被赋值一次。

输出语句

在 Swift 中，可以直接使用 println 函数来输出一段任意类型的信息，如下所示：

```
println("Hello Swift")
```

字符串连接

Swift 语言中的字符串连接同样也非常简单，如果两个值都是字符串，则可直接用加号连接，

如下所示：

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

如果将要连接的值中有其他类型，则直接使用\()包括就可进行连接，如下所示：

```
var hello = "Hello"
var num = 100
var str = "\(hello) \(num)"
```

从上面的示例可以看出，相比 Objective-C 或者 C/C++语言来说，简单太多了。

循环

传统的 C 语言中的 for 循环是这么写的：

```
for (int i = 0; i < 100; i++) {
    //TODO
}
```

在 Swift 中，将循环大大简化了，如下所示：

```
for i in 0..<100{
    //TODO
}
```

具体内容还有很多，详见本书正文。

条件判断

条件判断与 C 语言并没有区别，可以直接书写 C 语言的语法即可通过，如下所示：

```
if (count>5) {
    //TODO
}
```

在 Swift 中还可以再简洁一些，如下：

```
if count>5 {
    //TODO
}
```

另外，在 Swift 语言中，switch…case 语句可以不用加 break 关键字。

函数

如果要定义一个函数用来输出一段信息，则代码如下所示：

```
func sayHello(){
   println("Hello Swift")
}
```

这种写法非常简洁，我曾在 Dart 语言中见过这种写法，func 是一个关键字，用来指明所定义的是一个函数，sayHello 是函数名称，()中是该函数的传入参数。如果还想使用传入参数及返回值，如下所示：

```
func max(a:Int,b:Int)->Int{
   if a>b {
      return a
   }else{
      return b
   }
}
```

该函数名称为 max，可传入两个参数，都是整数类型，参数名称分别为 a、b，函数的返回值也是整数类型。

Swift 中的函数还可以同时返回多个值，如下所示：

```
func getNum()->(Int,Int){
   return (3,4)
}
```

如果想对该函数进行调用并获取到这两个返回结果值，则用法如下：

```
let (a,b) = getNum()
```

面向对象

类的定义非常简单，如下所示：

```
class Hello{
   func sayHello(){
      println("Hello Swift")
   }
}
```

该示例定义了一个名为 Hello 的类，其有一个成员函数名为 sayHello，如果想调用该类及相

应函数，则用法如下：

```
var h = Hello()
h.sayHello()
```

类的继承的写法也非常简单，这一点继承了 C++语言的优秀传统，如果想定义一个名为 Hi 的类继承自 Hello，则写法如下：

```
class Hi:Hello{
}
```

如果后期还想为某类添加功能，有两种方法。第一种是直接修改类的源码添加功能，第二种是为该类写扩展功能，下面我们来着重介绍第二种。如果想为 Hello 类再添加一个名为 sayHi 的方法，则代码如下所示：

```
extension Hello{
    func sayHi(){
        println("Hi Swift")
    }
}
```

其中，extension 关键字表示要扩展已经存在的类的功能，如果想扩展系统或者第三方的某个类的功能而得不到其源码时，采用这种方式将会是一个绝佳的选择。如下所示：

```
extension String{
    func printSelf(){
        println(self)
    }
}
var str = "Hello Swift"
str.printSelf()
```

通过这种方式扩展了系统的 String 类，为其增加了一个 printSelf()方法，在使用时可直接调用目标对象的 printSelf()方法，非常方便。

关于面向对象的更多特性，请看本书正文。

其实 extension 还有另一个用途，那就是模拟命名空间，请看下一个主题。

命名空间

在 Swift 语言中并没有专门的命名空间的关键字，但是可以模拟命名空间这个面向对象的特性，如下所示：

```
//定义命名空间 ime
class ime{
}

//在 ime 命名空间下定义 Hello 类
extension ime{
   class Hello{
      func sayHello(){
         println("Hello Swift")
      }
   }
}
```

使用该类及相关方法的代码如下所示：

```
var h = ime.Hello()
h.sayHello()
```

有没有让你眼前一亮的感觉呢？

结尾

就写到这里吧，想必大家已经对 Swift 有了一个初步的了解，而且迫不及待地想开始学习了，努力吧，少年，你就是未来的太阳！

目　录