



Swift

语言

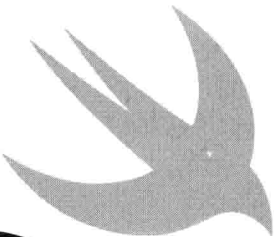
快速入门

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



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战案头工具书。

未经许可, 不得以任何方式复制或抄袭本书之部分或全部内容。
版权所有, 侵权必究。

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
开 本: 787×980 1/16
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

邮编: 100036
印张: 27 字数: 635 千字



凡所购买电子工业出版社图书有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系, 联系及邮购电话: (010) 88254888。

质量投诉请发邮件至 zlts@phei.com.cn, 盗版侵权举报请发邮件到 dbqq@phei.com.cn。

服务热线: (010) 88258888。

《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

第 1 章 马斯堡游记.....	1
1.1 买票.....	1
1.2 导游.....	13
1.3 飞碟.....	22
1.4 斗法.....	23
1.5 题诗.....	27
第 2 章 Swift 初见.....	28
2.1 Hello World.....	28
2.2 简单值.....	29
2.3 控制流.....	30
2.4 函数和闭包.....	33
2.5 对象和类.....	35
2.6 枚举和结构.....	39
2.7 协议和扩展.....	42
2.8 泛型.....	43
第 3 章 语法入门.....	45
3.1 基础.....	45
3.1.1 常量和变量.....	45
3.1.2 注释.....	48
3.1.3 分号.....	49
3.1.4 整数.....	49
3.1.5 浮点数.....	50

3.1.6	类型安全和类型推断	50
3.1.7	数值字面量	51
3.1.8	数值类型转换	52
3.1.9	类型别名	53
3.1.10	布尔值	54
3.1.11	元组	55
3.1.12	可选	56
3.1.13	断言	60
3.2	基本运算符	61
3.2.1	术语	61
3.2.2	赋值运算符	62
3.2.3	算术运算符	62
3.2.4	求余运算符	63
3.2.5	浮点数求余运算符	64
3.2.6	自增和自减运算符	64
3.2.7	单目负号运算符	65
3.2.8	单目正号运算符	65
3.2.9	复合赋值运算符	65
3.2.10	比较运算符	66
3.2.11	三目条件运算符	66
3.2.12	nil 合并运算符	67
3.2.13	区间运算符	68
3.2.14	逻辑运算符	69
3.3	字符串与字符	71
3.3.1	字符串字面量	72
3.3.2	初始化空字符串	73
3.3.3	字符串可变性	73
3.3.4	字符串是值类型	73
3.3.5	使用字符	74
3.3.6	连接字符串和字符	74
3.3.7	字符串插值	75
3.3.8	Unicode.....	75
3.3.9	计算字符数量	77
3.3.10	比较字符串	78

3.4 集合类型	79
3.4.1 集合的可变性	80
3.4.2 数组	80
3.4.3 字典	84
3.5 流程流	89
3.5.1 for 循环	89
3.5.2 while 循环	93
3.5.3 条件语句	96
3.5.4 控制转移语句	103
3.6 函数	108
3.6.1 函数的定义与调用	108
3.6.2 函数参数和返回值	110
3.6.3 函数参数名称	113
3.6.4 函数类型	119
3.6.5 嵌套函数	122
3.7 闭包	122
3.7.1 闭包表达式	123
3.7.2 尾随闭包	126
3.7.3 值捕获	128
3.7.4 闭包是引用类型	130
3.8 枚举	130
3.8.1 枚举语法	131
3.8.2 使用 switch 语句匹配枚举值	132
3.8.3 关联值	133
3.8.4 原始值	135
3.9 类和结构体	136
3.9.1 类和结构体的对比	137
3.9.2 结构体和枚举是值类型	139
3.9.3 类是引用类型	141
3.9.4 类和结构体的选择	142
3.9.5 字符串、数组及字典的赋值和复制行为	143
第 4 章 初级语法	144
4.1 属性	144

4.1.1	存储属性	144
4.1.2	计算属性	147
4.1.3	属性监察器	149
4.1.4	全局变量和本地变量	151
4.1.5	类型属性	151
4.2	方法	155
4.2.1	实例方法	155
4.2.2	类型方法	160
4.3	下标	163
4.3.1	下标语法	163
4.3.2	下标用法	164
4.3.3	下标选项	164
4.4	继承	166
4.4.1	定义一个基类	167
4.4.2	子类生成	168
4.4.3	重写	169
4.4.4	防止重写	173
4.5	构造	173
4.5.1	存储属性的初始赋值	173
4.5.2	自定义构造	174
4.5.3	默认构造器	178
4.5.4	值类型的构造器代理	179
4.5.5	类的继承和构造	180
4.5.6	通过闭包或函数来设置属性的默认值	190
4.6	析构	192
4.6.1	析构原理	192
4.6.2	析构实例	192
4.7	自动引用计数	194
4.7.1	ARC 的工作原理	195
4.7.2	ARC 实践	195
4.7.3	类实例间的强引用环	196
4.7.4	解决类实例间的强引用环	199
4.7.5	闭包产生的强引用环	205
4.7.6	解决闭包产生的强引用环	207

4.8 可选链	210
4.8.1 可选链替代可选强制解析	210
4.8.2 为可选链定义模型类	211
4.8.3 通过可选链获取属性	213
4.8.4 通过可选链调用方法	214
4.8.5 使用可选链获取下标	214
4.8.6 连接多层链	215
4.8.7 可选链中返回可选类型的方法	216
4.9 类型转换	217
4.9.1 定义一个类层次作为例子	217
4.9.2 类型检查	218
4.9.3 向下转换	219
4.9.4 Any 和 AnyObject 的转换	220
4.10 类型嵌套	222
4.10.1 类型嵌套实践	223
4.10.2 引用嵌套类型	224
第 5 章 高级语法	225
5.1 扩展	225
5.1.1 扩展语法	225
5.1.2 计算属性	226
5.1.3 构造器	227
5.1.4 方法	228
5.1.5 可变实例方法	229
5.1.6 下标	229
5.1.7 嵌套类型	230
5.2 协议	231
5.2.1 协议语法	231
5.2.2 属性要求	232
5.2.3 方法要求	233
5.2.4 可变方法要求	234
5.2.5 协议作为类型	235
5.2.6 集合中的协议类型	240
5.2.7 协议的继承	241

5.2.8	协议合成	242
5.2.9	检查协议的一致性	243
5.2.10	可选协议要求	245
5.3	泛型	247
5.3.1	泛型解决的问题	247
5.3.2	泛型函数	248
5.3.3	类型参数	249
5.3.4	命名类型参数	249
5.3.5	泛型类型	250
5.3.6	扩展泛型类型	252
5.3.7	类型约束	253
5.3.8	关联类型	256
5.3.9	where 子句	258
5.4	访问控制	260
5.4.1	模块和源文件	261
5.4.2	访问级别	261
5.4.3	访问控制语法	262
5.4.4	自定义类型	263
5.4.5	子类	265
5.4.6	常量、变量、属性和下标	266
5.4.7	协议	268
5.4.8	扩展	269
5.5	高级运算符	269
5.5.1	位运算符	270
5.5.2	溢出运算符	274
5.5.3	优先级和结合性	276
5.5.4	运算符函数	277
5.5.5	前置和后置运算符	278
5.5.6	组合赋值运算符	279
5.5.7	比较运算符	280
5.5.8	自定义运算符	280
5.5.9	自定义中置运算符的优先级和结合性	281
第 6 章	词法参考	282
6.1	关于词法参考	282

6.2	词法结构	283
6.2.1	空白与注释	283
6.2.2	标识符	283
6.2.3	关键字和标点符号	285
6.2.4	字面量	286
6.2.5	运算符	290
6.3	类型	292
6.3.1	类型标注	292
6.3.2	类型标识符	293
6.3.3	元组类型	293
6.3.4	函数类型	294
6.3.5	数组类型	295
6.3.6	字典类型	296
6.3.7	可选类型	296
6.3.8	隐式解析可选类型	297
6.3.9	合成协议	298
6.3.10	元类型	298
6.3.11	类型继承语法	299
6.3.12	类型推断	299
6.4	表达式	300
6.4.1	前缀表达式	300
6.4.2	二元表达式	301
6.4.3	赋值表达式	303
6.4.4	三元条件运算符	303
6.4.5	类型转换运算符	304
6.4.6	主表达式	305
6.4.7	后缀表达式	310
6.5	语句	314
6.5.1	循环语句	315
6.5.2	分支语句	317
6.5.3	带标签的语句	320
6.5.4	控制传递语句	321
6.6	声明	322
6.6.1	全局代码	323