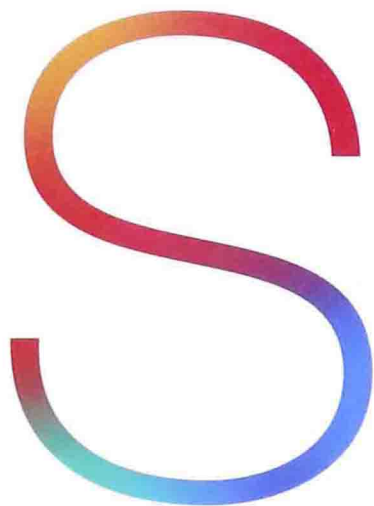




《Swift 语言入门》《Swift 语言进阶》



Swift

开发实战

全面解析了语句、函数、类、协议、闭包等 Swift 基本语法应用
145 个精彩实例，完美展示了 Swift 的开发技术
和控件开发相结合，阐述了 Swift 在 iOS 8 开发中的应用

管蕾 张玲玲 朱元波 编著



人民邮电出版社
POSTS & TELECOM PRESS



一本全面讲解用 Swift 语言开发的图书

S

Swift

开发实战

管蕾 张玲玲 朱元波 编著

人民邮电出版社

北京

图书在版编目 (C I P) 数据

Swift开发实战 / 管蕾, 张玲玲, 朱元波编著. —
北京: 人民邮电出版社, 2014. 10
ISBN 978-7-115-36827-0

I. ①S… II. ①管… ②张… ③朱… 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193129号

内 容 简 介

Swift 是苹果公司在 WWDC2014 大会上发布的一门全新的编程语言, 用于编写 OS X 和 iOS 应用程序。本书共分 24 章, 循序渐进地讲解了 Swift 语言的基本知识及实战技术。本书从 Swift 语言基础讲起, 依次讲解了 Swift 的语法、运算符、字符串和字符、表达式、集合类型、语句和流程控制、函数、闭包、枚举、类、构造函数和析构函数、属性、方法、下标脚本、自动引用计数、泛型、可选链、类型检查和嵌套类型、混编开发、扩展、协议、Sprite Kit 游戏开发等知识。本书几乎涵盖了 Swift 语言的全部内容, 讲解通俗易懂, 特别适合初学者学习。

本书适合 Swift 初学者、iOS 爱好者、iPhone 应用开发人员、iPad 应用开发人员、iOS 应用开发人员学习, 也可以作为相关培训学校和大专院校相关专业的教学用书。

◆ 编 著 管 蕾 张玲玲 朱元波

责任编辑 张 涛

责任印制 彭志环 焦志炜

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路 11 号

邮编 100164 电子邮件 315@ptpress.com.cn

网址 <http://www.ptpress.com.cn>

北京昌平百善印刷厂印刷

◆ 开本: 800×1000 1/16

印张: 31.25

字数: 760 千字

印数: 1—3 000 册

2014 年 10 月第 1 版

2014 年 10 月北京第 1 次印刷



定价: 69.00 元

读者服务热线: (010) 81055410 印装质量热线: (010) 81055316

反盗版热线: (010) 81055315

前言

Swift 是苹果公司在 WWDC2014 大会上所发布的一门全新的编程语言，用于编写 OS X 和 iOS 应用程序。苹果公司在设计 Swift 语言时，就有意让其和 Objective-C 共存，Objective-C 是苹果操作系统在导入 Swift 前使用的编程语言。为了帮助读者迅速掌握 Swift 开发的核心技术知识，笔者特意编写了本书。

Swift 的优势

在 WWDC2014 大会中，苹果公司展示了如何能让开发人员更快进行代码编写及显示得到的“Swift Playground”，在 Swift 编程环境中，左侧输入代码的同时，在右侧可以实时显示结果。苹果公司表示 Swift 是基于 Cocoa 和 Cocoa Touch 专门设计的。Swift 不仅可以用于基本的应用程序编写，如各种社交网络 App，同时还可以使用更先进的“Metal”3D 游戏图形优化工作。由于 Swift 可以与 Objective-C 兼容使用，开发人员可以在开发过程中进行无缝切换。

具体来说，Swift 语言的突出优势如下所示。

(1) 易学。作为一门苹果独立发布的支持型开发语言，Swift 语法简单、使用方便、易学，大大降低了开发者入门的门槛。同时，Swift 语言可以与 Objective-C 混合使用，对于用惯了高难度 Objective-C 语言的开发者来说，Swift 语言更加易学。

(2) 功能强大。Swift 允许开发者通过更简洁的代码来实现更多的内容。在 WWDC2014 发布会上，苹果演示了如何只通过一行简单的代码，完成一个完整图片列表加载的过程。另外，Swift 还可以让开发人员一边编写程序一边预览自己的应用程序，从而快速测试应用在某些特殊情况下的效果。

(3) 提升性能。Swift 语言可以提升程序性能，同时降低开发难度。

(4) 简洁、精良、高效。Swift 是一种非常简洁的语言。与 Python 类似，不必编写大量代码即可实现强大的功能，并且也有利于提高应用开发速度。

(5) 执行速度快。Swift 的执行速度比 Objective-C 更快，这样会在游戏中看见更引人入胜的画面（需要苹果新的 Metal 界面的帮助），而其他应用也会有更好的响应性。

(6) 全面融合。苹果对全新的 Swift 语言的代码进行了大量简化，在更快、更安全、交互更好的同时，开发者可以在同一款软件中同时用 Objective-C、Swift、C 三种语言。

(7) 测试工作更加便捷。方便快捷地测试所编写应用有助于开发者更快地开发出复杂的应用。以往，规模较大的应用编译和测试过程极为冗繁，Swift 能在这一方面带来较大的改进，应用开发者将可以更快地发布经过更彻底测试的应用。

本书特色

本书内容十分丰富，实例内容覆盖全面。我们的目标是通过一本书，提供多本书的价值，在内容的编写上，本书具有以下特色。

(1) 内容讲解循序渐进。本书从基础语法和搭建开发环境讲起，循序渐进地讲解 Swift 语言开发的基本语法知识和核心应用技术。

(2) 结构合理。从用户的实际需要出发，合理安排知识结构，内容由浅入深，叙述清楚。全书详细地讲解了和 Swift 开发有关的所有知识点。

(3) 易学易懂。本书条理清晰、语言简洁，可帮助读者快速掌握每个知识点。读者既可以按照本书编排的章节顺序进行学习，也可以根据自己的需求对某一章节进行有针对性地学习。

(4) 实用性强。本书彻底摒弃枯燥的理论和简单的操作，注重实用性和实战性，通过实例的实现过程，详细讲解各个知识点的具体应用。

(5) 内容全面。本书内容全面，无论是搭建开发环境，还是基本语法、面向对象、函数方法，都能在本书中找到解决问题的答案。

源程序下载地址为：www.toppr.net。

读者对象

iOS 开发初学者

Swift 初学者

大中专院校的老师和学生

毕业设计的学生

iOS 编程爱好者

相关培训机构的老师和学员

从事 iOS 开发的程序员

本书在编写过程中，得到了人民邮电出版社工作人员的大力支持，正是各位编辑的求实、耐心和效率，才使得本书在这么短的时间内出版。另外，也十分感谢我的家人在我写作的时候给予的巨大支持。由于作者水平有限，纰漏和不尽如人意之处在所难免，诚请读者提出意见或建议，以便修订并使之更臻完善。

编 者

目 录

第 1 章 工欲善其事，必先利其器

——Swift 语言基础.....1

- 1.1 Swift 概述.....1
 - 1.1.1 Swift 的创造者.....1
 - 1.1.2 Swift 的优势.....2
- 1.2 搭建开发环境.....3
 - 1.2.1 Xcode 6 介绍.....3
 - 1.2.2 下载并安装 Xcode 6.....5
- 1.3 使用 Xcode 开发环境.....8
 - 1.3.1 改变公司名称.....8
 - 1.3.2 通过搜索框缩小文件范围.....10
 - 1.3.3 格式化代码.....10
 - 1.3.4 代码缩进和自动完成.....11
 - 1.3.5 文件内查找和替换.....12
 - 1.3.6 快速定位到代码行.....14
 - 1.3.7 快速打开文件.....15
 - 1.3.8 使用书签.....16
 - 1.3.9 自定义导航条.....17
 - 1.3.10 使用 Xcode 帮助.....18
 - 1.3.11 调试代码.....19
- 1.4 启动 iOS 8 模拟器.....21

第 2 章 千里之行，始于足下

——Swift 语言基础.....24

- 2.1 第一段 Swift 程序.....24
- 2.2 简单值.....30
- 2.3 流程控制.....31
- 2.4 函数和闭包.....34
- 2.5 对象和类.....36
- 2.6 枚举和结构体.....38
- 2.7 协议和扩展.....40
- 2.8 泛型.....41

第 3 章 新语言，新特性——Swift 的基础

语法.....43

- 3.1 分号.....43
- 3.2 空白.....43
- 3.3 标识符和关键字.....44
 - 3.3.1 标识符.....44
 - 3.3.2 关键字.....45
- 3.4 常量和变量.....45
 - 3.4.1 声明常量和变量.....45
 - 3.4.2 声明变量.....46
 - 3.4.3 输出常量和变量.....49
 - 3.4.4 标注类型.....50
 - 3.4.5 常量和变量的命名规则.....51
- 3.5 注释.....52
 - 3.5.1 注释的规则.....52
 - 3.5.2 使用注释的注意事项.....53
- 3.6 数据类型.....54
 - 3.6.1 数据类型的分类.....55
 - 3.6.2 类型安全和类型推断.....55
 - 3.6.3 类型注解.....56
 - 3.6.4 类型标识符.....56
 - 3.6.5 元组类型.....57
 - 3.6.6 函数类型.....57
 - 3.6.7 数组类型.....58
 - 3.6.8 可选类型.....58
 - 3.6.9 隐式解析可选类型.....59
 - 3.6.10 协议合成类型.....59
 - 3.6.11 元类型.....60
 - 3.6.12 类型继承子句.....60
 - 3.6.13 类型推断.....61
- 3.7 最基本的数值类型.....61
 - 3.7.1 整数.....61
 - 3.7.2 浮点数.....63

3.8	字面量	64	4.7.4	组合逻辑	98
3.8.1	数值型字面量	64	4.7.5	使用括号设置运算优先级	98
3.8.2	整型字面量	65	4.8	位运算符	99
3.8.3	浮点型字面量	66	4.8.1	按位取反运算符	99
3.8.4	文本型字面量	67	4.8.2	按位与运算符	100
3.9	数值型类型转换	68	4.8.3	按位或运算符	101
3.9.1	整数转换	68	4.8.4	按位异或运算符	102
3.9.2	整数和浮点数转换	69	4.8.5	按位左移/右移运算符	103
3.10	类型别名	70	4.9	溢出运算符	107
3.11	布尔值	71	4.10	运算符函数	110
3.12	元组	72	4.10.1	前置和后置运算符	110
3.13	可选类型	73	4.10.2	组合赋值运算符	111
3.13.1	if 语句以及强制解析	74	4.10.3	比较运算符	112
3.13.2	可选绑定	75	4.11	自定义运算符	113
3.13.3	nil	76	4.12	运算符的优先级和结合性	113
3.13.4	隐式解析可选类型	76	第5章	字符串和字符	117
3.14	断言	77	5.1	字符和字符串基础	117
3.14.1	使用断言进行调试	77	5.2	字符串字面量	118
3.14.2	何时使用断言	78	5.3	初始化空字符串	119
第4章	运算符	83	5.4	字符串可变性	120
4.1	运算符概述	83	5.5	字符串是值类型	121
4.2	赋值运算符	84	5.6	字符串遍历	121
4.2.1	基本赋值运算符	84	5.7	计算字符数量	122
4.2.2	复合赋值运算符	85	5.8	连接字符串和字符	123
4.3	算术运算符	86	5.9	字符串插值	125
4.3.1	一元运算符	86	5.10	比较字符串	125
4.3.2	二元运算符	88	5.10.1	字符串相等	125
4.3.3	求余运算	89	5.10.2	前缀/后缀相等	125
4.3.4	浮点数求余计算	90	5.10.3	大写和小写字符串	127
4.4	比较运算符(关系运算符)	91	5.11	国际标准 Unicode	128
4.5	三元条件运算	93	5.11.1	Unicode 术语	128
4.6	区间运算符	94	5.11.2	字符串的 Unicode 表示	128
4.6.1	闭区间运算符	95	5.11.3	UTF-8	129
4.6.2	半闭区间运算符	95	5.11.4	UTF-16	129
4.7	逻辑运算	96	5.11.5	Unicode 标量	129
4.7.1	逻辑非	97	第6章	表达式	132
4.7.2	逻辑与	97	6.1	前缀表达式	132
4.7.3	逻辑或	97			

6.2	二元表达式	132	8.1.3	带标签的语句	172
6.3	赋值表达式	134	8.1.4	控制传递语句	172
6.4	三元条件运算符	135	8.2	for 循环	173
6.5	类型转换运算符	135	8.2.1	for-in	173
6.6	主表达式	136	8.2.2	for 条件递增	175
6.6.1	字符型表达式	136	8.3	while 循环	177
6.6.2	self 表达式	136	8.3.1	while	177
6.6.3	超类表达式	137	8.3.2	do-while	179
6.6.4	闭包表达式	137	8.4	条件语句	181
6.6.5	隐式成员表达式	138	8.4.1	if 语句	181
6.6.6	圆括号表达式	138	8.4.2	switch	182
6.6.7	通配符表达式	139	8.4.3	不存在隐式的贯穿	183
6.7	后缀表达式	139	8.4.4	区间匹配	184
6.7.1	函数调用表达式	139	8.4.5	元组	185
6.7.2	初始化函数表达式	139	8.4.6	值绑定	185
6.7.3	显式成员表达式	140	8.4.7	where	186
6.7.4	后缀 self 表达式	140	8.5	控制转移语句	187
6.7.5	动态表达式	140	8.5.1	continue	187
6.7.6	下标脚本表达式	141	8.5.2	break	188
6.7.7	强制取值表达式	141	8.5.3	贯穿 (Fallthrough)	190
6.7.8	可选链表达式	141	8.5.4	带标签的语句 (Labeled Statements)	191
第 7 章	集合类型	143	第 9 章	函数	193
7.1	数组	143	9.1	函数的分类	193
7.1.1	定义数组	143	9.1.1	从函数定义的角度划分	193
7.1.2	数组构造语句	144	9.1.2	从是否有返回值角度划分	193
7.1.3	访问和修改数组	145	9.1.3	从是否有参数角度划分	194
7.1.4	数组的遍历	148	9.1.4	库函数	194
7.1.5	创建并且构造一个数组	149	9.2	函数的定义	194
7.2	字典	152	9.2.1	无参函数的定义	195
7.2.1	字典字面量	152	9.2.2	有参函数的定义	195
7.2.2	读取和修改字典	153	9.3	函数声明	196
7.2.3	字典遍历	157	9.3.1	函数声明的格式	196
7.2.4	创建一个空字典	159	9.3.2	声明中的参数名	197
7.3	集合的可变性	160	9.3.3	声明中的特殊类型参数	199
第 8 章	语句和流程控制	163	9.4	函数调用	199
8.1	Swift 语句概述	163	9.4.1	调用函数的格式	199
8.1.1	循环语句	163			
8.1.2	分支条件语句	167			

9.4.2	函数调用的方式	201
9.5	函数参数	202
9.5.1	多重输入参数	202
9.5.2	无参函数	203
9.5.3	无返回值函数	203
9.6	返回值	204
9.7	函数参数的名称	205
9.7.1	外部参数名	206
9.7.2	简写外部参数名	206
9.7.3	默认参数值	207
9.7.4	默认值参数的外部参 数名	207
9.7.5	可变参数	208
9.7.6	常量参数和变量参数	208
9.7.7	输入/输出参数	209
9.8	函数类型	210
9.8.1	使用函数类型	211
9.8.2	函数类型作为参数类型	211
9.8.3	函数类型作为返回类型	213
9.9	嵌套函数	214
9.10	函数和闭包	215
9.11	内置库函数	218

第 10 章 闭包 225

10.1	闭包表达式	225
10.1.1	sort 函数	226
10.1.2	闭包表达式语法	227
10.1.3	根据上下文推断类型	229
10.1.4	单表达式闭包隐式 返回	229
10.1.5	参数名称的缩写	230
10.1.6	运算符函数	230
10.2	闭包的简写方式	230
10.3	尾随闭包	231
10.4	捕获值	234
10.5	闭包是引用类型	235

第 11 章 枚举 237

11.1	枚举基础	237
11.2	枚举语法	238

11.3	匹配枚举值和 switch 语句	240
11.4	相关值	241
11.5	原始值	243

第 12 章 类 247

12.1	类和结构体基础	247
12.1.1	定义类和结构体	248
12.1.2	声明结构体字段	249
12.2	类的成员	250
12.2.1	最简单的数据成员	250
12.2.2	最重要的函数成员	251
12.3	结构体成员	252
12.3.1	字段	252
12.3.2	函数	253
12.3.3	属性	253
12.4	类和结构体实例	254
12.5	类的继承	255
12.5.1	类的层次结构	255
12.5.2	继承概述	256
12.5.3	定义子类	257
12.5.4	重写	259
12.5.5	继承规则	263
12.6	属性访问	265
12.7	结构体和枚举是值类型	267
12.8	类是引用类型	268
12.8.1	恒等运算符	269
12.8.2	指针	269
12.9	类和结构体的选择	269
12.10	集合类型的赋值和复制行为	270
12.10.1	字典类型的赋值和拷贝 行为	270
12.10.2	数组的赋值和复制 行为	271
12.10.3	确保数组的唯一性	272
12.10.4	判定两个数组是否共用 相同元素	272
12.10.5	强制复制数组	273

第 13 章 构造函数和析构函数 274

13.1	构造函数概述	274
------	--------	-----

13.1.1	结构体中的构造函数	275	15.1.1	方法的局部参数名称和外部 参数名称	324
13.1.2	类中的构造函数	277	15.1.2	self 属性	325
13.2	构造过程详解	279	15.1.3	在实例方法中修改值 类型	326
13.2.1	为存储型属性赋初 始值	280	15.1.4	在变异方法中给 self 赋值	327
13.2.2	定制化构造过程	281	15.2	类型方法	328
13.2.3	默认构造器	283	第 16 章	下标脚本	331
13.2.4	值类型的构造器代理	284	16.1	下标脚本语法	331
13.2.5	类的继承和构造过程	286	16.2	下标脚本用法	332
13.2.6	通过闭包和函数来设置属性 的默认值	293	16.3	下标脚本选项	333
13.3	析构函数	295	第 17 章	自动引用计数	338
13.3.1	析构过程原理	295	17.1	iOS ARC 的背景	338
13.3.2	析构函数操作	296	17.2	Swift ARC 的工作机制	338
13.4	综合演练	298	17.3	自动引用计数实践	339
第 14 章	属性	308	17.4	类实例之间的循环强引用	341
14.1	属性的种类	308	17.5	解决实例之间的循环强引用	345
14.2	常用的声明属性	308	17.5.1	弱引用	345
14.3	类型属性	310	17.5.2	无主引用	351
14.4	存储属性	310	17.5.3	无主引用以及隐式 解析可选属性	353
14.4.1	常量和存储属性	311	17.6	闭包引起的循环强引用	358
14.4.2	懒惰存储属性	311	17.7	解决闭包引起的循环强引用	362
14.4.3	延迟存储属性	312	17.7.1	定义捕获列表	363
14.4.4	存储属性和实例变量	313	17.7.2	弱引用和无主引用	363
14.5	计算属性	314	第 18 章	泛型	365
14.5.1	基本操作	314	18.1	泛型所解决的问题	365
14.5.2	setter 声明的简略 写法	314	18.2	泛型函数	367
14.5.3	便捷 setter 声明	315	18.3	类型参数	368
14.5.4	只读计算属性	315	18.4	命名类型参数	371
14.6	属性监视器	318	18.5	泛型类型	371
14.7	全局变量和局部变量	319	18.6	类型约束	374
14.8	类型属性	320	18.6.1	类型约束语法	374
14.8.1	类型属性语法	320	18.6.2	类型约束行为	374
14.8.2	获取和设置类型属性 的值	321	18.7	关联类型	376
第 15 章	方法	323	18.7.1	关联类型行为	376
15.1	实例方法	323			

18.7.2	扩展一个存在的类型 为一指定关联类型	378	21.2.1	调用简单的 C 函数	399
18.8	where 语句	378	21.2.2	增加一个 C 键盘输入 函数	403
第 19 章	可选链	380	21.3	Swift 调用 C 函数的综合演练	405
19.1	可选链可替代强制解析	380	第 22 章	扩展	409
19.2	为可选链定义模型类	381	22.1	扩展语法	409
19.3	通过可选链调用属性	383	22.2	计算型属性	410
19.4	通过可选链调用方法	383	22.3	构造器	411
19.5	使用可选链调用子脚本	383	22.4	扩展方法	412
19.6	连接多层链接	384	22.5	下标	413
19.7	链接可选返回值的方法	385	22.6	嵌套类型	415
第 20 章	类型检查和嵌套类型	386	22.7	扩展字符串截取方法	416
20.1	定义一个类层次作为例子	386	22.7.1	演示扩展字符串的 用法	416
20.2	检查类型 (Checking Type)	387	22.7.2	使用 Swift String 扩展 截取字符串	417
20.3	向下转型 (Downcasting)	388	22.8	在 iOS 项目中使用扩展	419
20.4	Any 和 AnyObject 的类型 转换	389	22.9	用 Swift 编写 UITextField 扩展	430
20.4.1	AnyObject 类型	389	第 23 章	协议	438
20.4.2	Any 类型	390	23.1	协议的语法	438
20.5	嵌套类型	391	23.2	对属性的规定	440
20.5.1	嵌套类型实例	391	23.3	对方法的规定	441
20.5.2	嵌套类型的引用	392	23.4	对突变方法的规定	442
第 21 章	混编开发	393	23.5	协议类型	443
21.1	在同一个工程中使用 Swift 和 Objective-C	393	23.6	委托模式	444
21.1.1	Mix and Match 概述	393	23.7	在扩展中添加协议成员	446
21.1.2	在同一个应用的 target 中 导入	394	23.8	通过扩展补充协议声明	447
21.1.3	在同一个框架的 target 中 导入	395	23.9	集合中的协议类型	448
21.1.4	导入外部框架	396	23.10	协议的继承	448
21.1.5	在 Objective-C 中使用 Swift	396	23.11	协议合成	449
21.1.6	实战演练 Swift 与 Objective-C 的相互 调用	397	23.12	检验协议的一致性	449
21.2	Swift 调用 C 函数	399	23.13	对可选协议的规定	451
			第 24 章	Sprite Kit 游戏开发	453
			24.1	Sprite Kit 框架基础	453
			24.1.1	Sprite Kit 的优点和 缺点	453

24.1.2	Sprite Kit、Cocos2D、 Cocos2D-X 和 Unity 的 选择	453	24.3.1	新建工程	462
24.2	开发一个 Flappy Bird 游戏	454	24.3.2	载入有云的天空的动画	462
24.3	开发一个 iPad 飞行游戏 (基于 Swift)	462	24.3.3	实现二维飞行界面	468
			24.3.4	生成发射子弹视图	468
			24.3.5	定义敌人类	469
			24.4	开发一个方块游戏	471

第1章 工欲善其事，必先利其器

——Swift 语言基础

Swift 是苹果公司在 WWDC2014 大会上所发布的一门全新的编程语言，用于编写 OS X 和 iOS 应用程序。苹果公司在设计 Swift 语言时，就有意让其与 Objective-C 共存，Objective-C 是苹果操作系统在导入 Swift 前使用的编程语言。本章将带领大家初步认识 Swift 这门神奇的开发语言，让读者为学习本书后面的知识打下基础。

1.1 Swift 概述

Swift 是一种为开发 iOS 和 OS X 应用程序而推出的全新编程语言，是建立在 C 语言和 Objective-C 语言基础之上的，并且没有 C 语言的兼容性限制。Swift 采用安全模型的编程架构模式，并且使整个编程过程变得更容易、更灵活、更有趣。另外，Swift 完全支持市面上的主流框架——Cocoa 和 Cocoa Touch，这为开发人员重新构建软件和提高开发效率带来了巨大的帮助。本节将带领大家一起探寻 Swift 的诞生历程。

1.1.1 Swift 的创造者

苹果公司的 Swift 语言的创造者是苹果开发者工具部门总监 Chris Lattner（1978 年出生）。Chris Lattner 是 LLVM 项目的主要发起人与作者之一，也是 Clang 编译器的作者。Chris Lattner 曾经开发了 LLVM，这是一种用于优化编译器的基础框架，能将高级语言转换为机器语言。LLVM 极大地提高了高级语言的效率，Chris Lattner 也因此获得了首届 SIGPLAN 奖。

2005 年，Chris 加入 LLVM 开发团队，正式成为苹果的一名员工。在苹果公司的 9 年间，他由一名架构师一路升职为苹果开发者工具部门总监。目前，Chris Lattner 主要负责 Xcode 项目，这也为 Swift 的开发提供了灵感。

Chris Lattner 从 2010 年 7 月才开始开发 Swift 语言，当时它在苹果内部属于机密项目，只有很少人知道这一语言的存在。Chris Lattner 在个人博客上称，Swift 的底层架构大多是他自己开发完成的。2011 年，其他工程师开始参与项目开发，Swift 也逐渐获得苹果内部重视，直到 2013 年成为苹果主推的开发工具。

Swift 的开发结合了众多工程师的心血，包括语言专家、编译器优化专家等，苹果其他团队

也为改进产品提供了很大帮助。同时 Swift 也借鉴了其他语言（如 Objective-C、Rust、Ruby 等）的优点。

Swift 语言的核心吸引力在于 Xcode Playgrounds 功能和 REPL，它们使开发过程具有更好的交互性，也更容易上手。Playgrounds 在很大程度上受到了 Bret Victor 的理念和其他互动系统的启发。同样，具有实时预览功能的 Swift 使编程变得更简单，学习起来也更加容易，目前已经引起了开发者的极大兴趣。这有助于苹果吸引更多的开发者，甚至将改变计算机科学的教学方式。图 1-1 是 Chris Lattner 在 WWDC2014 大会上对 Swift 进行演示。

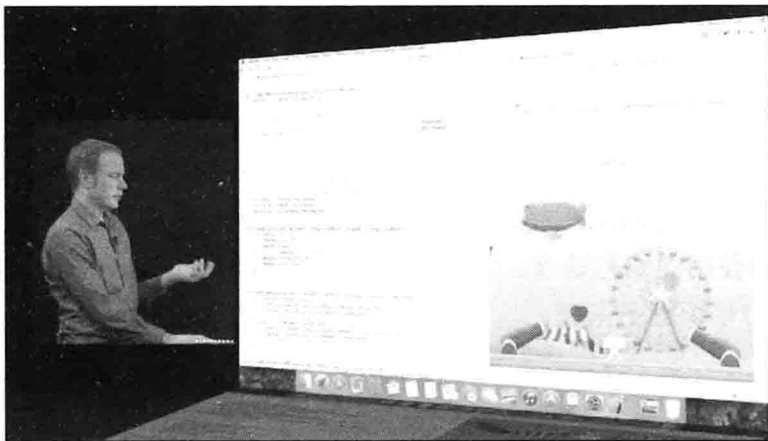


图 1-1 Chris Lattner 在 WWDC2014 大会上对 Swift 进行演示

1.1.2 Swift 的优势

在 WWDC2014 大会中，苹果公司展示了如何能让开发人员更快进行代码编写及显示结果的“Swift Playground”，在左侧输入代码的同时，可以在右侧实时显示结果。苹果公司表示，Swift 是基于 Cocoa 和 Cocoa Touch 专门设计的。由于 Swift 可以与 Objective-C 兼容使用，因此，开发人员可以在开发过程中进行无缝切换。

具体来说，Swift 语言的突出优势如下所示。

(1) 易学。

作为一门苹果独立发布的支持型开发语言，Swift 语言的语法内容混合了 Objective-C、JavaScript 和 Python，其语法简单、使用方便、易学，大大降低了开发者入门的门槛。同时 Swift 语言可以与 Objective-C 混合使用，对于用惯了高难度 Objective-C 语言的开发者来说，Swift 语言更加易学。

(2) 功能强大。

Swift 允许开发者通过更简洁的代码来实现更多的内容。在 WWDC2014 发布会上，苹果演示了如何只通过一行简单的代码，完成一个完整图片列表加载的过程。另外，Swift 还可以让开发人员一边编写程序，一边预览自己的应用程序。

(3) 提升性能。

Swift 语言可以提升程序性能，并同时降低开发难度，没有开发者不喜欢这样的编程语言。

(4) 简洁、精良、高效。

Swift 是一种非常简洁的语言。与 Python 类似，不必编写大量代码即可实现强大的功能，并且

也有利于提高应用开发速度。Swift 可以更快捷有效地编译出高质量的应用程序。

(5) 执行速度快。

Swift 的执行速度比 Objective-C 更快，这样会在游戏中看见更引人入胜的画面（需要苹果新的 Metal 界面的帮助），而其他应用也会有更好的响应性。

(6) 全面融合。

苹果对全新的 Swift 语言的代码进行了大量简化，在更快、更安全、更好的交互、更现代的同时，开发者们可以在同一款软件中同时用 Objective-C、Swift、C3 种语言，这样便实现了 3 类开发人员的完美融合。

1.2 搭建开发环境

都说“工欲善其事，必先利其器”，这一说法在编程领域同样行得通，学习 Swift 开发也离不开好的开发工具的帮助。在本节中，将详细讲解搭建 Swift 语言开发环境的基本知识。

1.2.1 Xcode 6 介绍

要开发 iOS 的应用程序，需要一台安装有 Xcode 工具的 Mac OS X 电脑。Xcode 是苹果提供的开发工具集，它提供了项目管理、代码编辑、创建执行程序、代码调试、代码库管理和性能调节等功能。这个工具集的核心就是 Xcode 程序，提供了基本的源代码开发环境。

Xcode 是一款强大的专业开发工具，可以简单、快速而且以我们熟悉的方式执行绝大多数常见的软件开发任务。相对于创建单一类型的应用程序所需要的能力而言，Xcode 要强大得多，它的设计目的是使我们可以创建任何想象到的软件产品类型，从 Cocoa 及 Carbon 应用程序，到内核扩展及 Spotlight 导入器等各种开发任务，Xcode 都能完成。Xcode 独具特色的用户界面可以帮助我们以各种不同的方式来漫游工具中的代码，并且可以访问工具箱下面的大量功能，包括 GCC、javac、jikes 和 GDB，这些功能都是制作软件产品所需要的。它是一个由专业人员设计又由专业人员使用的工具。

由于能力出众，Xcode 已经被 Mac 开发者社区广为采纳。而且随着苹果电脑向基于 Intel 的 Macintosh 迁移，转向 Xcode 变得比以往任何时候更加重要。这是因为使用 Xcode 可以创建通用的二进制代码，这里所说的通用二进制代码是一种可以把 PowerPC 和 Intel 架构下的本地代码同时放到一个程序包的执行文件格式。事实上，对于还没有采用 Xcode 的开发人员，转向 Xcode 是将应用程序连编为通用二进制代码的第一个必要的步骤。

Xcode 的官方地址是 <https://developer.apple.com/xcode/downloads/>，如图 1-2 所示。

截止到 2014 年 6 月，市面上最主流的版本是 Xcode 5，最新版本是为 Swift 语言推出的 Xcode 6 beta。Xcode 6 beta 具有以下几个最突出的特点。

(1) Xcode 6 增加了一个全新的 iOS 模拟器，允许开发者根据设备调整应用尺寸，除了“Resizable iPhone”和“Resizable iPad”之外，还包括 iPhone 5/5S、iPad 2/Retina/Air 等具体设备，如图 1-3 所示。



图 1-2 Xcode 的官方地址

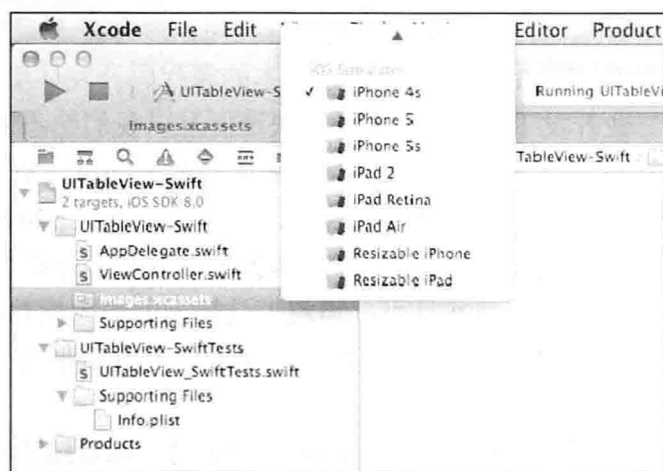


图 1-3 全新的 iOS 模拟器

(2) 完全支持 Swift 编程。Xcode 6 为开发者引入了一种全新的设计和开发应用的方式，深度支持 Swift 编程，开发者不仅能使用 100% 的 Swift 代码来创建一款崭新的应用，还可以向已存在的应用添加 Swift 代码或框架，并在 Swift 或 Objective-C 中查看文档。“Jump to Definition”、“Open Quickly”等在 Swift 中均能很好地工作，甚至 Objective-C 的头定义在 Swift 语法中也能良好地呈现。

(3) 实时的代码效果预览。现在，开发者在使用 Interface Builder 设计界面时，能够实时地预览代码效果。当程序运行时，自定义对象将在设计时展现。当开发者修改自定义视图代码时，Interface Builder 的设计画布则会自动更新，而无需任何的构建和运行操作。

此外，其所包含的 API 还支持向 IB Inspector 添加参数来快速修改视图，甚至开发者还可以预先填充示例数据视图来让界面更加准确。而支持 UIKit 大小类的 iOS 脚本则能够让开发者为所有 iOS

设备开发单一的通用脚本，不仅能为特定的设备尺寸或方向进行行为选择，还可以保持接口的一致性，且易于维护。

(4) 新增 View 调试功能。Xcode 6 实现了此前备受开发者期待的 View Debugger。现在，调试应用 UI 就像单击那样简单，开发者可以轻而易举地看到为什么一个视图可能会被裁剪或隐藏，并在 Inspector 中检查和调试约束及其他参数。当然，Xcode 还包含了其他新的调试工具，比如调试 Gauge 来监控 I/O 用法、增强版的 iCloud Gauge 等，而 Debug Navigator 也将显示更有用的信息，包括栈框架记录和块队列等。

1.2.2 下载并安装 Xcode 6

其实对于初学者来说，只需安装 Xcode 即可。通过使用 Xcode，既能开发 iPhone 程序，也能够开发 iPad 程序。并且 Xcode 还是完全免费的，通过它提供的模拟器就可以在计算机上测试 iOS 程序。但是，如果要发布 iOS 程序或在真实机器上测试 iOS 程序，就需要花 99 美元了。

1. 下载 Xcode

(1) 下载的前提是先注册成为一名开发人员，到苹果开发页面主页 <https://developer.apple.com/>，如图 1-4 所示。



图 1-4 苹果开发页面主页

(2) 登录 Xcode 的下载页面 <https://developer.apple.com/xcode/downloads/>，找到“Xcode 6 beta”选项，如图 1-5 所示。

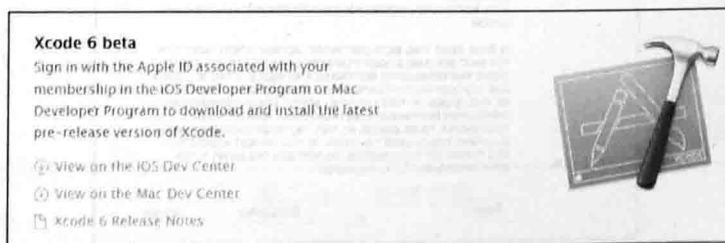


图 1-5 Xcode 的下载页面