TURING 图灵程序设计丛书

Big Nerd Ranch

- Amazon移动开发类畅销书
- 针对Swift 3.0和Xcode 8全新升级
- iOS和macOS开发入门与进阶必读

Swift 编程权威指南

（第2版）

[美] Matthew Mathias John Gallagher 著
陈晓亮 译

Swift Programming
The Big Nerd Ranch Guide, Second Edition

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Swift 编程权威指南（第2版）

[美] Matthew Mathias John Gallagher 著
陈晓亮 译

Swift Programming

The Big Nerd Ranch Guide, Second Edition

人民邮电出版社
北京

图书在版编目（CIP）数据

Swift编程权威指南 : 第2版 / (美) 马修·马赛厄斯 (Matthew Mathias), (美) 约翰·加拉格尔 (John Gallagher) 著 ; 陈晓亮译. -- 北京 : 人民邮电出版社, 2017.6
(图灵程序设计丛书)
ISBN 978-7-115-45746-2

Ⅰ. ①S… Ⅱ. ①马… ②约… ③陈… Ⅲ. ①程序语言－程序设计－指南 Ⅳ. ①TP312-62

中国版本图书馆CIP数据核字(2017)第112833号

内 容 提 要

Big Nerd Ranch 是美国一家专业的移动开发技术培训机构，本书是其培训教材。书中系统讲解了在 iOS 和 macOS 平台上，使用苹果的 Swift 语言开发 iPhone、iPad 和 Mac 应用的基本概念和编程技巧。主要围绕使用 Swift 语言进行 iOS 和 macOS 开发，结合大量代码示例，教会读者利用高级 iOS 和 macOS 特性开发真实的应用。

本书读者对象为 iOS 和 macOS 平台移动开发人员。

◆ 著 [美] Matthew Mathias John Gallagher
译 陈晓亮
责任编辑 杨 琳
责任印制 彭志环
◆ 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
北京市昌平百善印刷厂印刷
◆ 开本：800×1000 1/16
印张：24.25
字数：587千字 2017年6月第1版
印数：1－4 000册 2017年6月北京第1次印刷
著作权合同登记号 图字：01-2017-2563号

定价：89.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

目　录

第一部分　起步

第 1 章　起步 …… 2

1.1　Xcode 起步 …… 2
1.2　尝试 playground …… 4
1.3　修改变量并打印信息到控制台 …… 5
1.4　继续前进 …… 7
1.5　青铜挑战练习 …… 7

第 2 章　类型、常量和变量 …… 8

2.1　类型 …… 8
2.2　常量与变量 …… 9
2.3　字符串插值 …… 11
2.4　青铜挑战练习 …… 12

第二部分　基础知识

第 3 章　条件语句 …… 14

3.1　if/else …… 14
3.2　三目运算符 …… 16
3.3　嵌套的 if …… 17
3.4　else if …… 18
3.5　青铜挑战练习 …… 19

第 4 章　数 …… 20

4.1　整数 …… 20
4.2　创建整数实例 …… 22
4.3　整数操作符 …… 23
4.3.1　整数除法 …… 24
4.3.2　快捷操作符 …… 24
4.3.3　溢出操作符 …… 25
4.4　转换整数类型 …… 26
4.5　浮点数 …… 27
4.6　青铜挑战练习 …… 28

第 5 章　switch 语句 …… 29

5.1　什么是 switch …… 29
5.2　开始使用 switch …… 30
5.2.1　区间 …… 32
5.2.2　值绑定 …… 33
5.2.3　where 子句 …… 34
5.2.4　元组和模式匹配 …… 35
5.3　switch 与 if/else …… 38
5.4　青铜挑战练习 …… 39
5.5　白银挑战练习 …… 40

第 6 章　循环 …… 41

6.1　for-in 循环 …… 41
6.2　类型推断概述 …… 45
6.3　while 循环 …… 45
6.4　repeat-while 循环 …… 46
6.5　重提控制转移语句 …… 47
6.6　白银挑战练习 …… 50

第 7 章　字符串 …… 51

7.1　使用字符串 …… 51
7.2　Unicode …… 53
7.2.1　Unicode 标量 …… 53
7.2.2　标准等价 …… 55
7.3　青铜挑战练习 …… 57
7.4　白银挑战练习 …… 57

第 8 章　可空类型 …… 58

8.1　可空类型 …… 58

8.2 可空实例绑定……60
8.3 隐式展开可空类型……62
8.4 可空链式调用……63
8.5 原地修改可空实例……64
8.6 nil 合并运算符……65
8.7 青铜挑战练习……66
8.8 白银挑战练习……66

第三部分 容器和函数

第 9 章 数组……68
9.1 创建数组……68
9.2 访问和修改数组……69
9.3 数组相等……75
9.4 不可变数组……76
9.5 文档……77
9.6 青铜挑战练习……78
9.7 白银挑战练习……78
9.8 黄金挑战练习……78
第 10 章 字典……79
10.1 创建字典……79
10.2 填充字典……80
10.3 访问和修改字典……80
10.4 增加和删除值……82
10.5 循环……84
10.6 不可变字典……85
10.7 把字典转换为数组……85
10.8 白银挑战练习……86
10.9 黄金挑战练习……86
第 11 章 集合……87
11.1 什么是集合……87
11.2 创建集合……87
11.3 运用集合……89
11.3.1 并集……89
11.3.2 交集……90
11.3.3 不相交……91
11.4 青铜挑战练习……92
11.5 白银挑战练习……92
第 12 章 函数……93
12.1 一个基本的函数……93
12.2 函数参数……94
12.2.1 参数名字……95
12.2.2 变长参数……96
12.2.3 默认参数值……97
12.2.4 in-out 参数……98
12.3 从函数返回……99
12.4 嵌套函数和作用域……100
12.5 多个返回值……101
12.6 可空返回值类型……102
12.7 提前退出函数……103
12.8 函数类型……103
12.9 青铜挑战练习……104
12.10 白银挑战练习……104
12.11 深入学习：Void……105
第 13 章 闭包……106
13.1 闭包的语法……106
13.2 闭包表达式语法……107
13.3 函数作为返回值……110
13.4 函数作为参数……111
13.5 闭包能捕获变量……113
13.6 闭包是引用类型……115
13.7 函数式编程……116
13.8 青铜挑战练习……119
13.9 青铜挑战练习……119
13.10 黄金挑战练习……119

第四部分 枚举、结构体和类

第 14 章 枚举……122
14.1 基本枚举……122
14.2 原始值枚举……125
14.3 方法……128
14.4 关联值……131
14.5 递归枚举……133
14.6 青铜挑战练习……136
14.7 白银挑战练习……136

第 15 章　结构体和类 ……137
15.1　新工程 ……137
15.2　结构体 ……141
15.3　实例方法 ……144
15.4　mutating 方法 ……145
15.5　类 ……145
15.5.1　Monster 类 ……146
15.5.2　继承 ……147
15.6　应该用哪种类型 ……150
15.7　青铜挑战练习 ……150
15.8　白银挑战练习 ……150
15.9　深入学习：类型方法 ……151
15.10　深入学习：mutating 方法 ……152

第 16 章　属性 ……158
16.1　基本的存储属性 ……158
16.2　嵌套类型 ……159
16.3　惰性存储属性 ……160
16.4　计算属性 ……162
16.5　属性观察者 ……164
16.6　类型属性 ……165
16.7　访问控制 ……168
16.8　青铜挑战练习 ……171
16.9　白银挑战练习 ……171
16.10　黄金挑战练习 ……171

第 17 章　初始化 ……172
17.1　初始化方法语法 ……172
17.2　结构体初始化 ……172
17.2.1　结构体的默认初始化方法 ……173
17.2.2　结构体的自定义初始化方法 ……174
17.3　类初始化 ……177
17.3.1　类的默认初始化方法 ……177
17.3.2　初始化和类继承 ……177
17.3.3　类的必需初始化方法 ……183
17.3.4　反初始化 ……184
17.4　可失败的初始化方法 ……185
17.5　掌握初始化 ……188
17.6　白银挑战练习 ……188
17.7　黄金挑战练习 ……188
17.8　深入学习：初始化方法参数 ……189

第 18 章　值类型与引用类型 ……190
18.1　值语义 ……190
18.2　引用语义 ……192
18.3　值类型常量和引用类型常量 ……194
18.4　配合使用值类型和引用类型 ……196
18.5　复制 ……197
18.6　相等与同一 ……199
18.7　我应该用什么 ……200
18.8　深入学习：写时复制 ……201

第五部分　Swift 高级编程

第 19 章　协议 ……210
19.1　格式化表格数据 ……210
19.2　协议 ……214
19.3　符合协议 ……217
19.4　协议继承 ……218
19.5　协议组合 ……219
19.6　mutating 方法 ……220
19.7　白银挑战练习 ……221
19.8　黄金挑战练习 ……221

第 20 章　错误处理 ……222
20.1　错误分类 ……222
20.2　对输入字符串做词法分析 ……223
20.3　捕获错误 ……231
20.4　解析符号数组 ……232
20.5　用鸵鸟政策处理错误 ……236
20.6　Swift 的错误处理哲学 ……239
20.7　青铜挑战练习 ……240
20.8　白银挑战练习 ……240
20.9　黄金挑战练习 ……241

第 21 章　扩展 ……242
21.1　扩展已有类型 ……242
21.2　扩展自己的类型 ……244
21.2.1　用扩展使类型符合协议 ……244
21.2.2　用扩展添加初始化方法 ……245
21.2.3　嵌套类型和扩展 ……246

21.2.4 扩展中的函数 ····· 247
21.3 青铜挑战练习 ····· 248
21.4 青铜挑战练习 ····· 248
21.5 白银挑战练习 ····· 248

第 22 章 泛型 ····· 249

22.1 泛型数据结构 ····· 249
22.2 泛型函数和方法 ····· 251
22.3 类型约束 ····· 253
22.4 关联类型协议 ····· 254
22.5 类型约束中的 where 子句 ····· 257
22.6 青铜挑战练习 ····· 259
22.7 白银挑战练习 ····· 259
22.8 黄金挑战练习 ····· 259
22.9 深入学习：理解可空类型 ····· 260
22.10 深入学习：参数多态 ····· 260

第 23 章 协议扩展 ····· 262

23.1 为锻炼建模 ····· 262
23.2 扩展 Exercise ····· 264
23.3 带 where 子句的协议扩展 ····· 265
23.4 用协议扩展提供默认实现 ····· 266
23.5 关于命名：一个警世故事 ····· 268
23.6 青铜挑战练习 ····· 270
23.7 黄金挑战练习 ····· 270

第 24 章 内存管理和 ARC ····· 271

24.1 内存分配 ····· 271
24.2 循环强引用 ····· 272
24.3 用 weak 打破循环强引用 ····· 276
24.4 闭包中的循环引用 ····· 277
24.5 逃逸闭包和非逃逸闭包 ····· 281
24.6 青铜挑战练习 ····· 283
24.7 白银挑战练习 ····· 283
24.8 深入学习：我能获取实例的引用计数吗 ····· 283

第 25 章 Equatable 和 Comparable ····· 284

25.1 符合 Equatable ····· 284
25.1.1 插曲：中缀运算符 ····· 286
25.1.2 方法“买一赠一” ····· 287
25.2 符合 Comparable ····· 287
25.3 继承 Comparable ····· 289
25.4 青铜挑战练习 ····· 290
25.5 黄金挑战练习 ····· 290
25.6 白金挑战练习 ····· 291
25.7 深入学习：自定义运算符 ····· 291

第六部分 事件驱动的应用

第 26 章 第一个 Cocoa 应用 ····· 296

26.1 开始创建 VocalTextEdit ····· 297
26.2 模型–视图–控制器 ····· 298
26.3 设置视图控制器 ····· 299
26.4 在 Interface Builder 中设置视图 ····· 301
26.4.1 添加朗读和停止按钮 ····· 302
26.4.2 添加文本视图 ····· 303
26.4.3 自动布局 ····· 305
26.5 连接 ····· 307
26.5.1 为 VocalTextEdit 的按钮设置目标–动作对 ····· 307
26.5.2 连接文本视图出口 ····· 308
26.6 让 VocalTextEdit“说话” ····· 310
26.7 保存和加载文档 ····· 311
26.7.1 类型转换 ····· 313
26.7.2 保存文档 ····· 314
26.7.3 加载文档 ····· 316
26.7.4 按照 MVC 模式整理代码 ····· 318
26.7.5 现实世界中的 Swift ····· 320
26.8 白银挑战练习 ····· 320
26.9 黄金挑战练习 ····· 320

第 27 章 第一个 iOS 应用 ····· 321

27.1 开始创建 iTahDoodle ····· 322
27.2 布局用户界面 ····· 323
27.3 为待办事项列表建模 ····· 331
27.4 设置 UITableView ····· 335
27.5 保存和加载 TodoList ····· 337
27.5.1 保存 TodoList ····· 337
27.5.2 加载 TodoList ····· 339

27.6　青铜挑战练习 ……341
27.7　白银挑战练习 ……341
27.8　黄金挑战练习 ……341
第 28 章　互操作 ……342
28.1　一个 Objective-C 工程 ……342
28.2　在 Objective-C 工程中加入 Swift ……351
28.3　添加 Objective-C 类 ……361
28.4　白银挑战练习 ……368
28.5　黄金挑战练习 ……368
第 29 章　结语 ……369
29.1　接下来学习什么 ……369
29.2　插个广告 ……369
29.3　邀请 ……369

第一部分

起　步

这部分介绍编写Swift代码所需的工具链，包括Swift开发者的主要开发工具Xcode，并且使用playground来提供试运行代码的轻量环境。这几章还会帮你熟悉一些Swift最基本的概念，比如常量和变量，以便为学习本书的后续部分和深入理解这门语言打好基础。

第1章 起　步

本章将介绍如何设置环境，并简要了解iOS和macOS开发者日常使用的一些工具。此外，你还会自己动手写些代码以更好地了解Swift和Xcode。

1.1 Xcode起步

如果还没有安装Xcode，可以从App Store下载并安装。确保下载Xcode 8或者更新的版本。

安装完成后，启动Xcode。欢迎画面会给出一些选项，包括Get started with a playground和Create a new Xcode project（如图1-1所示）。

图1-1　Xcode欢迎画面

playground是随着Xcode 6发布的，能为你提供交互环境用来快速开发及运行Swift代码，已经成为了一个实用的原型工具。playground不需要编译运行整个工程，而是随时运行Swift代码，所以非常适合在轻量环境中测试和试验Swift语言。在阅读本书的过程中，你会经常用到playground，从所写的Swift代码快速得到结果。

除了playground之外，后面几章还要创建命令行工具。为什么只用playground不够呢？因为那

样会错过Xcode的很多特性，从而不能充分了解这个IDE。你之后会在Xcode上花费很多时间，所以最好尽早适应。

在欢迎画面选择Get started with a playground。

接下来，把playground命名为MyPlayground。选择平台时（iOS、macOS或tvOS），即使你是iOS开发者也请选择macOS（如图1-2所示）。我们即将用到的Swift特性对两个平台是通用的。点击Next。

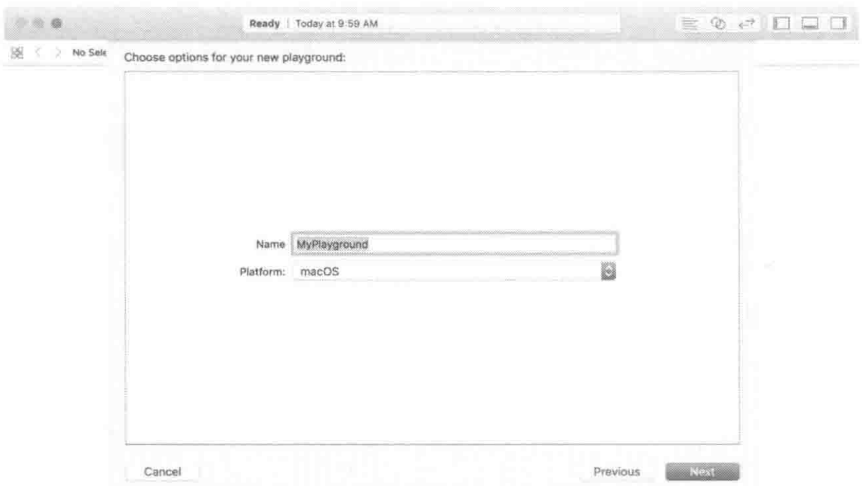

图1-2 命名playground

最后，Xcode会提示保存playground。在阅读本书的过程中，最好把所有的代码放到一个目录中。选择一个合适的路径并点击Create（如图1-3所示）。

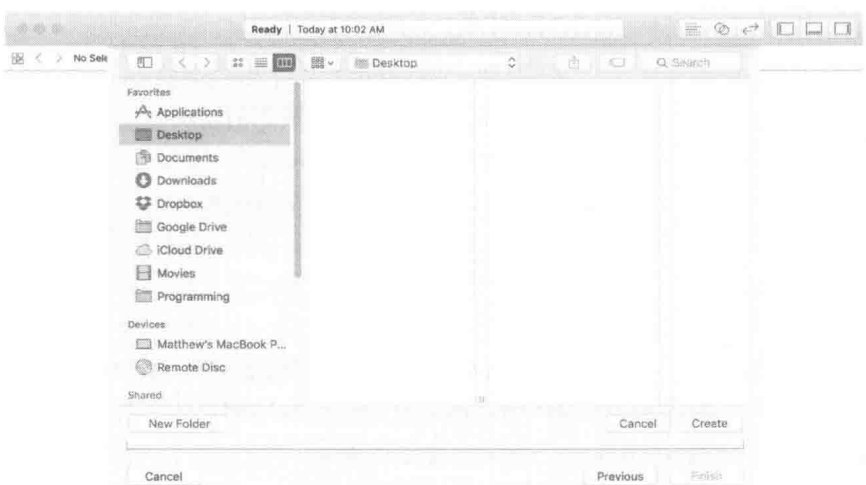

图1-3 保存playground

1.2　尝试 playground

在图1-4中可以看到，Swift的playground打开后有两个区域。左边是Swift代码编辑器，右边是运行结果侧边栏。每次源代码有变化时，就会从上至下运行编辑器中的代码，结果展示在运行结果侧边栏里。

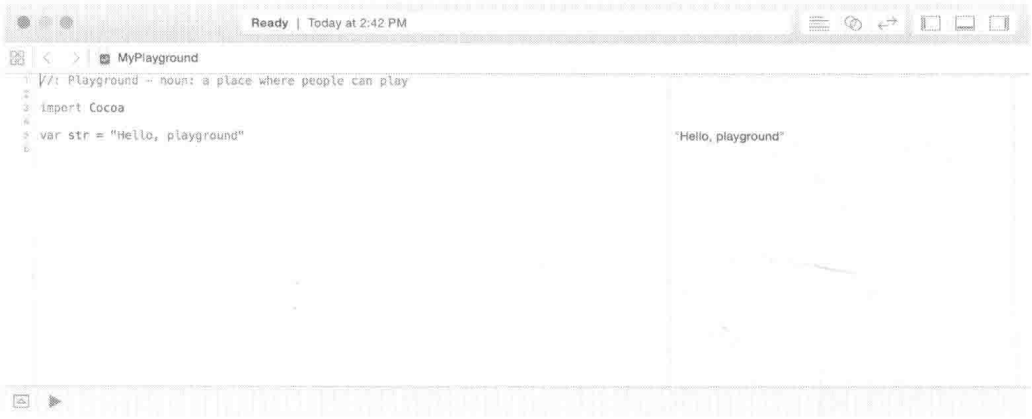

图1-4　新建的playground

来看看新建的playground。注意第一行文本是绿色的，以两个斜杠开头：`//`。斜杠告诉编译器这一行是注释，而Xcode把注释显示为绿色。

开发者可以把注释用作内嵌文档，也可以把它用作笔记来记录这里发生了什么事。斜杠的作用是告诉编译器不要把这一行当作代码。删掉斜杠后编译器会产生一个错误，抱怨自己无法解析表达式。用快捷键Command-/可以很方便地重新加上斜杠（如果是新安装的Xcode，快捷键不起作用的话，重启一下电脑再试试）。

注释下方，playground引入了Cocoa框架。该`import`语句表示playground可以完整访问Cocoa框架中的所有应用程序接口（API）。（API就是说明程序应该如何写的规定或者一组定义。）

`import`语句下面是一行`var str = "Hello, playground"`。引号中的文本被复制到了右边的运行结果侧边栏中：“Hello, playground”。现在仔细看看这行代码。

首先，注意赋值操作符等号。赋值操作符会把它右边的代码结果赋给左边的常量或变量。比如等号左边是文本`var str`。Swift的关键字`var`用来声明变量，下一章会详细介绍这个重要概念。现在，只要知道变量表示你预期会变化的某个值即可。

等号右边是`"Hello, playground"`。Swift中的引号表示字符串（`String`），是字符的有序集合。上面的样板把这个新变量命名为`str`，不过其实几乎可以起任何名字。（当然，也会有限制。试试把`str`改成`var`，看看会发生什么？你觉得为什么不能把变量命名为`var`？请务必把名字改回`str`再往下看。）

现在你能明白右边的运行结果侧边栏中打印的文本是什么了：是赋值给变量`str`的字符串值。

1.3 修改变量并打印信息到控制台

字符串（String）是一种类型，我们将变量str称为“字符串类型的一个实例”。类型是用来表示数据的特殊结构。Swift有很多类型，本书会一一介绍。每种类型都有特定的能力（能对数据做什么）和局限（不能对数据做什么）。比如说，字符串类型旨在处理字符的有序集合，并定义了一系列函数来处理这个字符的有序集合。

回忆一下，str是变量，这意味着你可以改变str的值。我们来给字符串末尾添加一个感叹号，使之成为有标点的完整句子。（在本书中，无论何时添加代码，都会加粗表示；删除则会用删除线表示。）

代码清单1-1 添加标点

```
import Cocoa

var str = "Hello, playground"
str += "!"
```

添加感叹号用到了*加法赋值运算符*+=。加法赋值运算符把加法（+）和赋值（=）运算符组合在了一起。（第3章会详细介绍运算符。）

注意到运算结果侧边栏里的变化了吗？你会看到一行新的结果表示str的新值，它已经补上了感叹号（如图1-5所示）。

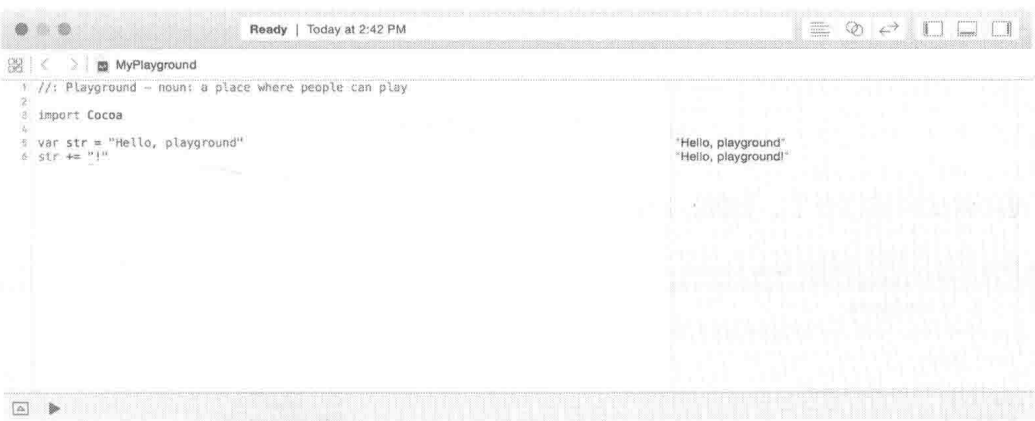

图1-5 修改str

接下来，再加一行代码来把str持有的值打印到控制台。在Xcode中，控制台显示你在程序运行过程中创建并输出为日志的文本消息。当发生警告和错误时，Xcode也会用控制台显示。

使用print()函数可以打印信息到控制台。*函数*是一组相关的代码，指示计算机完成特定的任务。print()是打印一个值到控制台然后换行的函数。跟playground不同的是，Xcode项目没有运行结果侧边栏，所以在写功能完备的应用时会频繁用到print()函数。在检查你关注的某个变量的值时，控制台十分有用。

代码清单1-2　打印到控制台

```
import Cocoa

var str = "Hello, playground"
str += "!"
print(str)
```

敲入这行代码并且等playground执行后，Xcode的底部会自动显示控制台（如果没有，打开调试区域就能看到。如图1-6所示，点击View → Debug Area → Show Debug Area。注意到最后一级菜单的快捷键了吗？也可以通过按下Shift-Command-Y来打开调试区域。）

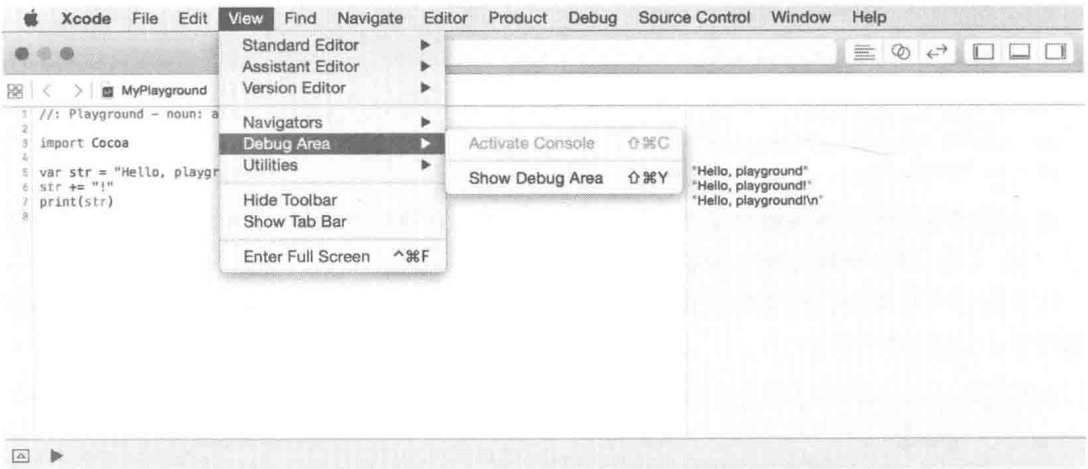

图1-6　显示调试区域

现在调试区域打开了，如图1-7所示。

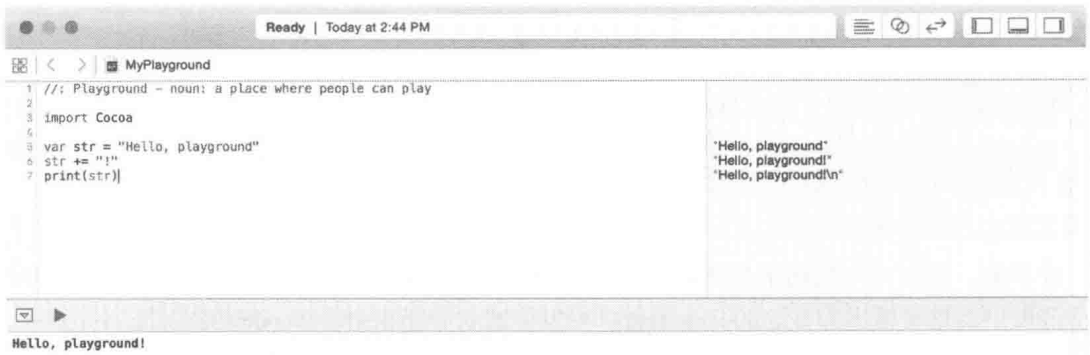

图1-7　第一段Swift代码

1.4 继续前进

现在回顾一下到目前为止的成就，你已经：

- 安装了Xcode；
- 创建并熟悉了playground；
- 使用并修改了变量；
- 知道了字符串类型；
- 用函数打印了一些信息到控制台。

很好！你很快就能开发自己的应用了。坚持下去，随着学习的深入，你会发现本书中几乎所有的内容都不过是基于目前学到的东西稍加变化而来的。

1.5 青铜挑战练习

本书很多章末尾都有一个或多个挑战练习。你要独立完成这些练习以加深对Swift的理解并积攒经验。下面就是你的第一题，不过首先要创建一个新的playground。

你已经知道了字符串类型以及如何用`print()`函数打印信息到控制台。在新的playground里创建一个字符串类型的实例，把实例的值置为你的姓，然后打印到控制台。

第2章

类型、常量和变量 2

本章介绍Swift的基本数据类型、常量和变量。这些元素是构建一切程序的基本构件。常量和变量用来存储值，以及在应用程序中传递数据。类型描述的是常量和变量所持有数据的本质。常量和变量之间、每种数据类型之间都有重要的区别，这些区别形成了它们各自独特的用法。

2.1 类型

变量和常量有数据类型。类型描述数据的本质，并为编译器提供数据处理方式的信息。根据常量或变量的类型，编译器知道该保留多少内存，并且能够进行类型检查（type checking）。这是Swift的一个特性，可以防止你错把其他类型的数据赋值给一个变量。

现在来看看实际操作。创建一个新的macOS playground。（在欢迎画面，选择Get started with a playground；在Xcode内的话，选择File → New → Playground...。）把playground命名为Variables。

假设你想用代码创造一座镇子。你可能想用一个变量表示镇上的红绿灯数量。删除模版自带的代码。创建一个名为numberOfStoplights的变量，给它赋个值，如代码清单2-1所示。（记住，要删除的代码用删除线表示。）

代码清单2-1 赋一个字符串给变量

```
import Cocoa

~~var str = "Hello, playground"~~
var numberOfStoplights = "Four"
```

这里把字符串类型的实例赋值给了numberOfStoplights变量。我们来一步步分析为什么会这样。赋值操作符（=）把右边的值赋给左边。Swift用类型推断（type inference）确定变量的数据类型。在上例中，编译器知道变量numberOfStoplights的类型是字符串，因为赋值操作符右边的值是字符串的实例。为什么"Four"是字符串类型的实例呢？因为引号表示这是字符串字面量。

现在像上一章那样使用+=给变量加上整数2，如代码清单2-2所示。

代码清单2-2 "Four"加2

```
import Cocoa
```

```
var numberOfStoplights = "Four"
numberOfStoplights += 2
```

编译器会报错，告诉你这个操作符不能这样用。报错是因为你要把一个数字和字符串相加，而它们的类型不同。把数字2和字符串相加是什么意思？是把字符串重复变成"FourFour"？还是把2添加到末尾变成"Four2"？没人知道。所以把数字和字符串相加是没有意义的。

如果一开始你就认为numberOfStoplights不应该是字符串类型，那么你是对的。因为这个变量表示虚拟镇子上的红绿灯数量，所以用数字类型更合理。Swift提供整型（Int）类型表示整数，显然更适合这个变量。修改代码，改用整型，如代码清单2-3所示。

代码清单2-3 使用数字类型

```
import Cocoa

var numberOfStoplights = "Four"
var numberOfStoplights: Int = 4
numberOfStoplights += 2
```

现在来看看这里的改动。改动之前，编译器靠类型推断来确定变量numberOfStoplights的数据类型。现在则利用Swift的类型注解（type annotation）语法显式地声明变量是整型类型。上面代码中的冒号表示“……是……类型”，所以这行代码可以这么理解：“声明一个名为numberOfStoplights的变量，其类型是Int，初始值是4。”

注意，即使有了类型注解，也不代表编译器不再关注等号两边的类型。举个例子，如果你试图利用类型注解把前面的字符串实例"Four"声明为整数，编译器会警告你字符串无法转化为整数。

Swift有大量常用数据类型。第7章会深入介绍包含文本数据的字符串，第4章则会介绍数字。其他常用类型还有几种容器类型（collection type），本书后面会讲到。

刚才敲入的新代码还有一个变化：错误消失了。在表示镇上红绿灯数量的整数变量上加2完全没有问题。事实上，因为这个实例被声明为变量，所以这样操作非常自然。

2.2 常量与变量

我们说类型描述的是常量或变量所持有数据的本质，那么什么是常量和变量呢？到目前为止，你只见过变量。变量的值可以变化，这意味着你可以给变量赋新值。通过下面这行代码，可以修改numberOfStopligths的值：numberOfStoplights += 2。

不过你经常需要创建值不变的实例。针对这种情况，可以用常量（constant）。顾名思义，常量的值不能改变。

你把numberOfStoplights声明为变量，并且改变了其值。但是如果不想修改numberOfStoplights的值，应该怎么做呢？这种情况下，最好把numberOfStoplights声明为常量。一条很棒的经验法则是：对必须变化的实例用变量，其他都用常量。