# 几乎所有语言都有的条件判断和循环

Back to series (/series/make-a-decision)

字号
字号
默认主题
金色主题
暗色主题

和所有其它编程语言一样，为了能够控制程序的执行路径，Swift提供了我们熟悉的循环和分支判断语句。它们大多和字面上的直观表意相同，在这一节中，我们就快速过一遍它们的基本用法。

## 条件分支判断语句

第一个要介绍的，是 `if...else if...else...` 。这是几乎每种语言都支持的分支表达方式，其中 `else if` 和 `else` 都是可选的部分，它们可以单独和 `if` 搭配形式各种分支条件的判断。基本上，看到代码，我们就可以直接了解这类判断的含义了。

```
var light = "red"
var action = ""

if light == "red" {
    action = "stop"
}
else if light == "yellow" {
    action = "caution"
}
else if light == "green" {
    action = "go"
}
else {
    action = "invalid"
}
```

在上面这个红绿灯的代码里，我们不断根据 `light` 的值，设置了变量 `action` 的值，它很简单。但通常，我们还是更多会使用 `if...else...` 表示非黑即白这样的简单关系。

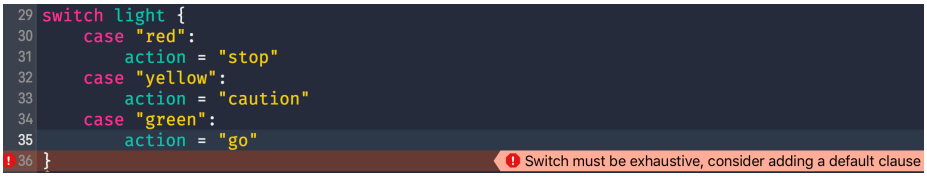
对于上面这种存在多种可能性的情况，在Swift里，我们通常还是会使用 `switch...case...` 来表示，它比 `if...else...` 更安全，也更有更好的表意：

```
switch light {
    case "red":
        action = "stop"
    case "yellow":
        action = "caution"
    case "green":
        action = "go"
    default:
        action = "invalid"
}
```

这里，我们使用 `switch...case...` 表达了和之前的 `if...else...` 相同的语义。但是，它更明确的表达了当 `light` 的值（ `switch` ）为各种情况（ `case` ）时，我们应该采取哪些措施，这样的概念。

但和C++/Java这样语言相比，Swift中的 `switch...case...` 也有一些自己独特的地方：

首先， `case` 语句必须exhausitive，也就是说，必须覆盖 `switch` 后面出现的表达式的所有情况，否则会导致编译错误。例如，我们去掉 `default` ，就会得到下面的错误：




因此，当你不需要对列出 `case` 的其他情况作出处理时，你也要在 `default` 分支写上一句 `break` ，明确表示你考虑到了其他的情况，只是你不需要更多额外处理而已。

其次，每个 case 语句不会自动“贯通”到下一个 case ，因此我们也无需在每个 case 最后一行写 break 表示结束；

最后，当我们要在一个 case 里匹配多个条件的时候，可以使用逗号把多个条件分开，在后面的视频里，我们会看到这个用法。

以上，就是和分支条件相关的两个最基本的场景和用法，接下来，我们了解循环。

## 循环控制语句

第一个要介绍的，是 for element in collection/range ，我们可以用它来方便的遍历一个集合类型或者范围：

```
let vowel = ["a", "e", "i", "o", "u"]

for char in vowel {
    print(char)
}
// aeiou

for number in 1...10 {
    print(number)
}
// 12345678910
```

要说明的是，传统C风格的三段式 for 循环，已经在Swift 3中被移除了：

```
// DO NOT use this style of for loop
// for var i = 0; i < 10; i += 1 {
//    print(i)
// }
```

第二个循环的方式是 while ，它有前置判断和后置判断两种形式，基本上保留了原汁原味的C用法：

```
// while

var i  = 0
while i < 10 {
    print(i)
    i += 1
}

// do ... while
repeat {
    print(i)
    i -= 1
} while i > 0
```

在这两类循环里，我们都可以用 continue 来停止执行当前循环中的语句，立即开始下一次循环。例如，打印所有的偶数：

```
for number in 1...10 {
    if number % 2 != 0 { continue }
    print(number)
}
// 2 4 6 8 10
```

在这个例子里，如果 number 是奇数，就会执行到 continue ，当前循环就停止并自动进入下一次循环了。

或者，我们也可以使用 break 来终止整个循环。例如，值大于8时，就终止循环：

```
for number in 1...10 {
    if number > 8 { break }
    print(number)
}
// 1 2 3 4 5 6 7 8
```

## What's next?

以上，就是Swift中代码分支和循环最基本的用法。面对我们日常可能需要判断的各种复杂的情况，Swift从函数式编程中借鉴了很多匹配复杂条件的方法，力图帮助我们从拼接各种复杂的逻辑条中解放出来，进而让代码呈现更好的语义表现方式。在下一节中，我们就来了解下Swift中丰富的条件匹配方式。

返回视频

(/series/make-a-decision)

使用简单的样式匹配

(https://www.boxueio.com/series/make-a-decision/ebook/135)

职场漂泊的你，每天多学一点。

从开发、测试到运维，让技术不再成为你成长的绊脚石。我们用打磨产品的精神去传播知识，把最新的移动开发技术，通过简单的图表， 清晰的视频，简明的文字和切实可行的例子一一向你呈现。让学习不仅是一种需求，也是一种享受。

## 泊学动态

一个工作十年PM终创业的故事（二） (https://www.boxueio.com/after-the-full-upgrade-to-swift3)
Mar 4, 2017

人生中第一次创业的"10有" (https://www.boxueio.com/founder-chat)
Jan 9, 2016

猎云网采访报道泊学 (http://www.lieyunwang.com/archives/144329)
Dec 31, 2015

What most schools do not teach (https://www.boxueio.com/what-most-schools-do-not-teach)
Dec 21, 2015

一个工作十年PM终创业的故事（一） (https://www.boxueio.com/founder-story)
May 8, 2015

## 泊学相关

关于泊学 >

加入泊学 >

泊学用户隐私以及服务条款 (HTTPS://WWW.BOXUEIO.COM/TERMS-OF-SERVICE)

版权声明 (HTTPS://WWW.BOXUEIO.COM/COPYRIGHT-STATEMENT)

## 联系泊学

Email: 10[AT]boxue.io (mailto:10@boxue.io)
QQ: 2085489246



友情链接 SwiftV (http://www.swiftv.cn) | Seay信息安全博客 (http://www.cnseay.com) | Swift.gg (http://swift.gg/) | Laravist (http://laravist.com/) | SegmentFault (https://segmentfault.com) | 靛青K的博客 (http://blog.dianqk.org/)