Swift 环境搭建 Swift 数据类型

Swift 基本语法

在上一章节中我们已经讲到如何创建 Swift 语言的 "Hello, World!" 程序。现在我们来复习下。

如果创建的是 OS X playground 需要引入 Cocoa ：

```
import Cocoa

/* 我的第一个 Swift 程序 */
var myString = "Hello, World!"

print(myString)
```

如果我们想创建 iOS playground 则需要引入 UIKit :

```
import UIKit
var myString = "Hello, World!"
print(myString)
```

执行以上程序，输出结果为：

```
Hello, World!
```

以上代码即为 Swift 程序的基本结构，接下来我们来详细说明结构的组成部分。

Swift 引入

我们可以使用 **import** 语句来引入任何的 Objective-C 框架（或 C 库）到 Swift 程序中。例如 **import cocoa** 语句导入了使用了 Cocoa 库和API，我们可以在 Swift 程序中使用他们。

Cocoa 本身由 Objective-C 语言写成，Objective-C 又是 C 语言的严格超集，所以在 Swift 应用中我们可以很简单的混入 C 语言代码，甚至是 C++ 代码。

Swift 标记

Swift 程序由多种标记组成，标记可以是单词，标识符，常量，字符串或符号。例如以下 Swift 程序由三种标记组成：

```
print("test!")
```

以上语句由 3 个符号组成：单词(**print**)、符号(**(**)、字符串(**"test"**)。

```
print
(
   "test!"
)
```

注释

Swift的注释与C语言极其相似，单行注释以两个反斜线开头：

```
//这是一行注释
```

多行注释以/*开始，以*/结束:

```
/* 这也是一条注释，
但跨越多行 */
```

与 C 语言的多行注释有所不同的是，Swift 的多行注释可以嵌套在其他多行注释内部。写法是在一个多行注释块内插入另一个多行注释。第二个注释块封闭时，后面仍然接着第一个注释块：

```
/* 这是第一个多行注释的开头

/* 这是嵌套的第二个多行注释 */

这是第一个多行注释的结尾 */
```

多行注释的嵌套是你可以更快捷方便的注释代码块，即使代码块中已经有了注释。

分号

与其它语言不同的是，Swift不要求在每行语句的结尾使用分号(;)，但当你在同一行书写多条语句时，必须用分号隔开：

```
import Cocoa
/* 我的第一个 Swift 程序 */
var myString = "Hello, World!"; print(myString)
```

标识符

标识符就是给变量、常量、方法、函数、枚举、结构体、类、协议等指定的名字。构成标识符的字母均有一定的规范，Swift语言中标识符的命名规则如下：

- 区分大小写，Myname与myname是两个不同的标识符；
- 标识符首字符可以以下划线（_）或者字母开始，但不能是数字；

- 标识符中其他字符可以是下划线（_）、字母或数字。

例如： userName、User_Name、_sys_val、身高等为合法的标识符，而2mail、room#和class为非法的标识符。

注意: Swift中的字母采用的是Unicode编码[1]。Unicode叫做统一编码制，它包含了亚洲文字编码，如中文、日文、韩文等字符，甚至是我们在聊天工具中使用的表情符号

如果一定要使用关键字作为标识符，可以在关键字前后添加重音符号（`），例如：

```
let `class` = "Runoob"
```

关键字

关键字是类似于标识符的保留字符序列，除非用重音符号（`）将其括起来，否则不能用作标识符。关键字是对编译器具有特殊意义的预定义保留标识符。常见的关键字有以下4种。

与声明有关的关键字

class	deinit	enum	extension
func	import	init	internal
let	operator	private	protocol
public	static	struct	subscript
typealias	var		

与语句有关的关键字

break	case	continue	default
do	else	fallthrough	for
if	in	return	switch
where	while		

表达式和类型关键字

as	dynamicType	false	is
nil	self	Self	super
true	_COLUMN_	_FILE_	_FUNCTION_
LINE			

在特定上下文中使用的关键字

associativity	convenience	dynamic	didSet
final	get	infix	inout
lazy	left	mutating	none
nonmutating	optional	override	postfix
precedence	prefix	Protocol	required
right	set	Type	unowned
weak	willSet		

Swift 空格

Swift语言并不是像C/C++，Java那样完全忽视空格，Swift对空格的使用有一定的要求，但是又不像Python对缩进的要求那么严格。

在Swift中，运算符不能直接跟在变量或常量的后面。例如下面的代码会报错：

```
let a= 1 + 2
```

错误信息是：

```
error: prefix/postfix '=' is reserved
```

意思大概是等号直接跟在前面或后面这种用法是保留的。

下面的代码还是会报错（继续注意空格）：

```
let a = 1+ 2
```

错误信息是：

```
error: consecutive statements on a line must be separated by ';'
```

这是因为Swift认为到1+这个语句就结束了，2就是下一个语句了。

只有这样写才不会报错：

```
let a = 1 + 2;  // 编码规范推荐使用这种写法
let b = 3+4 // 这样也是OK的
```

Swift 字面量

所谓字面量，就是指像特定的数字，字符串或者是布尔值这样，能够直接了当地指出自己的类型并为变量进行赋值的值。比如在下面：

```
42                  // 整型字面量
3.14159             // 浮点型字面量
"Hello, world!"     // 字符串型字面量
true                // 布尔型字面量
```

打印输出

Swift 使用 print 函数打印输出:

```
print("Runoob") // 输出 Runoob
```

print 函数是一个全局函数,完整的函数签名为:

```
public func print(items: Any..., separator: String = default, terminator: String = default)
```

如果我们想让其不换行输出,只需要将最后一个参数赋值为空字符串即可:

```
for x in 0...10{
    print("\(x) ", terminator: "")
}
print()
```

输出结果为：

```
0 1 2 3 4 5 6 7 8 9 10
```

如果你需要接收用户的输入可以使用 `readLine()`:

```
let theInput = readLine()
```

← Swift 环境搭建 Swift 数据类型 →

let 用于定义常量，定义完后不能修改。

var 用于定义变量，可以修改。

swift可以自动识别属性类别。

DJ 2年前 (2017-06-11)