

基础语法

@M了个J

<https://github.com/CoderMJLee>

<http://cnblogs.com/mjios>

码拉松



实力IT教育 www.520it.com

Hello World

```
print("Hello World!")
```

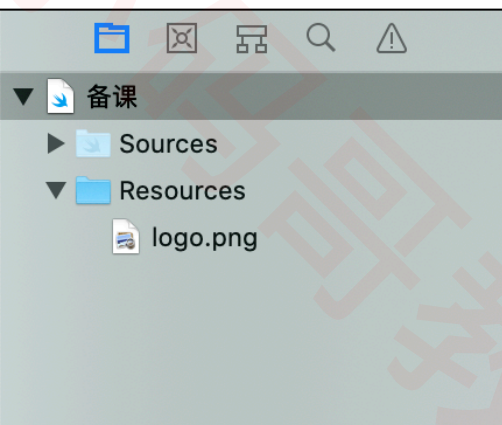
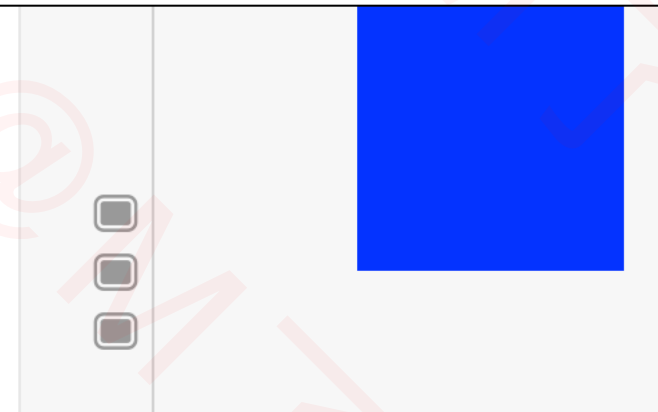
- 不用编写main函数，Swift将全局范围内的首句可执行代码作为程序入口
- 一句代码尾部可以省略分号（;），多句代码写到同一行时必须用分号（;）隔开
- 用var定义变量，let定义常量，编译器能自动推断出变量\常量的类型

1	let a = 10	10
2	let b = 20	20
3	var c = a + b	30
4	c += 30	60

- Playground可以快速预览代码效果，是学习语法的好帮手
- Command + Shift + Enter：运行整个Playground
- Shift + Enter：运行截止到某一行代码

Playground - View

```
1 import UIKit
2 import PlaygroundSupport
3
4 let view = UIView()
5 view.frame = CGRect(x: 0, y: 0, width: 100, height: 100)
6 view.backgroundColor = UIColor.blue
7 PlaygroundPage.current.liveView = view
```

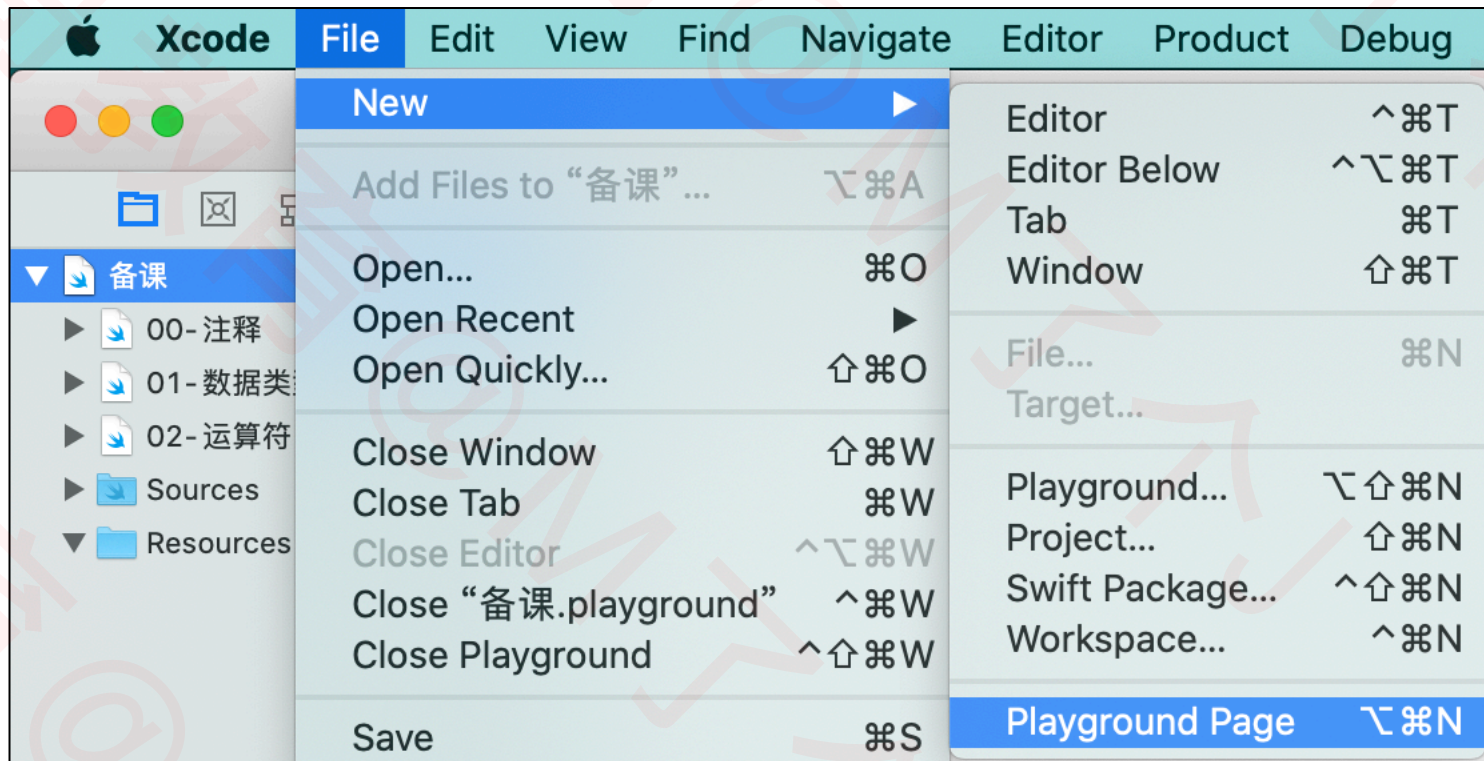
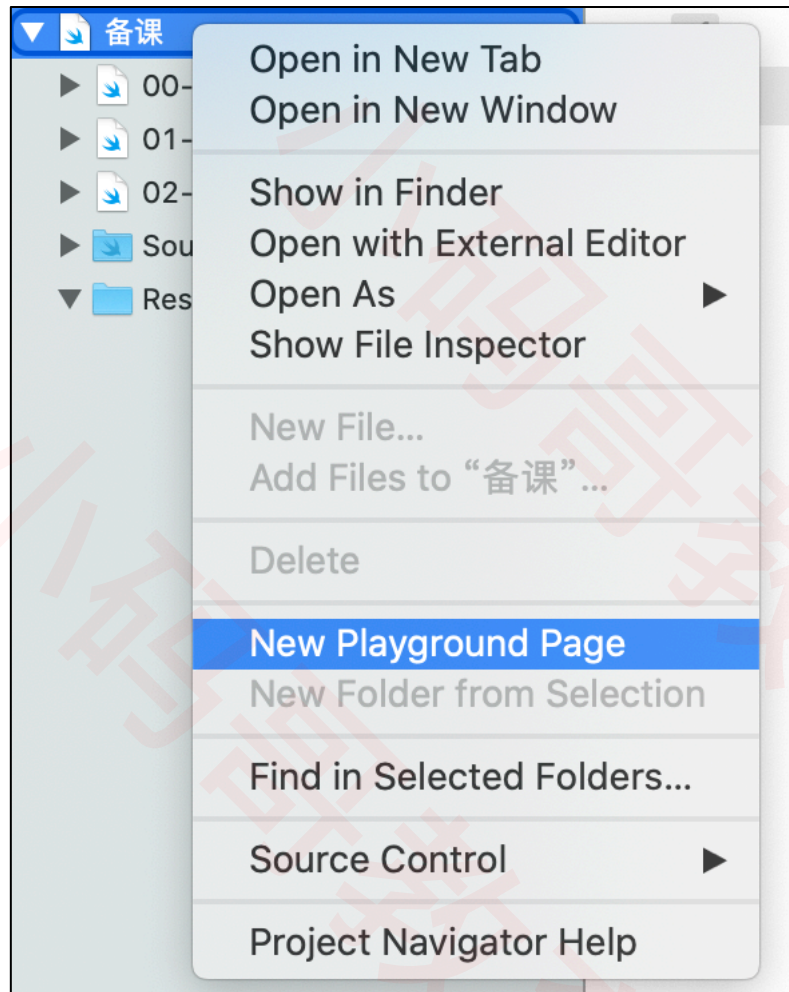


Playground - ViewController

```
1 import UIKit
2 import PlaygroundSupport
3
4 let vc = UITableViewController()
5 vc.view.backgroundColor = UIColor.lightGray
6 PlaygroundPage.current.liveView = vc
```



Playground – 多Page



```
// 单行注释
```

```
/*  
多行注释  
*/
```

```
/*  
1  
/* 多行注释的嵌套 */  
2  
*/
```

■ Playground的注释支持markup语法 (与markdown相似)

```
//: 开始markup
```

```
/*:  
开始markup  
*/
```

■ 开启markup渲染效果 : Editor -> Show Rendered Markup

■ 注意 : Markup只在Playground中有效


```
//: [上一页](@previous)
//: [下一页](@next)
```

上一页 下一页

```
/*:
# 一级标题

## 无序列表
- First Item
- Secound Item

## 有序列表
1. First Item
2. Secound Item

## 笔记
> This is a note
---

## 图片
![Logo](logo.png "Local image")

## 链接
* [小码哥教育](https://520it.com)

## 粗体/斜体
这是**Bold**, 这是*Italic*
*/
```

一级标题

无序列表

- First Item
- Secound Item

有序列表

1. First Item
2. Secound Item

笔记

Note
This is a note

图片



链接

- 小码哥教育

粗体/斜体

这是Bold, 这是Italic

- 只能赋值1次
- 它的值不要求在编译时期确定，但使用之前必须赋值1次

```
let age1 = 10

let age2: Int
age2 = 20

func getAge() -> Int {
    return 30
}

let age3 = getAge()
```

- 常量、变量在初始化之前，都不能使用

```
let age: Int
var height: Int
print(age)
print(height)
```

! Constant 'age' used before being initialized

! Variable 'height' used before being initialized

- 下面代码是错误的

```
let age
age = 20
```

! Found an unexpected second identifier in constant declaration; is there an accidental break?

标识符

- 标识符（比如常量名、变量名、函数名）几乎可以使用任何字符
- 标识符不能以数字开头，不能包含空白字符、制表符、箭头等特殊字符

```
func 🐮🍺() {  
    print("666")  
}
```

```
🐮🍺()
```

```
let 🛸 = "ET"
```

```
var 🥛 = "milk"
```

常见数据类型

值类型 (value type)	枚举 (enum)	Optional
	结构体 (struct)	Bool、Int、Float、Double、Character
		String、Array、Dictionary、Set
引用类型 (reference type)	类 (class)	

- 整数类型：Int8、Int16、Int32、Int64、UInt8、UInt16、UInt32、UInt64
- 在32bit平台，Int等价于Int32；在64bit平台，Int等价于Int64
- 整数的最值：UInt8.max、Int16.min
- 一般情况下，都是直接使用Int即可
- 浮点类型：Float，32位，精度只有6位；Double，64位，精度至少15位

```
let letFloat: Float = 30.0
let letDouble = 30.0
```

字面量

```
// 布尔  
let bool = true // 取反是false
```

```
// 整数  
let intDecimal = 17 // 十进制  
let intBinary = 0b10001 // 二进制  
let intOctal = 0o21 // 八进制  
let intHexadecimal = 0x11 // 十六进制
```

```
// 浮点数  
let doubleDecimal = 125.0 // 十进制, 等价于1.25e2, 0.0125等价于1.25e-2  
let doubleHexadecimal1 = 0xFp2 // 十六进制, 意味着15x2^2, 相当于十进制的60.0  
let doubleHexadecimal2 = 0xFp-2 // 十六进制, 意味着15x2^-2, 相当于十进制的3.75  
/* 以下都是表示12.1875  
   十进制: 12.1875、1.21875e1  
   十六进制: 0xC.3p0 */
```

```
// 数组  
let array = [1, 3, 5, 7, 9]
```

```
// 字符串  
let string = "小码哥"
```

```
// 字符 (可存储ASCII字符、Unicode字符)  
let character: Character = "🐶"
```

- 整数和浮点数可以添加额外的零或者添加下划线来增强可读性
- 100_0000、1_000_000.000_000_1、000123.456

```
// 字典  
let dictionary = ["age" : 18, "height" : 168, "weight" : 120]
```

类型转换

// 整数转换

```
let int1: UInt16 = 2_000
let int2: UInt8 = 1
let int3 = int1 + UInt16(int2)
```

// 整数、浮点数转换

```
let int = 3
let double = 0.14159
let pi = Double(int) + double
let intPi = Int(pi)
```

// 字面量可以直接相加，因为数字字面量本身没有明确的类型

```
let result = 3 + 0.14159
```

元组 (Tuple)

```
let http404Error = (404, "Not Found")  
print("The status code is \(http404Error.0)")
```

```
let (statusCode, statusMessage) = http404Error  
print("The status code is \(statusCode)")
```

```
let (justTheStatusCode, _) = http404Error
```

```
let http200Status = (statusCode: 200, description: "OK")  
print("The status code is \(http200Status.statusCode)")
```