



XCPlayground

Nate Cook 撰写、*Yifan Xiao* 翻译、发布于2015年5月11日

等一下! 既然是 Playground 这个标题, 为什么不试试用 Playground 来读这篇文章 (英文) 呢? [点击下载 →](#)

玩儿.

联想起童年时代的各种精力旺盛, 我们通常把 玩 当做 工作 的对立. 其实不然, 玩中的很多要素其实深入到了我们每天的工作当中: 实验, 探索, 发现。

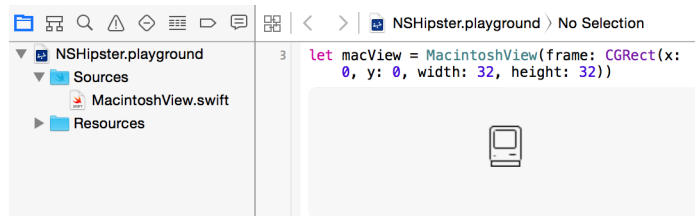
Playgrounds 本质上并不是 Swift 语言的一个特性——相反的, 他们是 Swift 对于其所有功能, 从其效率和性能, 到他的深度和透明度的展现。Playgrounds 使得创建一个可以工作的程序变得如此的简单——的确, 每一个新的 Playground 都开始于一个我们再熟悉不过的 "Hello, playground". 在这时, Playgrounds 隐藏了需要他强大的特性, 使得探索成为了唯一发现他们超高级性能的方式了。

本周, 我们将要透过 Playgrounds 的表面, 告诉你那些能够带给你开发极大帮助的工具。继续阅读关于 Playground 的 sources 和 resource, captured values 和 extended execution, 以及整合的 rich formatting, 讲 Playground 变成了具有互动性的教学工具。

注意: 最近刚刚发行的电子版本的 *NSHipster: Obscure Topics in Cocoa & Swift* 包括了一个 Playgrounds 的包——每一个对应书中的一个章节。每一个 Playground 都提供了一个探索和实验里面所提到的概念的机会, 包括延伸的案例。

Sources

更新后的 Playground 包, 包括了一个 "Sources" 的代码文件夹, 十分稳定。所有在 "Sources" 路径下的文件都是经过编译到一个单独的模块 (只有一次, 并不是每次你输入都会编译), 并且自动导入到 Playground。除了简化你的 Playground 中的代码, 该编译极大的提升了执行速度。这意味着当你在文件中定义了一个 public 的类型, 函数, 或者全局常量的时候, 就可以立即的使用:



在你的 Playground 中试试，打开 Project Navigator (⌘1) 并展开 Playground 文件，你就能看到"Sources"路径。

导入Frameworks

如果想要导入外部 framework，创建一个 Xcode Workspace 包含了 framework 项目和你的 Playground。在 Build 之后，就可以通过常规的import命令导入对应的包。

Resources

没有 sandbox 的 Playground 是不完整的，同样，Swift Playground 也不会让你失望。Playgrounds 有两个与相关的resources关联起来：一个是每一个独立的 playground 本地的，另一个则是 playground 之间共享的。在你的实验过程中，Playgrounds 能够支持 XML，JSON 数据，XIB，和图像文件。这也增加了其使用可用性。

本地

Resources 文件夹, 与 Sources 文件夹一样在 Playground 的包路径中, 通过 Project Navigator 就可见了——只需要简单的拖拽图像和数据文件，就可以在 Playground 中使用了。对应的内容在 main bundle 中也是可见的。比如，我们可以像这样非常快捷的加载一个包含天气数据的 JSON 文件：

Swift

```
let jsonPath = NSBundle.mainBundle().bundlePath.stringByAppendingPathComponent("weather.json")
if let
    jsonData = NSData(contentsOfFile: jsonPath),
    json = NSJSONSerialization.JSONObjectWithData(jsonData, options: nil, error: nil) as? [String:
{
    // ...
}
```

共享

"共享 Playground 数据"的内容在你的"Documents"文件夹路径下，也同样对于你创建的任何 Playground 都可见。我们通过XCPSHaredDataDirectoryPath常量来访问该共享文件夹。

如果你自习想尝试，需要在 "~/Documents/Shared Playground Data" 下简历一个文件夹。这里我们尝试载入一个名字叫做 "image.png" 的图片文件：

Swift

```
let sharedImagePath = XCSharedDataDirectoryPath.stringByAppendingPathComponent("image.png")
if let image = UIImage(contentsOfFile: sharedImagePath) {
    // ...
}
```

捕获值（Captured Values）

一个 Playground 通常立即显示简单表达式的结果。数组，字符串，数字等等，会在结果面板把计算后的结果显示出来。那么，随着时间改变的值是如何处理的呢？

通过使用 `XCPCaptureValue()` 函数，我们可以随着一系列的迭代建立一个变动值的图。回到我们上面提到的天气例子，让我们来看看按小时计的温度数据，使用 `XCPCaptureValue` 来在辅助编辑界面以时间线的方式显示 温度的值：

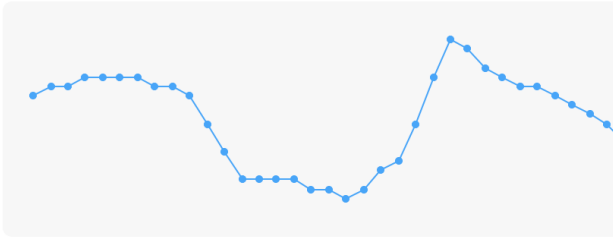
Swift

```
import XCPlayground

for forecast in forecasts {
    if let
        tempString = forecast["temp"]?["english"] as? String,
        temperature = tempString.toInt()
    {
        XCPCaptureValue("Temperature", temperature)
    }
}
```

另一种可选的方式是，选择 **Editor → Show Result For Current Line** 就会捕获当前线的数值并且直接以图表的形势显示在 Playground 流中：

```
for forecast in forecasts {  
    if let  
        tempString = forecast["temp"]?["english"] as? String,  
        temperature = tempString.toInt()  
    {  
        temperature  
    }  
}
```



异步执行（Asynchronous Execution）

不同于大部分 Swift 代码，是作为框架或者应用的一部分，Playgrounds 被当做是 *高级代码*。Playground 中的高级代码是按照指令接着指令的顺序从上到下执行的。这种无容器风格的代码执行提供了立即反馈，但是存在着一个问题：在执行到了 Playground 底部后，会立即停止。网络请求，计时器，以及长时间运行的后台队列都会在提供反馈成功或者失败之前被立即终止。

为了保持执行能够足够长，以提供上述异步操作的结果，XCPlayground 模块包括了一个能够延长该过程的函数：

Swift

```
import XCPlayground  
  
XCPSetExecutionShouldContinueIndefinitely(continueIndefinitely: true)  
  
let url = NSURL(string: "http://httpbin.org/image/png")!  
let task = NSURLSession.sharedSession().dataTaskWithURL(url) {  
    data, _, _ in  
        let image = UIImage(data: data)  
        // ...  
}  
task.resume()
```

除了在 Playground 中的代码，通过控制台运行的脚本中的，在 Swift REPL，或者在项目可选的 "main.swift" 文件中的 Swift 代码，也都被当做为高级代码。伴随着顺序依存的代码执行，高级代码成为了唯一能够包含定义在函数，或者封装在类型之外的表达式的种类。

文档（Documentation）

除了实验用途，Playgrounds 在展示 Swift 语言的工具和框架中也一样强大。特别文档部分可以作为丰富格式的方式展示出来，以提供对于代码的清晰解释从而展示某个技术或者正确使用某个 Library 的方式。

不同于 **Swift's other documentation syntax**, Swift Playgrounds 使用 Markdown 来显示多格式的文档。
(如果你现在读了这篇文章的 Playground，你现在其实就在阅读 Markdown 了)。一个加入一行，或者多行评论的冒号(:)，就表示了一个多格式的评论：

Swift

```
//: This line will have bold and italic text.

/*:
## Headers of All Sizes

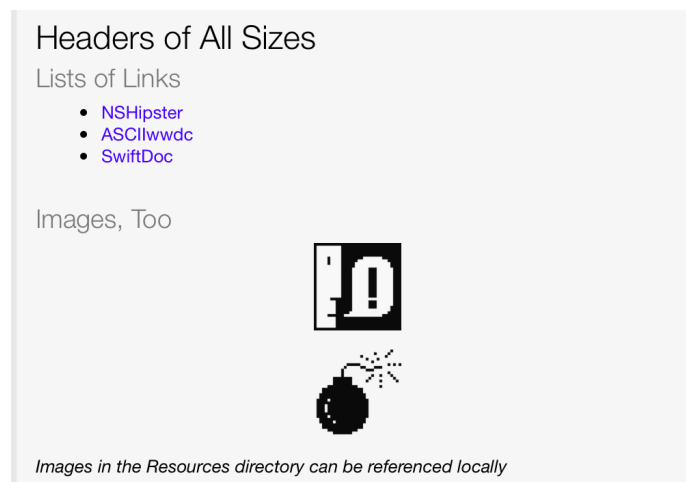
### Lists of Links

- [NSHipster](http://nshipster.com)
- [ASCIIfwwdc](http://asciifwwdc.com)
- [SwiftDoc](http://swift-doc.org)

### Images, Too

! [Remote Image](http://nshipster.s3.amazonaws.com/alert.gif)
! [Local Image](bomb.gif)

*Images in the Resources directory can be referenced locally*
*/
```



可以通过选择 **Editor** → **Show Rendered Markup** 菜单，从而选择切换到获得多格式文档，也可以在 File Inspector (⌘⇧1) 选中 **Render Documentation** 复选框。

Playgrounds 提供了一个我们关于分享和学习 OS X 和 iOS 相关工具的方式的重大改变。Playground 可以展示每一个特性，并且为将来的用户探索 and 发现你创建的库提供了空间。丢掉你的静态 README.md，换成可互动的 README.playground 吧，再玩起来！

作者

**Nate Cook**

Nate Cook (@nnnnnnnn) is an independent web and application developer who writes frequently about topics in Swift, and the creator of SwiftDoc.org.

翻译者

**Yifan Xiao**

下一篇文章

iOS 9

WWDC 2015 虽不像往届那样精彩纷呈，但却丝毫不缺乏亮点。这周我们将一起探讨 iOS 9 给我们熟悉和热爱的 API 带来了哪些改进。

相关文章

- [Unmanaged](#)
- [NSCalendarUnitYear](#)
- [Swift System Version Checking](#)

© 除非另有声明、本网站采用知识共享「署名-非商业性使用 3.0 中国大陆」许可协议授权。

本站文章由 [Croath Liu](#)、[Delisa Mason](#)、[Jack Flintermann](#)、[Matt Thompson](#)、[Mike Lazer-Walker](#)、[Natasha Murashev](#) 和 [Nate Cook](#) 撰写、[Andrew Yang](#)、[April Peng](#)、[Bob Liu](#)、[Candyan](#)、[Chester Liu](#)、[Croath Liu](#)、[Daniel Hu](#)、[David Liu](#)、[GWesley](#)、[Henry Lee](#)、[JJ Mao](#)、[Lin Xiangyu](#)、[Ricky Tan](#)、[Sheldon Huang](#)、[Tiny Tian](#)、[Tony Li](#)、[Yifan Xiao](#)、[Yu Jin](#) 和 [Zihan Xu](#) 翻译。

