

SwiftCon China 2016

www.swiftconchina.com



拥抱 Swift 3.0 与未来 展望

2016-04-23 @图拉鼎

简介

- Swift 3.0 的大目标
- Swift 3.0 已经和即将发生的变化
- Swift 的周边和未来

为什么要讲 Swift 3.0
这个「未来的话题」？

2016 年，你是否打算迁移到 Swift 开发
(单选)

2015 年就在使用 Swift 了 (已选)

371票 11%

2016 年上半年就开始用 Swift

275票 8%

希望 2016 年下半年 Swift 3.0 稳定后再考虑

1030票 32%

2016 年打算继续使用 Objective-C

742票 23%

自己很想迁移到 Swift，无奈公司不允许

262票 8%

明年打算继续使用 Objective-C

250票 7%

我不是 iOS 开发者，我来围观看答案

280票 8%

不到 20% 的开发者正在
使用 Swift

—来自唐巧的「iOS开发」的调查



32% 的开发者在观望
Swift 3.0 的表现再考虑

—来自唐巧的「iOS开发」的调查

分享的目标

- 给正在观望的开发者：提前做好拥抱的准备
- 给正在使用的开发者：提前做好再次被虐的准备

个人被虐经历

初识的惊喜



图拉鼎 

2014-6-3 02:47 来自 奇点


新的语言：**Swift**!!!

阅读 3147 推广

 1

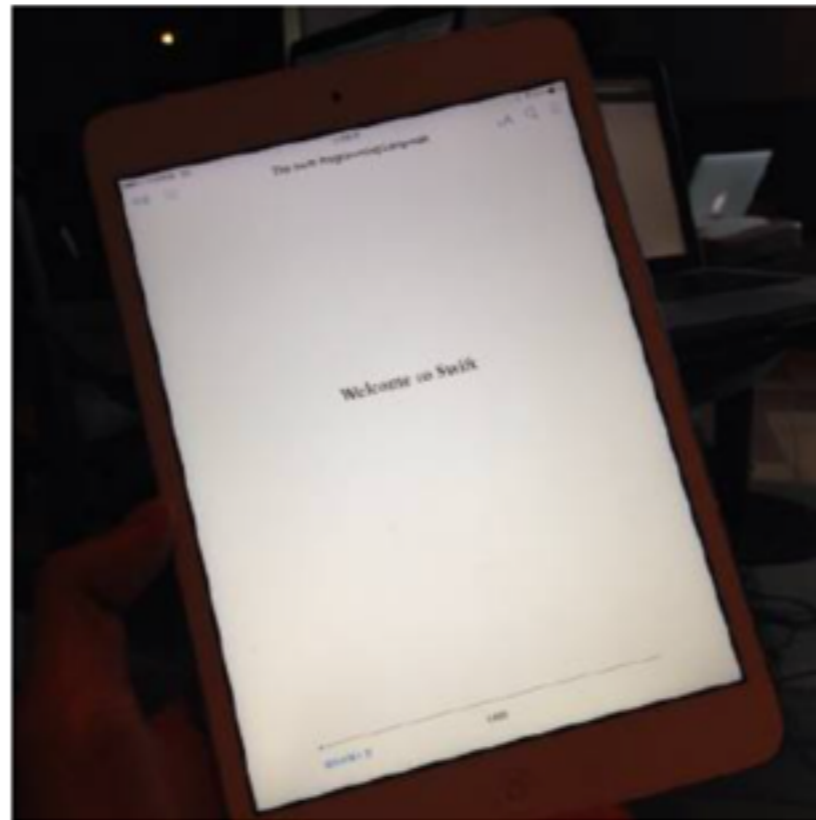
积极地学习



图拉鼎 

2014-6-3 03:13 来自 iOS

开始学习 **Swift** 了



阅读 6606 推广

 9

 11

动手实践

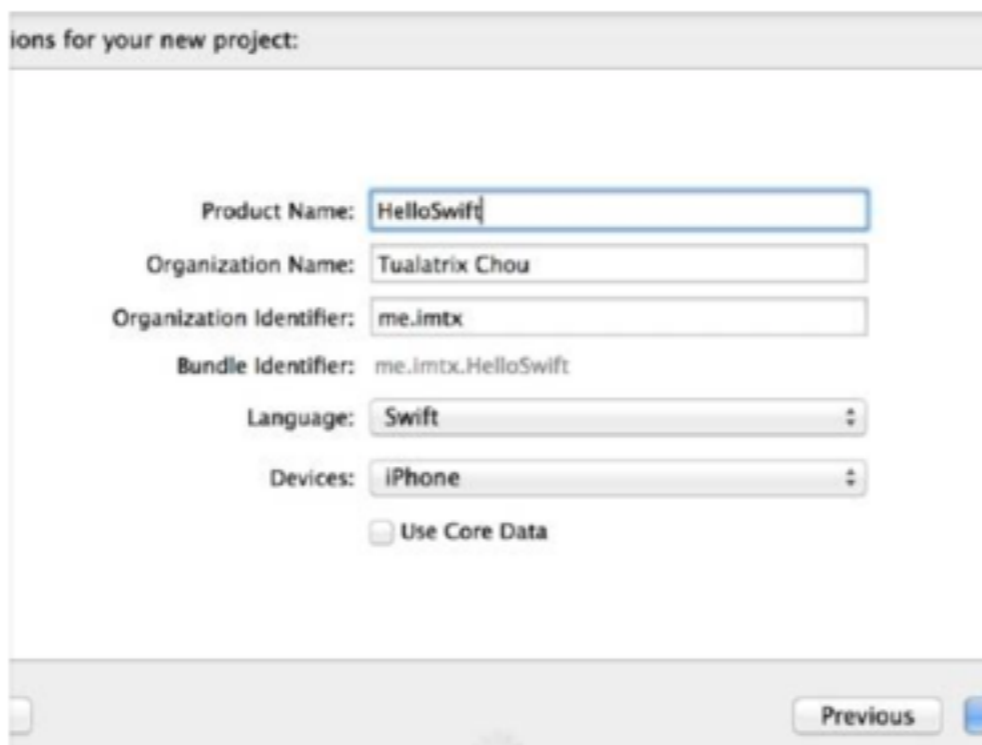


图拉鼎

2014-6-3 09:12 来自 OS X

用 Xcode 6 创建了我的第一个 **Swift** 语言项目：Hello **Swift**。目前情绪稳定，这个语言依然有 Objective-C 式的有名字的参数，可读性非常

强：
`self.tableView.insertRowsAtIndexPaths([indexPath],
withRowAnimation: .Automatic)`



阅读 5116 推广

5

9

赞

第一次遇到阻碍



图拉鼎

2014-6-4 10:57 来自 OS X

乱写几行 **Swift** 代码后，编译器直接 Segment Fault 了...不敢玩了。😞

收起 | 查看大图 | 向左旋转 | 向右旋转

```
1. While emitting IR SIL function
   @_TFC10HelloSwift20MasterViewController11viewDidLoadfS0_F
   T_T_ for 'viewDidLoad' at /Users/tualatrix/Desktop/
   HelloSwift/HelloSwift/MasterViewController.swift:20:14
   <unknown>:0: error: unable to execute command:
   Segmentation fault: 11
   <unknown>:0: error: swift frontend command failed due to
   signal (use -v to see invocation)
   Command /Applications/Xcode6-Beta.app/Contents/Developer/
   Toolchains/XcodeDefault.xctoolchain/usr/bin/swift failed
   with exit code 254
```

Command /Applications/Xcode6-Beta.app/Contents/Developer... [more](#)



Activity Log Complete 6/4/14, 10:53 AM
1 error

阅读 2.8万 推广

19

5

1

后悔吹牛

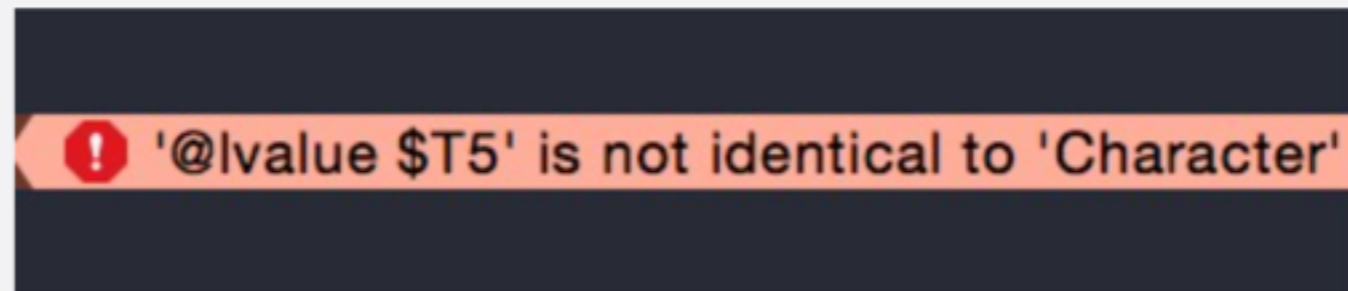


图拉鼎

2014-11-10 22:22 来自 微博 weibo.com

Swift 的错误提示真不是人看的，当初我为何要吹下牛逼用 Swift 写一款 App...后悔莫及

收起 | 查看大图 | 向左旋转 | 向右旋转



阅读 6270 推广

1

11

3

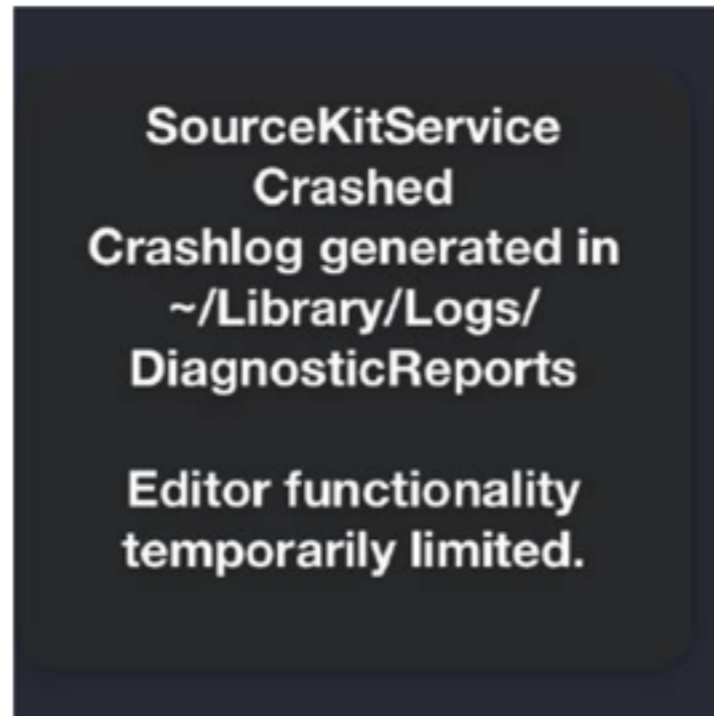
渐渐麻木



图拉鼎

2014-11-10 23:15 来自 微博 weibo.com

写 **Swift** 的同学，你碰到过一秒钟弹这个提示 10 次的情况吗？



阅读 6175 推广

转发

6

赞

再次被震惊



图拉鼎

2014-11-23 15:55 来自 微博 weibo.com

晴天霹雳! **Swift** 在 Xcode 下实际上是二等公民!

收起 | 查看大图 | 向左旋转 | 向右旋转



Can't refactor Swift code.

Xcode can only refactor C and Objective-C code.

OK

阅读 1.1万 推广

6

9

6

被虐经历的背后

- 奇点 for 微博：2014 年 6 月用 Swift 重写，2015 年 1 月发布
- Manico 2：2015 年 4 月用 Swift 重写，12 月发布
- Swift 开发环境也在不断变好...

Swift 3.0 大目标

ABI 稳定

ABI 是什么?

- Application Binary Interface (应用二进制接口)
- ABI: 二进制级接口, API: 源码级接口
- 不稳定的 ABI == 不兼容的二进制包

ABI 不稳定

Swift 2.2
二进制包

+

Swift 2.2 项目

=

Swift 2.2 App

Swift 2.2
二进制包

+

Swift 3.0 项目

≠

Swift 3.0 App

ABI 稳定

Swift 3.0
二进制包

+

Swift 3.0 项目

=

Swift 3.0 App

Swift 3.0
二进制包

+

Swift 4.0 项目

=

Swift 4.0 App

 **strougtonsmith** 4h
@jckarter interesting, thought that was for v2 🙄

0 Faves 0 Retweets

15/10/3 at 上午11:00 via Twitter Web Client

← ↻ ☆ ↗ ⚙

In reply to...

 **jckarter** ★
@strougtonsmith Swift's not ABI-stable yet so still needs to be embedded in the apps that use it.

 **strougtonsmith** 4h
@jckarter surprised the standard set of libraries isn't included with the OS yet. v3?

 **jckarter** ★
@strougtonsmith Yep. iOS 9's Calculator was rewritten in Swift too.

 **strougtonsmith** 4h
Curiously enough, Dock does look like it's using Swift for some of the new Mission Control stuff. Though it embeds parts of the std lib?

“Swift’s not ABI-stable yet so still needs to be embedded in the apps that use it.”

– @jckarter (Apple Swift 开发组成员)

ABI 稳定的里程碑意义

- 使分发二进制（闭源）三方库成为可能
- iOS 系统将内置 Swift 库：Swift App 体积减少
- 生态圈更加活跃：和 OC 一样成为生态圈一级公民

大目标：可移植性

- 完整支持 Linux（及其他）
- Package Manager 包管理：
 - 取代 Carthage、CocoaPods，跨平台

Swift 3.0 大目标小结

- ABI 稳定
- 跨平台：支持 Linux
- Package Manager
- 最大的大目标...

API 及核心语言完善

- 内置库的 API 规范
- 与 Objective-C 更棒的兼容性
- 已经部分完成

回看 Swift 2.2

#selector 语法

```
UITapGestureRecognizer(target: self,  
                        action: "tapAction")
```

```
UITapGestureRecognizer(target: self,  
                        action: #selector(Controller.tapAction))
```

#selector 语法优点

- 编译器检查
- 代码补全

Swift 3.0 之 #keyPath

- 专门用于访问 property 形式的 key

传统的 Objective-C 的用法

```
@interface Person : NSObject  
  
@property (nonatomic, strong) NSString *name;  
  
@end
```

```
NSString *name = person.name;  
person.name = @"图拉鼎";
```

```
[person name];  
[person setName:@"图拉鼎"];
```

传统的 Objective-C 的用法

```
[person valueForKey:@"name"];  
[person setValue:@"图拉鼎" forKey:@"图拉鼎"];
```

```
NSDictionary *keyValues = @{@"name": @"图拉鼎"};
```

```
[keyValues enumerateKeysAndObjectsUsingBlock:^(id  
key, id obj, BOOL *stop) {  
    [person setValue:obj forKey:key];  
}];
```


Swift 3.0 的 #keyPath 做法

```
class Person: NSObject {  
    dynamic var name: String  
  
    init(name: String) {  
        self.name = name  
    }  
}
```

```
person valueForKey(#keyPath(Person.name))
```

```
let keyValues = [#keyPath(Person.name): "图拉鼎"]
```

```
for (key, value) in keyValues {  
    person.setValue(key, forKey: value)  
}
```

Swift 3.0 之标准库重命名

- 去 Objective-C 化
- 简单和通用化
- 参数化

去 Objective-C 化

```
"Swift".stringByReplacingOccurrencesOfString("Swift",  
withString: "OK")
```

```
"Swift".replacingOccurrences("Swift", with: "OK")
```

标准化命名 (一)

```
"Swift".lowercaseString
```

```
"Swift".uppercaseString
```

```
"Swift".lowercased()
```

```
"Swift".uppercased()
```

标准化命名 (二)

`sortInPlace()` -> `sort()`

`sort()` -> `sorted()`

`reverse()` => `reversed()`

`enumerate()` => `enumerated()`

`SequenceType.minElement()` => `.min()`, `.maxElement()`
=> `.max()`

参数化命名

```
["Hello", "World"].joinWithSeparator(",")
```

```
["Hello", "Swift"].joined(separator:",")
```

```
"Hello, world".componentsSeparatedByString(",")
```

```
"Hello, world".components(separatedBy:",")
```

Swift 标准库重命名的意义

- API 更清晰
- 渐渐摆脱 Objective-C 的阴影

Swift 3.0 之 废除 IUO

IUO 是什么?

- ImplicitlyUnwrappedOptional
- 一种自动解包的 Optional、解到 nil 就 crash 的 Optional
- `var x: String!`

为何废除 IUO 类型

- 强制 unwrap 时遇 nil 会 Crash
- 有传播性

Swift 2.2 的 IUO 类型

```
func g() -> Int! {  
    return nil  
}
```

```
let a = g()
```

```
print(a)
```

```
> fatal error: unexpectedly found nil while unwrapping an Optional  
value
```

IUO 的变化

- 从 Swift 2.2 的 Type 变成 Swift 3.0 的 Attribute

Swift 3.0 的 IUO 属性

```
func g() -> Int! {  
    return nil  
}
```

```
let a = g()
```

```
print(a)
```

```
> nil
```

Swift 3 的 IUO 属性的原则

- 不随赋值传播下去，除非开发者继续声明为！
- 遇到不能以 Optional 进行的计算，才触发 unwrap

Swift 3.0 的其他特性

- SE-0025: Scoped Access Level
- SE-0038: Package Manager C Language Target Support
- SE-0048: Generic Type Aliases
- 更多可见：<https://github.com/apple/swift-evolution>

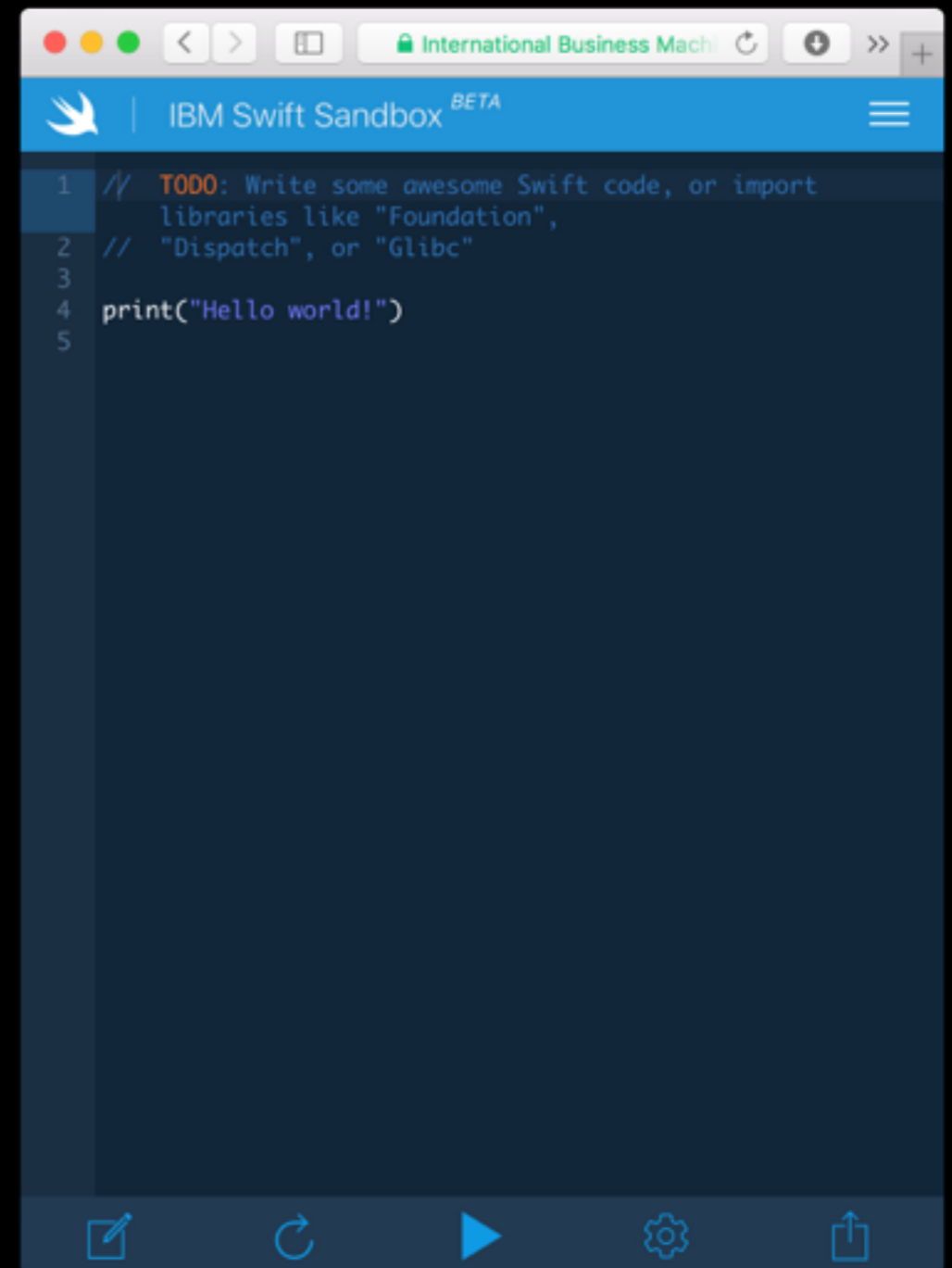
Swift 3.0 的生态圈

- <http://www.zewo.io>
- <http://perfect.org/>
- <https://github.com/necolt/Swifton>



来自 IBM 的 Swift 资源

- Swift Sandbox: <https://swiftlang.ng.bluemix.net/>
- Swift Package Catalog <https://swiftpkgs.ng.bluemix.net/>
- Kitura <https://github.com/IBM-Swift/Kitura>



The screenshot shows a web browser window titled "IBM Swift Sandbox ^{BETA}". The browser's address bar shows "International Business Machi". The page content is a code editor with the following Swift code:

```
1 // TODO: Write some awesome Swift code, or import
  libraries like "Foundation",
2 // "Dispatch", or "Glibc"
3
4 print("Hello world!")
5
```

The code editor has a dark blue background with light blue text. The line numbers 1 through 5 are visible on the left side. At the bottom of the editor, there are five icons: a pencil (edit), a refresh symbol, a play button (run), a gear (settings), and a share icon.

Thank You!