

CSDN

博客 (/blog.csdn.net/)

学院 (/edu.csdn.net/)

下载中心 (/download.csdn.net/)

GitChat (http://gitbook.cn/?ref=csdn)

论坛 (http://bbs.csdn.net)

写博客

发Chat

登录 (/so.csdn.net/so/) 注册 (/so.csdn.net/register?utm_source=csdnblog1)

Swift内存模型的那点儿

原创

2017年07月20日 11:55:00

1887

跟OC类似，Swift提供了MemoryLayout类静态测量对象大小，注意是在编译时确定的，不是运行时哦！作为Java程序员想想如何测量Java对象大小？参考Java对象到底有多大？(http://blog.csdn.net/brycegao321/article/details/52987906)

写这篇博客的目的在于说明2个黑科技：

1、类/结构体的成员变量声明顺序会影响它的占用空间。原理是内存对齐，有经验的码农会把占用空间大的变量写在前面，占用空间小的写在后面。PS: 大道同源，C/C++/Object-C/Swift/Java都需要字节对齐；如果面试时间问你内存优化有什么经验？你告诉他这个一定会另眼相看！！

2、可以篡改Swift结构体/类对象的私有成员(通过指针操作内存)。Java要用反射实现。

为什么要内存对齐呢？简单来说就是CPU寻址更快，详情参见内存对齐原因 (https://stackoverflow.com/questions/381244/purpose-of-memory-alignment/381368#381368)

在不同机型上数据类型占用的空间大小也不同，例如iPhone5上Int占4个字节，iPhone7上Int占8个字节。本文是在iPhone7模拟器上验证的。

内存分配：参考结构体和类的区别 (http://blog.csdn.net/brycegao321/article/details/53206324)

Stack(栈)，存储值类型的临时变量，函数调用栈，引用类型的临时变量指针，结构体对象和类对象引用

Heap(堆)，存储引用类型的实例，例如类对象

Swift3.0提供了内存操作类MemoryLayout(注意：Swift跟OC一样，内存排列时需要对齐，造成一定的内存浪费，我们称之为内存碎片)，它有3个主要参数：

1、实例方法alignment和静态方法 alignment(ofValue: T)：

字节对齐属性，它要求当前数据类型相对于起始位置的偏移必须是alignment的整数倍。例如在iPhone7上Int占8个字节，那么在类/结构体中Int型参数的起始位置必须是8的整数倍（可认为类/结构体第一个成员变量的内存起始位置为0），后面会用实例说明。

2、实例成员变量size和静态方法size(ofValue: T)：

得到一个T数据类型实例占用连续内存字节的大小。

3、实例成员变量stride和静态方法 stride(ofValue: T)：

在一个T类型的数组中，其中任意一个元素从开始地址到结束地址所占用的连续内存字节的大小就是stride。如图：



加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

brycegao321 (http://blog...)

+ 关注

(http://blog.csdn.net/brycegao321)

码云

1

(https://g

utm_sourc

149

19

18

碧桂园珊瑚宫殿

广告

- 他的最新文章
- 更多文章 (http://blog.csdn.net/brycegao321)
- android apk编译打包过程 (http://blog.csdn.net/brycegao321/article/details/79127159)
- Java CAS(Compare And Swap)的一点理解 (http://blog.csdn.net/brycegao321/article/details/79099637)
- HTTPS的一点思考 (http://blog.csdn.net/brycegao321/article/details/79076290)
- Activity的生命周期是谁调用的？ (http://blog.csdn.net/brycegao321/article/details/79061805)
- Xcode小技巧-使用ImageLiteral、ColorLiteral智能选取图片/颜色 (http://blog.csdn.net/brycegao321/article/details/79009292)

- 文章分类
- Java (http://blog.csdn.net/b...) 24篇
- Android (http://blog.csdn.ne...) 44篇
- 设计模式 (http://blog.csdn.n...) 4篇
- Swift基础 (http://blog.csdn...) 52篇
- iOS (http://blog.csdn.net/br...) 29篇
- 注册
- 登录

注释：数组中有四个 T 类型元素，虽然每个 T 元素的大小为 size 个字节，但是因为需要内存对齐的限制，每个 T 类型元素实际消耗的内存空间为 stride 个字节，而 stride - size 个字节则为每个元素因为内存对齐而浪费的内存空间。

所以，一个对象或变量占用的空间是由本身大小和偏移组成的！我们改变不了数据类型本身的大小，但我们可以尽量的缩小偏移，后面会讲怎么做！

下面用几个实例说明：

```
0
[Java]
01. class People1: NSObject{
02.     var name: String?
03. }
04.
05. class People2: NSObject{
06.     var name: String?
07.     var age: Int?
08. }
09. let people1 = People1()
10. let people2 = People2()
```

people1和people2占用多大内存？？

别晕！这是类对象的引用，而引用占用的内存空间是固定的，即people1和people2占用内存大小相同，区别是指向的内存空间占用大小不同！

如果将类改成结构体会怎样？

```
[Java]
01. struct People1 {
02.     var name: String?
03. }
04.
05. struct People2 {
06.     var name: String?
07.     var age: Int?
08. }
09. let people1 = People1(name: "zhangsan")
10. let people2 = People2(name: "zhangsan", age: 1)
```

结构体是值类型，在iPhone7上Int占8个字节，String占24个字节；所以people1占用24个字节，people2占用32个字节。

类/结构体的成员声明顺序会影响占用空间，原理是变量要以自身类型的alignet整数倍作为起始地址，不足的话要在前面补齐字节（即内存碎片）。下面示例说明：结构体People1和People2的参数相同，区别是先后顺序不一致。

```
[Java]
01. //在iPhone7上占16个字节
02. struct People2 {
03.     var enable = false //占用1个字节
04.     var age = 1 //Int占8个字节，因为是8字节对齐，必须以8的整数倍开始，所以前面要补齐7个字节偏移。
05. }
06. //在iPhone7上占9个字节
07. struct People3 {
08.     var age = 1
09.     var enable = false //alignet是1，所以不要添加偏移
10. }
11. print("People2: size=\(MemoryLayout<People2>.size) align=\(MemoryLayout<People2>.alignment) stride=\(MemoryLayout<People2>.stride)")
12.
13. print("People3: size=\(MemoryLayout<People3>.size) align=\(MemoryLayout<People3>.alignment) stride=\(MemoryLayout<People3>.stride)")
```

输出：

People2: size=16 align=8 stride=16

People3: size=9 align=8 stride=16

加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

展开 v (https://passp...

文章存档

| | |
|--------------------------------|-----|
| 2018年1月 (http://blog.csdn....) | 6篇 |
| 2017年12月 (http://blog.csdn...) | 8篇 |
| 2017年11月 (http://blog.csdn...) | 11篇 |
| 2017年10月 (http://blog.csdn...) | 3篇 |
| 2017年9月 (http://blog.csdn....) | 8篇 |

展开 v

他的热门文章

Android5.0以上app进程保活的正确姿势 (http://blog.csdn.net/brycegao321/article/details/52312030)

10778

Android EditText限制输入字符的5种实现方式 (http://blog.csdn.net/brycegao321/article/details/52277255)

10379

OkHttp实现多线程并发下载的笔记 (http://blog.csdn.net/brycegao321/article/details/52131932)

8145

解决Android里getActivity()为空的问题 (http://blog.csdn.net/brycegao321/article/details/52061853)

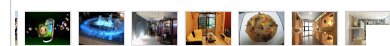
6010

Swift3.0 GCD多线程详解 (http://blog.csdn.net/brycegao321/article/details/53898919)

5905



团队拓展



联系我们



请扫描二维码联系客服

webmaster@csdn.net (mailto:webmaster@csdn.net)

400-660-0108

网站客服 (http://wpa.qq.com/msgrv?3&uin=2431299880&siteid=123456789)

关于 招聘 广告服务 阿里云

©2018 CSDN 京ICP证09002463号

(http://www.miibeian.gov.cn/)

登录

注册

×

Optional即可选数据类型会增加1个字节空间， 由于内存对齐的原因， Optional可能占用更多的内存空间。下面以Int为例：

```
[java]
01. print("Int:  size=\(MemoryLayout<Int>.size) align=\(MemoryLayout<Int>.alignment) stride=\(MemoryLayout<Int>.stride)")
02. print("Optional Int:  size=\(MemoryLayout<Optional<Int>>.size) align=\(MemoryLayout<Optional<Int>>.alignment) stride=\(MemoryLayout<Optional<Int>>.stride)")
```

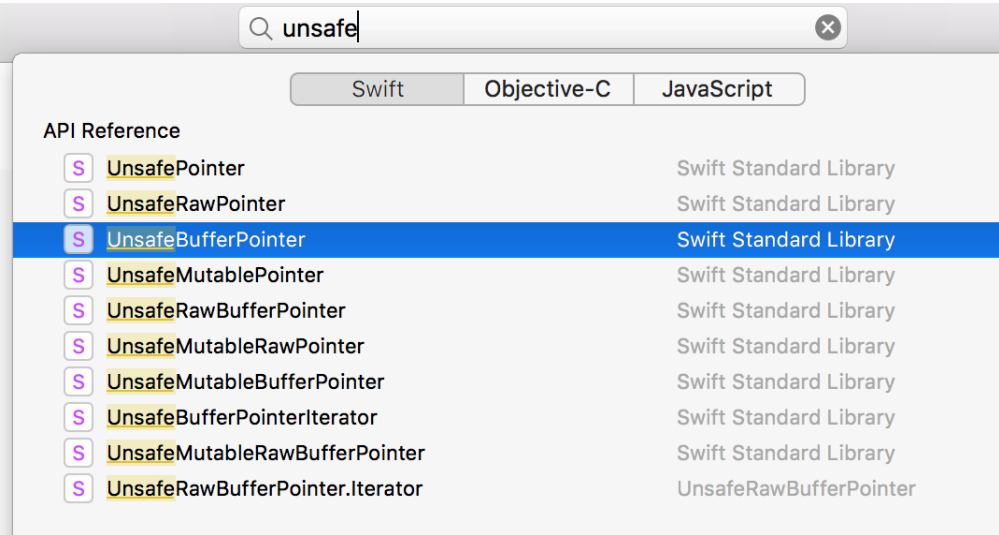
Int: size=8 align=8 stride=8
Optional Int: size=9 align=8 stride=16
Optional Int的size是9， Int的size是8。

空类和空结构体占多大空间呢？

```
[java]
01. class EmptyClass {
02.     // 占用一个引用的大小
03. }
04. struct EmptyStruct {
05.     // 占1个字节，因为需要唯一的地址
06. }
07. print("EmptyClass:  size=\(MemoryLayout<EmptyClass>.size) align=\(MemoryLayout<EmptyClass>.alignment) stride=\(MemoryLayout<EmptyClass>.stride)")
08.
09. print("EmptyStruct:  size=\(MemoryLayout<EmptyStruct>.size) align=\(MemoryLayout<EmptyStruct>.alignment) stride=\(MemoryLayout<EmptyStruct>.stride)")
```

EmptyClass: size=8 align=8 stride=8
EmptyStruct: size=0 align=1 stride=1
EmptyClass占用8个字节(跟引用占用空间大小相等)， 即使在EmptyClass添加几个成员变量， 得到的size仍然是8， 其实这里实际上测量的是引用； EmptyStruct的size为0但stride为1， 说明占用1个字节空间， 因为每个实例都需要唯一的地址。
如果你想知道类到底占用多大内存， 那么你可以尝试改为结构体后测量一下！ 因为你测量的是类的引用。

相信你对Swift内存占用情况有了一定的理解， 现在说说如何篡改内存。Swift提供了UnsafePointer类操作指针（还记得Java怎样操作指针吗？看我前面的博客）， iOS不负责UnsafePointer指向内存的回收（如果调用了allocate方法， 则要调用deinitialize和dealloc方法释放内存）， 所有在使用它时要注意回收内存。下面是Swift的所有指针操作类：



如果你想操作类/结构体的内存， 可以继承于PropertiesMetricizable或对应函数。 下面代码摘自HandyJSON：

```
[java]
01. extension PropertiesMetricizable {
02.
03.
04. mutating func headPointerOfStruct() -> UnsafeMutablePointer<Byte> {
```

经营性网站备案信息 (https://passp
(http://www.hd315.gov.cn/beian/view.asp?bianhao=010202001032100010)
网络110报警服务 (http://www.cyberpolice.cn/)
中国互联网举报中心 (http://www.12377.cn/)
北京互联网违法和不良信息举报中心 (http://www.bjjubao.org/)

(https://passp

```

05.
06.         return withUnsafeMutablePointer(to: &self) {
07.             return UnsafeMutableRawPointer($0).bindMemory(to: Byte.self, capacity: MemoryLayout
08.         }
09.     }
10.
11.     // locating the head of a class type object in memory
12.     mutating func headPointerOfClass() -> UnsafeMutablePointer<Byte> {
13.
14.         let opaquePointer = Unmanaged.passUnretained(self as AnyObject).toOpaque()
15.         let mutableTypedPointer = opaquePointer.bindMemory(to: Byte.self, capacity: MemoryLayout
16.         return UnsafeMutablePointer<Byte>(mutableTypedPointer)
17.     }
18.
19.     // memory size occupy by self object
20.     static func size() -> Int {
21.         return MemoryLayout<Self>.size
22.     }
23.
24.     // align
25.     static func align() -> Int {
26.         return MemoryLayout<Self>.alignment
27.     }
28.
29.     // Returns the offset to the next integer that is greater than
30.     // or equal to Value and is a multiple of Align. Align must be
31.     // non-zero.
32.     static func offsetToAlignment(value: Int, align: Int) -> Int {
33.         let m = value % align
34.         return m == 0 ? 0 : (align - m)
35.     }
36. }

```

使用结构体亲测一下篡改私有变量（原理：拿到对象头指针、判断出成员变量的偏移和占用内存大小，然后写内存）：

```

[Java]
01. //在iPhone7上测试
02. struct Pig {
03.     private var count = 4    //8字节
04.     var name = "Tom"        //24字节
05.
06.     //返回指向 Pig 实例头部的指针
07.     mutating func headPointerOfStruct() -> UnsafeMutablePointer<Int8> {
08.         return withUnsafeMutablePointer(to: &self) {
09.             return UnsafeMutableRawPointer($0).bindMemory(to: Int8.self, capacity: MemoryLayout
10.         }
11.     }
12.
13.     func printA() {
14.         print("Animal a:\(count)")
15.     }
16. }
17.
18. var pig = Pig()
19. let pigPtr: UnsafeMutablePointer<Int8> = pig.headPointerOfStruct() //头指针，类型同
const void *
20. //有了头指针，还需要知道每个变量的偏移位置和大小才可以修改内存
21. let rawPtr = UnsafeMutableRawPointer(pigPtr) //转换指针 类型同void *
22. let aPtr = rawPtr.advanced(by: 0).assumingMemoryBound(to: Int.self) //advanced函数时
字节偏移，assumingMemoryBound是内存大小
23. print("修改前: \(aPtr.pointee)") //4
24. pig.printA() //count等于4
25. aPtr.initialize(to: 100) //将count参数修改为100，即篡改了私有成员
26. print("修改后: \(aPtr.pointee)") //100
27. pig.printA()

```

输出：
修改前：4

Animal a:4

修改后：100

(https://passp

Animal a:100

类是引用类型，实例是在Heap堆区域里，而Stack栈里只是存放了指向它的指针；而Swift是ARC即自动回收的，类相比于结构体需要额外的内存空间用于存放类型信息和引用计数。在32bit机型上类型信息占4个字节，在64bit机型上类型信息占8字节；引用计数占8字节。考虑到内存对齐原因，类属性总是从16字节开始。

要修改类成员变量，跟上面介绍的结构体类似，但成员起始位置是第16个字节，因为类型信息和引用计数占用的内存空间在类成员属性前面。将Pig修改为类，对比一下：

```
[java]
01. //在iPhone7上测试
02. class Pig {
03.     private var count = 4 //8字节
04.     var name = "Tom" //24字节
05.
06.     // 得到Pig对象在堆内存的首位置
07.     func headPointerOfClass() -> UnsafeMutablePointer<Int8> {
08.
09.         let opaquePointer = Unmanaged.passUnretained(self as AnyObject).toOpaque()
10.         let mutableTypedPointer = opaquePointer.bindMemory(to: Int8.self, capacity: MemoryLayout<Int8>.size)
11.         return UnsafeMutablePointer<Int8>(mutableTypedPointer)
12.     }
13.
14.     func printA() {
15.         print("Animal a:\(count)")
16.     }
17. }
18. var pig = Pig()
19. let pigPtr: UnsafeMutablePointer<Int8> = pig.headPointerOfClass() //头指针，类型同
const void *
20. //有了头指针，还需要知道每个变量的偏移位置和大小才可以修改内存
21. let rawPtr = UnsafeMutableRawPointer(pigPtr) //转换指针 类型同void *
22.
23. //在iPhone7上类型信息和引用计数参数占用16个字节，类成员属性相对起始位置要偏移16个字节
24. let aPtr = rawPtr.advanced(by: 16).assumingMemoryBound(to: Int.self) //advanced函数时字节偏移，
assumingMemoryBound是内存大小
25. print("修改前: \(aPtr.pointee)") //4
26. pig.printA() //count等于4
27. aPtr.initialize(to: 100) //将count参数修改为100，即篡改了私有成员
28. print("修改后: \(aPtr.pointee)") //100
29. pig.printA()
```

输出：

修改前：4

Animal a:4

修改后：100

Animal a:100

结构体和类对象在篡改内存数据时，结构体成员参数起始偏移为0，类成员参数起始偏移为类型信息和引用计数占用空间之和。

直接操作内存是高阶玩法，要判断当前机型CPU的位数（即需要适配），然后要理解内存模型。出个小题考验一下你是否理解了Swift内存模型：

```
[java]
01. struct Point {
02.     var a: Double?
03.     var b = 0
04. }
```

从内存角度考虑，Point结构体有什么问题？

如果你没懵逼，那么恭喜你已经掌握了Swift内存模型原理。老司机应该这样写：加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录 注册 X

```
01. struct Point {
02.     var b = 0
03.     var a: Double?
04. }
```

(https://passp

理由：因为Optional会多占1个字节， 第一种写法后面的Int型参数b会先内存对齐，然后再分配内存，即多占了一个Int型空间（PS：要减1）。

0

参考：

- Swift 对象内存模型探究（一）
(https://mp.weixin.qq.com/s/zlkB9KnAt1YPWGOOwyqY3Q)
- Swift进阶之内存模型和方法调度
(http://blog.csdn.net/Hello_Hwc/article/details/53147910)
- Swift操作指针详解 (http://www.jb51.net/article/103538.htm)



s123456slll (/s123456slll) 2017-07-26 19:43 回复 1楼
(/s123456slll)

使用Swift 字典模型互转 就是这么简单

hejunmeng 2016年04月29日 11:34 7734

写在面前的话现在很多iOS项目的开发开始转向Swift语言。相信 Swift语言很快会成为iOS工程师 必备技能。字典转模型，模型转转字典在开发过程中扮演非常重要的角色。今天就和大家分享一下...
(http://blog.csdn.net/hejunmeng/article/details/51280570)

Swift – 不借助第三方库转模型

qq_26598821 2016年07月06日 15:02 2166

swift – 不借助第三方库转模型 在写代码的时候为了更好的联想到模型中的属性，习惯在属性前面加上前缀。...
(http://blog.csdn.net/qq_26598821/article/details/51839009)

2018年，想要涨薪50%，应该从哪方面准备？



都知道开春是涨薪的好时候，如何快速搭上涨薪的快车，老司机教你怎么跟老板硬气的要工资！

(https://edu.csdn.net/promotion_activity?id=1?utm_source=blog10-1)

Swift 开发三方库收集


C_calary 2017年05月26日 15:32 564

前言一个APP的诞生肯定少不了站在巨人的肩膀上，所以使用这些开源的库，可以让你的开发更加的顺利，快速。如果你想获得更好的阅读体验，请移步简书。网络请求 Alamofire (Swift) 封装好的...
(http://blog.csdn.net/C_calary/article/details/72779514)


加入CSDN，享受更精准的内容推荐，与500万程序员共同成长！

登录 注册 X

Swift-封装购物车Model 数据模型

 u012701023

2015年12月10日 16:30


 1756

(https://passp


import UIKit class QHGoodModel: NSObject { //是否已经加入购物车 var alreadyAddShoppingCart: Bo...

(http://blog.csdn.net/u012701023/article/details/50251331)

swift对象存储

 ldw220817


2016年05月23日 12:18

 10944


swift对象存储简介OpenStack Object Storage(Swift)是OpenStack开源云计算项目的子项目之一，被称为对象存储，提供了强大的扩展性、冗余和持久性。对象存储，用于永久...

(http://blog.csdn.net/ldw220817/article/details/51480712)

Swift进阶之内存模型和方法调度

 Richard_Jason


2016年11月16日 16:34

 641


前言 Apple今年推出了Swift3.0，较2.3来说，3.0是一次重大的升级。关于这次更新，在这里都可以找到，最主要的还是提高了Swift的性能，优化了Swift API的设计（命名）规范。...

(http://blog.csdn.net/Richard_Jason/article/details/53188058)

Swift 对象内存模型探究

 mazegong


2017年05月12日 16:51

 601


HandyJSON 是 Swift 处理 JSON 数据的开源库之一，类似 JOSNModel，它可以直接将 JSON 数据转化为类实例在代码中使用。由于 Swift 是一种静态语言，没有 O...

(http://blog.csdn.net/mazegong/article/details/71747317)

Swift 对象内存模型探究（一）

 Tencent_Bugly



2017年05月15日 14:31

 1007

本文来自于腾讯Bugly公众号（weixinBugly），未经作者同意，请勿转载，原文地址：https://mp.weixin.qq.com/s/zlkB9KnAt1YPWGOOwyqY3Q作者：王振...


(http://blog.csdn.net/Tencent_Bugly/article/details/72145868)

Swift开发必备技巧：内存管理、weak和unowned


因为 Playground 本身会持有所有声明在其中的东西，因此本节中  u013406800 2017年02月18日 09:22  1376 的示例代码需要在 Xcode 项目环境中运行。在 Playground 中可能无法得到正确的结果。不管在什么语言里，内存管...

(http://blog.csdn.net/u013406800/article/details/55653854)

Swift进阶之内存模型和方法调度

 Hello_Hwc

2016年11月13日 16:08


 5126

前言Apple今年推出了Swift3.0，较2.3来说，3.0是一次重大的升级。关于这次更新，在这里都可以找到，最主要的还是提高了Swift的性能，优化了Swift API的设计（命名）规范。前段时间...


(http://blog.csdn.net/Hello_Hwc/article/details/53147910)

数据模型使用（ios），浅谈ios和swift数据模型使用，set和get方法使用

浅谈ios中oc和swift数据模型的使用 开发工具：xcod7.3 简单理解oc和swift数据模型的使用。 源码：swiftDemo ocDemo 使用swi...


 flyToSky_L

2017年03月09日 15:04


 1083

(http://blog.csdn.net/flyToSky_L/article/details/60960640)

swift 3.0踏坑之旅 ---- 自带模型转换

 u012877287


2017年03月20日 16:27

 565


在OC中用惯了JSONModel，养成了一个习惯：所有的字段都用string接。无疑，在OC 中，继承自JSONModel的模型里的属性都定义成string会避免一些空值时的报错，以及一些不是每...

(http://blog.csdn.net/u012877287/article/details/64127806)

Swift的74个常用内置函数介绍

 banma2008

2015年06月04日 14:01

 1340

Swift包含了74个内置函数，但在 The Swift Programming Language 一书中只介绍了其中的7个，其它的都没有在文档中体现。这篇文章列举出了所有的Swift库函数。又...

(http://blog.csdn.net/banma2008/article/details/46360333)

(https://passp

程序员不会英语怎么行？

老司机教你一个数学公式秒懂天下英语



Swift的自动引用计数->解决内存泄露



qq_14920635 2016年05月24日 12:05 2119

Swift的自动引用计数->解决内存泄露 在swift中的变量一般分为三种： 1、 strong 强引用 默认 2、 weak 弱引用 定义时前面加 we...

(http://blog.csdn.net/qq_14920635/article/details/51489124)

Swift 对象内存模型探究



mazegong 2017年05月12日 16:51 601

HandyJSON 是 Swift 处理 JSON 数据的开源库之一，类似 JOSNModel，它可以直接将 JSON 数据转化为类实例在代码中使用。由于 Swift 是一种静态语言，没有 O...

(http://blog.csdn.net/mazegong/article/details/71747317)

Swift进阶之内存模型和方法调度



Hello_Hwc 2016年11月13日 16:08 5126

前言Apple今年推出了Swift3.0，较2.3来说，3.0是一次重大的升级。关于这次更新，在这里都可以找到，最主要的还是提高了Swift的性能，优化了Swift API的设计（命名）规范。前段时间...

(http://blog.csdn.net/Hello_Hwc/article/details/53147910)

Swift 对象内存模型探究（一）



Tencent_Bugly 2017年05月15日 14:31 1007

本文来自于腾讯Bugly公众号（weixinBugly），未经作者同意，请勿转载，原文地址：https://mp.weixin.qq.com/s/zlkB9KnAt1YPWGOOwyqY3Q作者：王振...

(http://blog.csdn.net/Tencent_Bugly/article/details/72145868)

Swift – 不借助第三方库转模型



qq_26598821 2016年07月06日 15:02 2166

swift – 不借助第三方库转模型 在写代码的时候为了更好的联想到模型中的属性，习惯在属性前面加上前缀。...

(http://blog.csdn.net/qq_26598821/article/details/51839009)

swift 字典转模型框架



weixin_35755389 2016年10月13日 16:41 632

swift 字典转模型框架

(http://blog.csdn.net/weixin_35755389/article/details/52807927)

使用Swift 字典模型互转 就是这么简单



hejunmeng 2016年04月29日 11:34 7734

写在前面的话现在很多iOS项目的开发开始转向Swift语言。相信 Swift语言很快会成为iOS工程师 必备技能。字典转模型，模型转转字典在开发过程中扮演非常重要的角色。今天就和大家分享一下...

(http://blog.csdn.net/hejunmeng/article/details/51280570)