

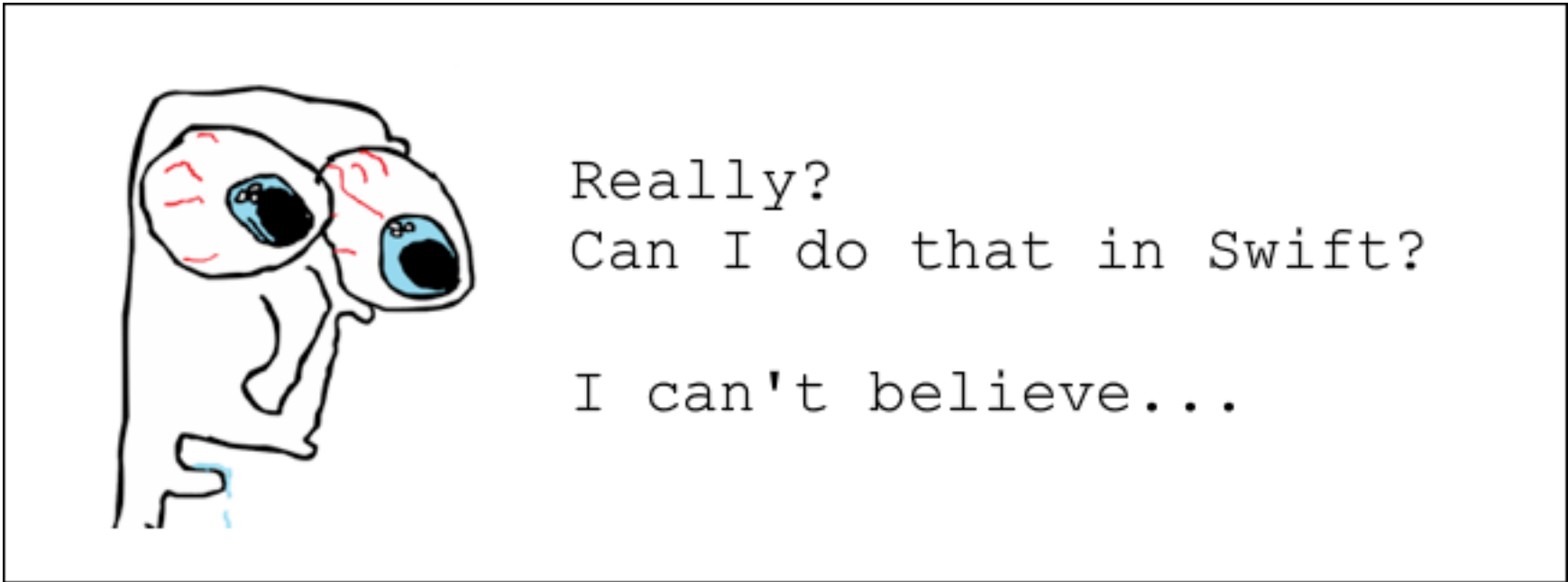
道长的 Swift 面试题

 故胤道长

★

关注

🔥 2 2016.12.15 13:36:48 字数 1,151 阅读 19,274



1. 给一个数组，要求写一个函数，交换数组中的两个元素

- 二X程序员：
好简单啊，直接写出以下结果

```
1 func swap(_ nums: inout [Int], _ p: Int, _ q: Int) {
2     let temp = nums[p]
3     nums[p] = nums[q]
4     nums[q] = temp
5 }
```

- 普通程序员：
首先跟面试官沟通，是什么类型的数组？面试官会说，任意。普通程序员微微一笑，写出以下代码

```
1 func swap<T>(_ nums: inout [T], _ p: Int, _ q: Int) {
2     let temp = nums[p]
3     nums[p] = nums[q]
4     nums[q] = temp
5 }
```

- 文艺程序员：
与面试官沟通，是什么类型的数组？有什么其他要求和限制？面试官会说，这是一个Swift面试题。文艺程序员心领神会，于是写出以下答案

```
1 func swap<T>(_ nums: inout [T], _ p: Int, _ q: Int) {
2     (nums[p], nums[q]) = (nums[q], nums[p])
3 }
```

同时对以上代码写上相应测试，检测各种边界情况，再确认无误后，才会说，这道题目我完成了。

百度智能云

2021

云

低至

消费满

 故胤道长

总资产295 (约

百度智能云

2021

热

1

消费满

2. 实现一个函数，输入是任一整数，输出要返回输入的整数 + 2

这道题很多人上来就这样写：

```
1 func addTwo(_ num: Int) -> Int {
2     return num + 2
3 }
```

接下来面试官会说，那假如我要实现 + 4 呢？程序员想了一想，又定义了另一个方法：

```
1 func addFour(_ num: Int) -> Int {
2     return num + 4
3 }
```

这时面试官会问，假如我要实现返回 + 6, + 8 的操作呢？能不能只定义一次方法呢？正确的写法是利用 Swift 的 Currying 特性：

```
1 func add(_ num: Int) -> (Int) -> Int {
2     return { val in
3         return num + val
4     }
5 }
6
7 let addTwo = add(2), addFour = add(4), addSix = add(6), addEight = add(8)
```

3. 精简以下代码

```
1 func divide(dividend: Double?, by divisor: Double?) -> Double? {
2     if dividend == nil {
3         return nil
4     }
5     if divisor == nil {
6         return nil
7     }
8     if divisor == 0 {
9         return nil
10    }
11    return dividend! / divisor!
12 }
```

这题考察的是 `guard let` 语句以及 optional chaining，最佳答案是

```
1 func divide(dividend: Double?, by divisor: Double?) -> Double? {
2     guard let dividend = dividend, let divisor = divisor, divisor != 0 else {
3         return nil
4     }
5
6     return dividend / divisor
7 }
```

4. 以下函数会打印出什么？

```
1 var car = "Benz"
2 let closure = { [car] in
3     print("I drive \(car)")
4 }
5 car = "Tesla"
6 closure()
```

因为 clousre 已经申明将 car 复制进去了 ([car])，此时clousre 里的 car 是个局部变量，不再与外面的 car有关，所以会打印出"I drive Benz"。

此时面试官微微一笑，将题目略作修改如下：

```
1 | var car = "Benz"
2 | let closure = {
3 |   print("I drive \(car)")
4 | }
5 | car = "Tesla"
6 | closure()
```

此时 closure 没有申明复制拷贝 car，所以clousre 用的还是全局的 car 变量，此时将会打印出 "I drive Tesla"

5. 以下代码会打印出什么？

```
1 | protocol Pizzeria {
2 |   func makePizza(_ ingredients: [String])
3 |   func makeMargherita()
4 | }
5 |
6 | extension Pizzeria {
7 |   func makeMargherita() {
8 |     return makePizza(["tomato", "mozzarella"])
9 |   }
10 | }
11 |
12 | struct Lombardis: Pizzeria {
13 |   func makePizza(_ ingredients: [String]) {
14 |     print(ingredients)
15 |   }
16 |   func makeMargherita() {
17 |     return makePizza(["tomato", "basil", "mozzarella"])
18 |   }
19 | }
20 |
21 | let lombardis1: Pizzeria = Lombardis()
22 | let lombardis2: Lombardis = Lombardis()
23 | lombardis1.makeMargherita()
24 | lombardis2.makeMargherita()
```

答案：打印出如下两行

["tomato", "basil", "mozzarella"]

["tomato", "basil", "mozzarella"]

在Lombardis的代码中，重写了makeMargherita的代码，所以永远调用的是Lombardis 中的 makeMargherita。

再进一步，我们把 protocol Pizzeria 中的 func makeMargherita() 删掉，代码变为

```
1 | protocol Pizzeria {
2 |   func makePizza(_ ingredients: [String])
3 | }
4 |
5 | extension Pizzeria {
6 |   func makeMargherita() {
7 |     return makePizza(["tomato", "mozzarella"])
8 |   }
9 | }
10 |
11 | struct Lombardis: Pizzeria {
12 |   func makePizza(_ ingredients: [String]) {
13 |     print(ingredients)
14 |   }
```

```
15     func makeMargherita() {
16         return makePizza(["tomato", "basil", "mozzarella"])
17     }
18 }
19
20 let lombardis1: Pizzeria = Lombardis()
21 let lombardis2: Lombardis = Lombardis()
22 lombardis1.makeMargherita()
23 lombardis2.makeMargherita()
```

这时候打印出如下结果：

["tomato", "mozzarella"]

["tomato", "basil", "mozzarella"]

因为lombardis1 是 Pizzeria，而 makeMargherita() 有默认实现，这时候我们调用默认实现。

6. Swift 中定义常量和 Objective-C 中定义常量有什么区别？

一般人会觉得没有差别，因为写出来好像也确实没差别。

OC是这样定义常量的：

```
1 | const int number = 0;
```

Swift 是这样定义常量的：

```
1 | let number = 0
```

首先第一个区别，OC中用 const 来表示常量，而 Swift 中用 let 来判断是不是常量。

上面的区别更进一步说，OC中 const 表明的常量类型和数值是在 compilation time 时确定的；

而 Swift 中 let 只是表明常量（只能赋值一次），其类型和值既可以是静态的，也可以是一个动态的计算方法，它们在 runtime 时确定的。

7. Swift 中 struct 和 class 什么区别？举个应用中的实例

struct 是值类型，class 是引用类型。

看过WWDC的人都知道，struct 是苹果推荐的，原因在于它在小数据模型传递和拷贝时比 class 要更安全，在多线程和网络请求时尤其好用。我们来看一个简单的例子：

```
1 | class A {
2 |     var val = 1
3 | }
4 |
5 | var a = A()
6 | var b = a
7 | b.val = 2
```

此时 a 的 val 也被改成了 2，因为 a 和 b 都是引用类型，本质上它们指向同一内存。解决这个问题的方法就是使用 struct：

```
1 | struct A {
2 |     var val = 1
3 | }
4 |
5 | var a = A()
6 | var b = a
7 | b.val = 2
```


此时 A 是struct，值类型，b 和 a 是不同的东西，改变 b 对于 a 没有影响。

8. Swift 到底是面向对象还是函数式的编程语言？

Swift 既是面向对象的，又是函数式的编程语言。

说 Swift 是 Object-oriented，是因为 Swift 支持类的封装、继承、和多态，从这点上来看与 Java 这类纯面向对象的语言几乎毫无差别。

说 Swift 是函数式编程语言，是因为 Swift 支持 map, reduce, filter, flatmap 这类去除中间状态、数学函数式的方法，更加强调运算结果而不是中间过程。

203人点赞 >

故胤道长

卡内基梅隆大学硕士毕业，常年居住于美国的 iOS 开发者。 现 Q...
总资产295 (约28.32元) 共写了8.3W字 获得3,896个赞 共7,353个粉丝

关注

近视1300度

面试培训班

舆情监测平台

前十名婚纱照

面试口语

广告

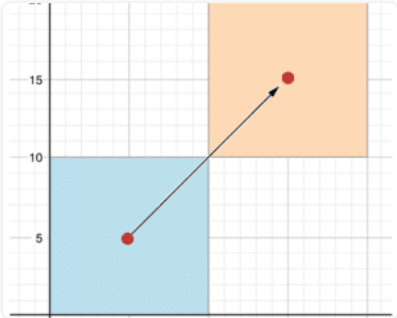
推荐阅读

更多精彩内容 >

Swift 4.0 编程语言（四）

109.关联值 上部分实例展示枚举分支是一个定义值（类型值）。 你可以把一个常量或者变量设置成 Planet.ea...

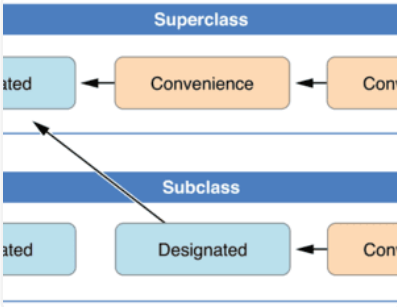
无泮 阅读 436 评论 0 赞 3



Swift 4.0 编程语言（五）

123.继承 一个类可以从另外一个类继承方法,属性和其他特征。当一个类继承另外一个类时, 继承类叫子类, 被继承的...

无泮 阅读 476 评论 2 赞 3



swift3.0 基础知识点

```
importUIKit classViewController:UITabBarController{ enumD...
```

明哥_Young 阅读 862 评论 1 赞 3

《迟》文/箬竹


你总是咀嚼着农事 没日没夜的忙 你对我 总是一副满脸不在乎的样子 你总是把眉头紧锁把关爱隐藏 多少次我也怀疑过你 ...

箬竹_泠萱 阅读 57 评论 2 赞 5



陆小曼：一个敢爱敢恨的民国女子

茶余饭后，谈起民国美女会让你想起周璇和阮玲玉；谈起民国才女会让你想起林徽因、苏雪林、张爱玲、蒋碧微、萧红和许广平。...

 Cat猫眼 阅读 1,016 评论 7 赞 19

