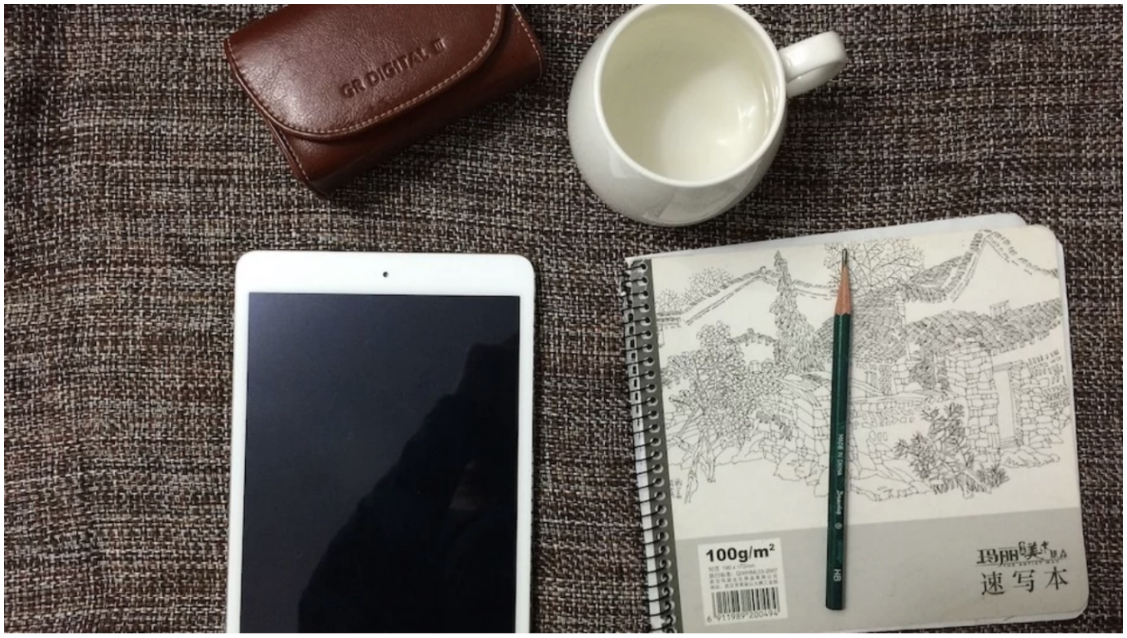


# 出一套 iOS 高级面试题

一千个读者眼中有一千个哈姆雷特，一千名 iOS 程序员心目中就有一千套 iOS 高级面试题。本文就是笔者认为可以用来面试高级 iOS 程序员的面试题。

这套题的题目跟公司和业务都没有关系，而且也并不代表笔者本人可以把这些题回答得非常好，笔者只是将一部分觉得比较好的题从收集的面试题里面抽出来了而已。

收集的面试题有以下三个来源：

1. 笔者在准备面试的过程中搜集并整理过的面试题。
2. 笔者在准备面试的过程中自己思考过的新题。
3. 笔者在面试过程中遇到的觉得比较好的题。

本文分为三个部分展开：

1. 面试题
2. 喜欢考察的和不喜欢考察的题
3. 建议准备数据结构和算法题

3. 建议准备数据结构和算法题

第一部分就是面试题了；第二部分给出了笔者喜欢考察和不喜欢考察的题以及原因；第三部分是笔者建议大家准备数据结构和算法题的原因。

# 面试题

## iOS 基础题

1. 分类和扩展有什么区别？可以分别用来做什么？分类有哪些局限性？分类的结构体里面有哪些成员？
2. 讲一下atomic的实现机制；为什么不能保证绝对的线程安全（最好可以结合场景来说）？
3. 被weak修饰的对象在被释放的时候会发生什么？是如何实现的？知道sideTable么？里面的结构可以画出来么？
4. 关联对象有什么应用，系统如何管理关联对象？其被释放的时候需要手动将所有的关联对象的指针置空么？
5. KVO的底层实现？如何取消系统默认的KVO并手动触发（给KVO的触发设定条件：改变的值符合某个条件时再触发KVO）？
6. `Autoreleasepool`所使用的数据结构是什么？`AutoreleasePoolPage`结构体了解么？
7. 讲一下对象，类对象，元类，跟元类结构体的组成以及他们是如何相关联的？为什么对象方法没有保存的对象结构体里，而是保存在类对象的结构体里？
8. `class_ro_t` 和 `class_rw_t` 的区别？
9. iOS 中内省的几个方法？`class`方法和`objc_getClass`方法有什么区别？
10. 在运行时创建类的方法`objc_allocateClassPair`的方法名尾部为什么是pair（成对的意思）？
11. 一个int变量被`__block`修饰与否的区别？
12. 为什么在block外部使用`__weak`修饰的同时需要在内部使用`__strong`修饰？
13. RunLoop的作用是什么？它的内部工作机制了解么？（最好结合线程和内存管理来说）
14. 哪些场景可以触发离屏渲染？（知道多少说多少）

## iOS 实战题

1. AppDelegate如何瘦身?
2. 反射是什么？可以举出几个应用场景么？（知道多少说多少）
3. 有哪些场景是NSOperation比GCD更容易实现的？（或是NSOperation优于GCD的几点，知道多少说多少）
4. App 启动优化策略？最好结合启动流程来说（main()函数的执行前后都分别说一下，知道多少说多少）
5. App 无痕埋点的思路了解么？你认为理想的无痕埋点系统应该具备哪些特点？（知道多少说多少）
6. 你知道有哪些情况会导致app崩溃，分别可以用什么方法拦截并化解？（知道多少说多少）
7. 你知道有哪些情况会导致app卡顿，分别可以用什么方法来避免？（知道多少说多少）

# 网络题

1. App 网络层有哪些优化策略?
2. TCP为什么要三次握手，四次挥手?
3. 对称加密和非对称加密的区别？分别有哪些算法的实现?
4. HTTPS的握手流程？为什么密钥的传递需要使用非对称加密？双向认证了解么?
5. HTTPS是如何实现验证身份和验证完整性的?
6. 如何用Charles抓HTTPS的包？其中原理和流程是什么?
7. 什么是中间人攻击？如何避免?

# 计算机系统题

1. 了解编译的过程么？分为哪几个步骤?
2. 静态链接了解么？静态库和动态库的区别?
3. 内存的几大区域，各自的职能分别是什么?

3. 内存的几大区域，各自的职能分别是什么？
4. static和const有什么区别?
5. 了解内联函数么?
6. 什么时候会出现死锁？如何避免?
7. 说一说你对线程安全的理解?
8. 列举你知道的线程同步策略?
9. 有哪几种锁？各自的原理？它们之间的区别是什么？最好可以结合使用场景来说

# 设计模式题

1. 除了单例，观察者设计模式以外，还知道哪些设计模式？分别介绍一下
2. 最喜欢哪个设计模式？为什么?
3. iOS SDK 里面有哪些设计模式的实践?
4. **设计模式是为了解决什么问题的?
5. **设计模式的成员构成以及工作机制是什么?
6. **设计模式的优缺点是什么?

# 架构 & 设计题

1. MVC和MVVM的区别？MVVM和MVP的区别?
2. 面向对象的几个设计原则了解么？最好可以结合场景来说。
3. 可以说几个重构的技巧么？你觉得重构适合什么时候来做?
4. 你觉得框架和设计模式的区别是什么?
5. 看过哪些第三方框架的源码，它们是怎么设计的？设计好的地方在哪里，不好的地方在哪里，如何改进？（这道题的后三个问题的难度已经很高了，如果不是太N的公司不建议深究）

# 数据结构&算法题

1. 链表和数组的区别是什么？插入和查询的时间复杂度分别是多少?
2. 哈希表是如何实现的？如何解决地址冲突?
3. 排序题：冒泡排序、选择排序、插入排序、快速排序（二路、三路）能写出那

3. 排序题：冒泡排序，选择排序，插入排序，快速排序（二路，三路）能写出那些？
4. 链表题：如何检测链表中是否有环？如何删除链表中等于某个值的所有节点？
5. 数组题：如何在有序数组中找出和等于给定值的两个元素？如何合并两个有序的数组之后保持有序？
6. 二叉树题：如何反转二叉树？如何验证两个二叉树是完全相等的？

# 喜欢出的和不喜欢出的题

不难看出，整套面试题中的iOS部分占比其实并不大（三分之一），因为笔者认为：

> 高级 iOS 开发 = 高级开发 + （高级） iOS 开发。

而其中**高级开发**的部分应该作为优先考核的内容，目的在于首先要验证面试者**是否具备高级开发必备的基本素质**。这部分知识的掌握程度会直接影响一个开发者的研究和设计能力，包括横向和纵向的。而笔者个人觉得后面的**（高级） iOS 开发**的部分仅仅考查的是面试者对于 iOS 本身的理解程度（API，系统，开发工具等等）。

在这套里面，笔者个人最喜欢的几道题是：

1. iOS SDK 里面有哪些设计模式的实践？
2. 说一说你对线程安全的理解？
3. 你知道有哪些情况会导致app崩溃，分别可以用什么方法拦截并化解？
4. 看过哪些第三方框架的源码，它们是怎么设计的？
5. 可以说几个重构的技巧么？你觉得重构适合什么时候来做？

> 1. 这道题一箭双雕，不仅考察了面试者对设计模式这种通用性知识的了解，还可以考察其对iOS SDK的熟悉和思考程度。这里可以简单提几个：单例：UIApplication；观察者模式：KVO；类簇：NSNumber；装饰者模

> 式：分类；命令模式：NSInvocation；享元模式：UITableviewCell（UITableview的重用）。还有更多，有兴趣的读者可以看一下《Objective-C 编程之道》这本书，它介绍了很多在 iOS SDK中使用的设计模式。
> 2. 这道题我看到网上有些答案是错的，说的大概的意思是“同一时刻只有一个线程访问”。但是如果按照这个定义的话，那么那些无法改变的常量就不算是线程安全的了，所以显然这类定义就是错的。所以说学东西要具备批判性思维，尤其是看博客的时候，很多情况需要自己想想，主动去认证，去思考。
> 3. 导致app崩溃的原因有很多，比如向某个对象发送其无法响应的方法，数组越界，集合类中添加nil对象，string访问越界，KVO不合理的移除关联key（KVO导致的崩溃不仅仅这一种原因）等。而崩溃非常影响用户体验，所以笔者认为一名高级 iOS 开发应该具备避免这些崩溃的能力，起码至少也要知道这些容易导致崩溃的场景。
> 4. 看一些优秀开源框架的代码，梳理实现思路和细节可以帮助我们提高在类似场景下设计系统的能力。其实道理很简单，小时候学习写作文的办法是什么？- 就是背诵课文而已啊。因为写作是一种输出，所以如果你没有好词好句的积累（输入），自然写不出辞藻丰富的文章。写代码也是一样的道理~
> 5. 重构的能力是笔者非常看重的能力。其实笔者个人认为关于重构的技巧可以早早学习，在后面写代码的时候尽可能做到一步到位（如果在排期允许的情况下），而且也对设计代码方面能力的提高有帮助：怎样才能设计出一个低耦合，高内聚，易扩展，易修改的系统？有专门的一本书来介绍重构：《重构 改善既有代码的设计》。

上面说了笔者喜欢考察的问题，下面说一下笔者不喜欢考察的是哪些问题：

1. 如何查询线上的崩溃?
2. 了解发布流程么？几个证书的区别?
3. 有没有做过支付/地图/分享?
4. dysm文件是什么，有什么作用?

笔者不考察这类问题的原因有两个：

1. 这类问题考查不了面试者作为一名程序员的基本素质，因为其考察的内容仅仅局限于iOS本身。
2. 这类问题往往是“做过即知道”，再没办法量化能力，在实际开发中遇到了就做

2. 这类问题往往是“做过即知道”，更没办法量化能力。在实际开发中遇到了就做过了；就算没遇到，没做过，笔者也相信一名优秀的程序员在第一次也会高效地做好。

# 建议准备数据结构和算法题

在本文的最后说一下数据结构和算法题。

这类问题是比较大的公司喜欢考核的内容，也就是说大部分公司其实并不考（但是如果了解的话是会加分的）。但是笔者个人认为如果时间上允许，多少还是准备一些会比较好。除了应对面试，其实算法方面的学习会对编程能力的提高有帮助，这一点笔者自己深有体会：

笔者这次准备面试的过程中，在LeetCode上面刷了一些道题，其中链表，数组，二叉树的题加起来有30道左右，并把这些题放在了个人仓库里面：awesome-algorithm-question-solution。欢迎PR Swift，Java的算法题和答案~

在刷题和学习的过程中渐渐能够感觉到对代码的理解能力提高了很多，尤其是链表题可以强化对指针操作的理解，而且对执行条件的检查，边界问题的处理能力也提升了一些~

好了，这套题就分享到这里了，在文章后面也建议大家平时注意数据结构和算法方面的学习。和上一篇一样，这篇博客主观方面的内容还是多一些的，还是希望读者可以多多和我交流~