

< / > 在这里 / 有面试笔试常见技巧的提炼与总结;
< / > 在这里 / 有面试笔试高频iOS知识点的整理与剖析;
< / > 在这里 / 有面试笔试历年iOS真题的解答与拓展。

PROGRAMMER

> Interview and written examination

iOS

程序员 > 面试笔试宝典

猿媛之家 / 组编 蒋信厚 等 / 编著

98%

高频知识点

本书覆盖了近**3年**程序员面试笔试中超过**98%**的iOS高频知识点

当你细细品读完本书后，各类企业的offer将任由你挑选

一 书 在 手 / 工 作 不 愁 > .

 机械工业出版社
CHINA MACHINE PRESS



iOS 程序员面试笔试宝典

猿媛之家 组编

蒋信厚 等编著

机械工业出版社

本书针对当前各大 IT 企业面试笔试中的特性与侧重点,精心挑选了近 3 年多家顶级 IT 企业的 iOS 面试笔试真题,由于这些企业涉及业务包括系统软件、搜索引擎、电子商务、手机软件、安全关键软件等领域,所以面试笔试真题非常具有代表性与参考性。同时,本书对这些题目进行了合理的划分与归类,并进行了庖丁解牛式的分析与讲解,针对试题中涉及的部分重点、难点问题,本书都做了适当地扩展与延伸,力求对知识点的讲解清晰而不紊乱,全面而不啰唆。不仅如此,本书除了对 iOS 的基础语言知识进行深度剖析以外,还针对数据库、设计模式等相关知识的面试笔试做了非常详细的介绍。读者学习本书能够获取到求职的知识,更有针对性地进行求职准备,最终能够收获一份满意的工作。

本书是一本适合计算机相关专业毕业生面试笔试的求职用书,同时也适合期望在计算机软、硬件行业大显身手的计算机爱好者阅读。本书起于 iOS 面试笔试,但收获的不止面试笔试,还有更多的 iOS 实用操作技能。书中附有大量面试笔试真题,让读者能够最大限度地提升应试技能。

图书在版编目(CIP)数据

iOS 程序员面试笔试宝典 / 猿媛之家组编; 蒋信厚等编著. —北京: 机械工业出版社, 2018.9

ISBN 978-7-111-60927-8

I. ①i… II. ①猿… ②蒋… III. ①移动终端—应用程序—程序设计
IV. ①TN929.53

中国版本图书馆 CIP 数据核字 (2018) 第 212924 号

机械工业出版社 (北京市百万庄大街 22 号 邮政编码 100037)

策划编辑: 时 静 责任编辑: 时 静 王 荣

责任校对: 张艳霞 责任印制: 张 博

三河市国英印务有限公司印刷

2018 年 10 月第 1 版·第 1 次印刷

184mm×260mm·19 印张·462 千字

0001—3000 册

标准书号: ISBN 978-7-111-60927-8

定价: 69.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

电话服务

网络服务

服务咨询热线: (010) 88361066

机工官网: www.cmpbook.com

读者购书热线: (010) 68326294

机工官博: weibo.com/cmp1952

(010) 88379203

教育服务网: www.cmpedu.com

封面无防伪标均为盗版

金 书 网: www.golden-book.com

前言

本书是我读研期间开始着手起草，到交稿给出版社为止，整整耗时一年半。在此之前，我一直对 iOS 技术情有独钟，充满兴趣和学习热情。回想过去四五年的 iOS 学习历程，很庆幸当时的决心和之后的坚持，正是这种决心和坚持使我能收获一项自己最擅长的技能。

我从大二开始接触 iOS 开发，那时候 iOS 5 刚刚发布，iOS 技术刚火起来，而我已有的技术积累并不算多，所以学习难度非常大。当时，能够熟练开发 iOS 应用的同学真的是凤毛麟角，他们很让人敬佩，而这也更激发了我想学会 iOS 开发的欲望。为了学习 iOS 基础，我借阅了学校图书馆所有关于 Objective-C 的书，对于计算机基础还不扎实的我去自学 Objective-C 真的是很痛苦，各种 iOS 开发指南和开发案例的书也让我很吃力。好在那时候我做了一个正确的选择，用自己刚拿到的奖学金和攒下的钱毅然决然地买了一台低配的新款 Mac Pro 笔记本式计算机（就是 2013 年刚出视网膜屏幕且大大瘦身的那款，至今这台笔记本式计算机仍在我手中服役），这也是我能够长期保持学习热情以及后来深入 iOS 平台开发领域的敲门砖。

我最开始主要是拿别人的简单 DEMO 来学习，在别人写好的代码上改改、调调，我觉得这是入门最快的方法。在基本入门以后，我便开始系统地去看书、去验证，去实现自己的小想法，去尝试做一个小项目等。此外每当遇到问题，我都会去开发论坛交流，当时经常逛的网站有 Cocoa China、Stack Overflow 和 GitHub 等，在此过程中慢慢积累、慢慢武装自己。学习 iOS 的周期还是很漫长的，入门以后很长一段时间我并没有太大的提高，直到后来加入实际项目、参加公司实习以及能够研究一些优秀的开源代码之后，我才踏入进阶之路，同时也伴随着自己计算机专业水平的同步提高。事实上，我本科期间学习的汇编、编译原理、计算机组成原理、数据结构、数据库原理等，都一直在起着潜移默化的作用，而 iOS 技术的学习，则像是一个实践的平台，一个深入行业领域的路线。条条河流最终都是要汇入大海的，无论是一开始选择 iOS 开发、安卓开发，还是选择 Web 开发等路线，其最终目标都应该是借此打一口深深的井，钻下去然后慢慢扩散渗透，成为一名优秀的计算机行业专家，而不仅仅是一个初级平台开发者而已。

在编写整理本书期间，我从头到尾系统地梳理了自己的知识体系，不断地去验证、去挖掘重点、去剖析学习中最容易忽视的知识点，揭开我们学习中常常不愿意揭开的伤疤，然后认真地做出解析、敷上药膏。希望本书能够帮助更多的读者压缩这个学习过程的成本和周期，更快速地进入到更高的技术层面，更顺利地适应工作岗位，成为优秀的行业开发者。毕业后，我进入游戏行业，仍一直坚持 iOS 的使用和学习，这个长板对我尤其重要，是我平时想法和技术实践的主要移动平台。

本书技术部分将精选收录的题目进行了分类。第 1 章收录了 iOS 开发中的一些基础概念相关的问题；第 2 章和第 3 章分别归类了 Objective-C 语言从基础知识到中高级特性的问题，由浅入深地剖析了 Objective-C 语言开发各方面的核心问题；第 4 章收录的主要是有关官方

Cocoa Touch 框架的常见问题;第 5 章将 iOS 开发中的对象间通信机制相关的问题集中在一起,进行了总结和分析;第 6 章讨论了 iOS 中的一个重要话题:图层与动画,并结合问题进行了知识的总结和延伸;第 7 章总结了数据持久化有关的问题;第 8 章和第 9 章包含了 iOS 开发进阶之路的重中之重:内存管理和网络多线程编程;第 10 章收录了一些其他的重要的零碎话题,包括函数响应式编程、设计模式、第三方框架和程序调试问题等;第 11 章介绍了数据库相关知识;第 12 章介绍了操作系统相关知识。

对于书中的任何问题或困惑,读者都可以通过邮件联系我: yuancoder@foxmail.com。期待你的来信。

编 者
2018 年 5 月

目 录

前言

面试笔试经验技巧篇

经验技巧 1	如何巧妙地回答面试官的问题	2
经验技巧 2	如何回答技术性问题	3
经验技巧 3	如何回答非技术性问题	5
经验技巧 4	如何回答快速估算类问题	5
经验技巧 5	如何回答算法设计问题	6
经验技巧 6	如何回答系统设计题	9
经验技巧 7	如何解决求职中的时间冲突问题	11
经验技巧 8	如果面试问题曾经遇见过，是否要告知面试官	12
经验技巧 9	被企业拒绝后是否可以再申请	12
经验技巧 10	如何应对自己不会回答的问题	13
经验技巧 11	如何应对面试官的“激将法”语言	13
经验技巧 12	如何处理与面试官持不同观点这个问题	14
经验技巧 13	什么是职场暗语	15
经验技巧 14	名企 iOS 工程师行业访谈录	18
经验技巧 15	iOS 开发的前景如何	20
经验技巧 16	如何选择 iOS 开发语言	20
经验技巧 17	React Native 和 Weex 重要吗	21
经验技巧 18	企业对 iOS 开发者的要求有哪些	22
经验技巧 19	iOS 开发招聘有哪些要求	24
经验技巧 20	iOS 技术岗位面试精选	27

面试笔试技术攻克篇

第 1 章	iOS 开发基础概念	35
1.1	概念	35
1.1.1	什么是“应用瘦身”	35
1.1.2	什么是 Cocoa 和 Cocoa Touch	36
1.1.3	什么是谓词	36
1.1.4	什么是响应者链	38

1.1.5	什么是“懒加载”	39
1.1.6	类工厂方法是什么	40
1.1.7	App ID 和 Bundle ID 有什么不同	40
1.1.8	什么是糖衣语法	40
1.2	其他	43
1.2.1	什么是 SpriteKit 和 SceneKit	43
1.2.2	iOS 应用的生命周期回调方法主要有哪些	43
1.2.3	iOS 应用有哪几种不同状态? 分别表示什么含义	44
1.2.4	UIButton 到 NSObject 之间的继承关系是怎样的	45
1.2.5	Git 和 SVN 有什么异同	45
第 2 章	Objective-C 语言基础	47
2.1	Objective-C 语言基础特性	47
2.1.1	Objective-C 的优缺点有哪些	47
2.1.2	相对于 Objective-C 而言, Swift 有什么新特性	48
2.1.3	Foundation 对象与 Core Foundation 对象有什么区别	52
2.1.4	Objective-C 中的类方法和实例方法有什么本质区别和联系	53
2.1.5	子类初始化时为什么要调用 self = [super init]	54
2.1.6	#import 与#include 以及#import<>与#import" "各有什么区别	54
2.1.7	Objective-C 中@class 代表什么	54
2.1.8	Objective-C 中有二维数组吗? 如何实现	55
2.1.9	在 Objective-C 的数组或字典中, 添加 nil 对象会有什么问题	55
2.1.10	Objective-C 中的可变和不可变类型是什么	56
2.2	数据类型	57
2.2.1	在 Objective-C 中, 常量有哪几种类型	57
2.2.2	Objective-C 中数据类型的限定词有哪些	58
2.2.3	Objective-C 中的 NSInteger 类型和 C 语言中的 int 类型有什么区别	59
2.2.4	NSNumber 与 NSInteger 有什么区别	59
2.3	运算符和表达式	60
2.3.1	在 Objective-C 中, 前置运算和后置运算有什么区别	60
2.3.2	整型值和浮点值在赋值操作中如何相互转换	61
第 3 章	Objective-C 语言的高级特性	63
3.1	Objective-C 中的属性	63
3.2	Objective-C 语言的多态性	72
3.2.1	什么叫多态	72
3.2.2	重载、重写和隐藏的区别是什么	74
3.2.3	Objective-C 和 Swift 中有重载吗	74
3.3	Objective-C 语言的动态性	75
3.3.1	什么是编译时与运行时	79
3.3.2	所谓的 Objective-C 是动态运行时语言是什么意思	79

3.3.3	Objective-C 中的 id 类型指的是什么? id、nil 代表什么	82
3.3.4	一般的方法 method 和 Objective-C 中的选择器 selector 有什么区别	83
3.3.5	什么时候会报 unrecognized selector 错误	83
3.3.6	什么是目标-动作机制	83
3.3.7	在 runtime 中类与对象如何表示	84
3.3.8	如何打印一个类中所有的实例变量	85
3.3.9	如何使用 runtime 动态添加一个类	87
3.3.10	如何在 Category 中增加属性 (关联对象)	88
3.3.11	如何理解消息传递机制	89
3.3.12	如何理解消息转发机制	91
3.3.13	isKindOfClass 和 isMemberOfClass 有什么区别与联系	94
3.3.14	Objective-C 有私有方法吗? 有私有变量吗	94
3.4	Objective-C 中的类别与扩展机制	96
3.4.1	类别与其他特性 (类扩展和继承) 有什么区别	97
3.4.2	Objective-C 中类别特性的作用及其局限性是什么	97
3.4.3	类别和类扩展如何使用	98
3.4.4	为什么类别只能添加扩展方法而不能添加属性变量	101
3.5	Method Swizzling 魔法	102
3.5.1	Method Swizzling 的应用场景有哪些	102
3.5.2	如何使用 runtime 进行方法交换	105
3.6	其他问题	107
3.6.1	C 和 Objective-C 如何混用	107
3.6.2	Swift 和 Objective-C 如何互调	107
3.6.3	Objective-C 对象可以被 copy 的条件是什么	109
3.6.4	Objective-C 等同性中的字符串相等如何判断	110
3.6.5	一个 Objective-C 对象如何进行内存布局 (考虑有父类的情况)	111
第 4 章	Cocoa Touch 框架相关	112
4.1	UITableView	112
4.1.1	UITableViewCell 的复用原理是怎么样的	112
4.1.2	能否在一个视图控制器中嵌入两个 tableView 控制器	114
4.1.3	一个 tableView 是否可以关联两个不同的 datasource 数据源	115
4.1.4	如何对 UITableView 的滚动加载进行优化, 防止卡顿	116
4.2	UI 视图	117
4.2.1	viewDidLoad 和 viewWillAppear 的区别是什么	117
4.2.2	frame 和 bounds 有什么区别	117
4.2.3	masksToBounds 属性是什么? 它有什么作用	118
4.2.4	tintColor 的作用是什么	118
4.2.5	UIViewController 的生命周期方法有哪些	118
4.2.6	View 和 View 之间的传值方式有哪些	119

4.3	其他问题	119
4.3.1	xib 和 storyboard 相比各自的优缺点是什么	119
4.3.2	如何进行 iOS 6 和 iOS 7 的适配	120
4.3.3	imageNamed 和 imageWithContentsOfFile 有什么区别	120
4.3.4	UIDevice 如何获取设备信息	120
4.3.5	iOS 中是如何使用自定义字体的	122
第 5 章	iOS 开发中的对象间通信机制	125
5.1	iOS 中的 Protocol 和 Delegate	125
5.2	推送和通知	126
5.2.1	什么是消息推送? 和 Notification 有什么区别	126
5.2.2	什么是 Notification? 什么时候用 Delegate 或 Notification	128
5.2.3	NSNotification 是同步还是异步	129
5.3	Objective-C 中的键值编码和键值观察	130
5.3.1	什么是键值编码 KVC? 键路径是什么? 什么是键值观察 KVC	130
5.3.2	KVC 的应用场景有哪些	132
5.3.3	如何运用 KVO 进行键值观察	134
5.3.4	如何使用 KVO 设置键值观察依赖键	136
5.3.5	KVO 的背后原理是什么	137
5.3.6	setValueForKey:方法的底层实现是什么	138
5.3.7	NSMutableDictionary 中 setValue 和 setObject 有什么区别	139
5.3.8	NSNotification、Delegate、Block 和 KVO 的区别是什么	140
第 6 章	iOS 中的图层与动画	142
6.1	图层	142
6.1.1	UIView 和 CALayer 的区别与联系是什么	142
6.1.2	什么是 Layer 层对象	143
6.1.3	如何使用 CAShapeLayer 绘制图层	146
6.1.4	iOS 中如何实现为 UIImageView 添加圆角	148
6.1.5	contentsScale 属性有什么作用	149
6.1.6	如何理解 anchorPoint 和 position 的作用	150
6.1.7	如何理解 drawRect:方法	151
6.1.8	如何使用 mask 属性实现图层蒙版功能	152
6.1.9	如何解决 masksToBounds 离屏渲染带来的性能损耗	153
6.1.10	QuartzCore 和 Core Graphics 有什么区别	154
6.2	动画	154
6.2.1	UIView 动画原理是什么? 以 UIView 类的 animateWithDuration 方法为例	154
6.2.2	什么是隐式动画和显式动画	154
6.2.3	隐式动画的实现原理是什么? 如何禁用图层的隐式动画	155
6.2.4	CGAffineTransform 和 CATransform3D 分别有什么作用	157
6.2.5	CATransition 中过渡类型动画有哪几种 type	158

6.2.6	如何使用 UIView 动画自定义过渡动画	159
6.2.7	如何理解并使用 CAKeyframeAnimation	160
6.2.8	如何自定义 UIViewController 之间的转场动画	161
6.2.9	如何保持视图界面为动画结束时的状态	164
第 7 章	iOS 中的数据持久化	167
7.1	沙盒机制	169
7.2	数据持久化方案	170
7.2.1	iOS 平台做数据的持久化方式有哪些	170
7.2.2	如何使用 NSUserDefaults 偏好设置保存数据	170
7.2.3	如何使用 NSUserDefaults 保存自定义对象	171
7.2.4	什么是序列化或者归档	172
第 8 章	iOS 中的内存管理	174
8.1	内存管理	174
8.1.1	什么是内存泄漏？什么是安全释放	174
8.1.2	僵尸对象、野指针、空指针分别指什么？它们有什么区别	174
8.1.3	Objective-C 有 GC 垃圾回收机制吗	175
8.1.4	在 Objective-C 中，与 alloc 语义相反的方法是 dealloc 还是 release	175
8.2	内存管理机制	176
8.2.1	当使用 block 时，什么情况会发生引用循环？如何解决	176
8.2.2	CAAnimation 的 delegate 是强引用还是弱引用	176
8.2.3	按照默认法则，哪些关键字生成的对象需要手动释放	177
8.2.4	Objective-C 是如何实现内存管理的	179
8.2.5	如何实现 autoreleasepool	179
8.2.6	如果一个对象释放前被加到了 NotificationCenter 中，不在 NotificationCenter 中，那么 remove 对象可能会怎样	181
8.2.7	NSArray 和 NSMutableArray 在 Copy 和 MutableCopy 下的内存情况是怎样的	182
第 9 章	iOS 中的网络和多线程编程	183
9.1	iOS 网络编程与多线程基础	183
9.1.1	什么是线程？线程与进程有什么区别？为什么要使用多线程	189
9.1.2	如何理解多线程	190
9.1.3	如何理解 HTTP 协议	191
9.1.4	HTTPS 协议与 HTTP 协议有什么区别与联系	193
9.1.5	UIKit 类要在哪一个应用线程上使用	194
9.1.6	iOS 中有哪几种从其他线程回到主线程的方法	195
9.1.7	用户下载一个大图片，分成很多份下载，如何使用 GCD 实现	196
9.1.8	项目中什么时候选择使用 GCD？什么时候选择 NSOperation	196
9.1.9	NSOperation 如何实现线程依赖	196
9.1.10	什么是线程死锁	197
9.1.11	dispatch_barrier_(a)sync 的作用是什么	198

9.1.12	如何理解线程安全	199
9.1.13	如何实现 Cocoa 中多线程的安全	201
9.1.14	如何使用 NSURLConnection 进行网络请求	202
9.1.15	如何使用 NSURLSession 进行网络请求	205
9.2	block 与 GCD	206
9.2.1	block 有哪一种定义的方式	206
9.2.2	在 ARC 环境下, 是否需要使用 copy 关键字来修饰 block	207
9.2.3	在 block 内如何修改 block 外部变量	208
9.2.4	在 block 中使用 self 关键字是否一定导致循环引用	209
9.2.5	GCD 中有哪几种队列	211
9.2.6	如何理解 GCD 死锁	211
9.2.7	如何使用 GCD 实现线程之间的通信	212
9.2.8	GCD 如何实现线程同步	213
9.2.9	GCD 多线程编程中什么时候会创建新线程	215
9.2.10	iOS 中如何触发定时任务或延时任务	217
9.2.11	如何解决网络请求的依赖关系	220
第 10 章	iOS 其他话题	221
10.1	iOS 中函数响应式编程及 ReactiveCocoa 的使用	221
10.1.1	什么是 ReactiveCocoa? 如何使用	221
10.1.2	如何使用 RAC 防止 button 短时间内重复单击	223
10.2	iOS 基础设计模式	223
10.2.1	什么是单例模式	224
10.2.2	什么是 MVC 设计模式	225
10.2.3	如何理解 MVVM 设计模式	228
10.3	第三方框架	229
10.3.1	SDWebImage 是什么? 加载图片的原理是什么	230
10.3.2	什么是 CocoaPods	233
10.4	程序调试	234
10.4.1	BAD_ACCESS 在什么情况下出现	234
10.4.2	如何调试 BAD_ACCESS 错误	235
10.4.3	如何查看设备应用的 crash 日志	235
10.4.4	如何检测内存泄漏	237
10.4.5	lldb(gdb)常用的调试命令有哪些	237
第 11 章	数据库	239
11.1	数据库基础知识	239
11.2	SQL 语言的功能有哪些	240
11.3	内连接与外连接有什么区别	242
11.4	什么是事务	243
11.5	什么是存储过程? 它与函数有什么区别与联系	245

11.6	一三四范式有什么区别	245
11.7	什么是触发器	247
11.8	什么是游标	248
11.9	如果数据库日志满了,那么会出现什么情况	249
11.10	union 和 union all 有什么区别	249
11.11	什么是视图	250
11.12	什么是数据库三级封锁协议	251
11.13	索引的优缺点有哪些	251
第 12 章	操作系统	253
12.1	进程管理	253
12.1.1	进程与线程有什么区别	253
12.1.2	线程同步有哪些机制	254
12.1.3	内核线程和用户线程有什么区别	254
12.2	内存管理	255
12.2.1	内存管理有哪几种方式	255
12.2.2	什么是虚拟内存	256
12.2.3	什么是内存碎片?什么是内碎片?什么是外碎片	256
12.2.4	虚拟地址、逻辑地址、线性地址、物理地址有什么区别	257
12.2.5	Cache 替换算法有哪些	257
12.3	用户编程接口	259
12.3.1	库函数调用与系统调用有什么不同	259
12.3.2	静态链接与动态链接有什么区别	259
12.3.3	静态链接库与动态链接库有什么区别	260
12.3.4	用户态和核心态有什么区别	260
12.3.5	用户栈与内核栈有什么区别	261
附录		262
真题 1		262
真题 2		264
真题 3		265
真题 1 答案		267
真题 2 答案		274
真题 3 答案		281
参考文献		290

面试笔试经验技巧篇

想找到一份程序员的工作，一点技术都没有显然是不行的，但是，只有技术也是不够的。面试笔试经验技巧篇主要提供 iOS 程序员面试笔试经验、面试笔试问题方法讨论等。通过本篇的学习，求职者必将获取到丰富的应试技巧与方法。

经验技巧 1 如何巧妙地回答面试官的问题

所谓“来者不善，善者不来”，程序员面试中，求职者不可避免地需要回答面试官各种刁钻、犀利的问题，回答面试官的问题千万不能简单地回答“是”或者“不是”，而应该具体分析“是”或者“不是”的理由。

回答面试官的问题是一门很深的学问。那么，面对面试官提出的各类问题，如何才能条理清晰地回答呢？如何才能让自己的回答不至于撞上枪口呢？如何才能让自己的回答结果令面试官满意呢？

谈话是一种艺术，回答问题也是一种艺术，同样的话，不同的回答方式，往往也会产生出不同的效果，甚至是截然不同的效果。在此，编者提出以下几点建议，供读者参考。首先回答问题务必谦虚谨慎。既不能让面试官觉得自己很自卑，唯唯诺诺，也不能让面试官觉得自己清高自负，而应该通过问题的回答表现出自己自信从容、不卑不亢的一面。例如，当面试官提出“你在项目中起到了什么作用”的问题时，如果求职者回答：我完成了团队中最难的工作，此时就会给面试官一种居功自傲的感觉，而如果回答：我完成了文件系统的构建工作，这个工作被认为是整个项目中最具有挑战性的一部分内容，因为它几乎无法重用以前的框架，需要重新设计。这种回答不仅不傲慢，反而有理有据，更能打动面试官。

其次，回答面试官的问题时，不要什么都说，要适当地留有悬念。人一般都有猎奇的心理，面试官自然也不例外，而且，人们往往对好奇的事情更有兴趣、更加偏爱，也更加记忆深刻。所以，在回答面试官问题时，切记说关键点而非细节，说重点而非和盘托出，通过关键点，吸引面试官的注意力，等待他们继续“刨根问底”。例如，当面试官对你的简历中一个算法问题有兴趣，希望了解时，可以如下回答：我设计的这种查找算法，对于 80% 以上的情况，都可以将时间复杂度从 $O(n)$ 降低到 $O(\log n)$ ，如果您有兴趣，我可以详细给您分析具体的细节。

最后，回答问题要条理清晰、简单明了，最好使用“三段式”方式。所谓“三段式”，有点类似于中学作文中的写作风格，包括“场景/任务”“行动”和“结果”三部分内容。以面试官提的问题“你在团队建设中，遇到的最大挑战是什么”为例，第一步，分析场景/任务：在我参与的一个 ERP 项目中，我们团队一共四个人，除了我以外的其他三个人中，两个人能力很强，人也比较好相处，但有一个人不太好相处，每次我们小组讨论问题的时候，他都不太爱说话，也很少发言，分配给他的任务也很难完成。第二步，分析行动：为了提高团队的综合实力，我决定找个时间和他好好单独谈一谈。于是我利用周末时间，约他一起吃饭，吃饭的时候，顺便讨论了一下我们的项目，我询问了一些项目中他遇到的问题，通过他的回答，我发现他并不懒，也不糊涂，只是对项目不太了解，缺乏经验，缺乏自信而已，所以越来越孤立，越来越不愿意讨论问题。为了解决这个问题，我尝试着把问题细化到他可以完成的程度，从而建立起他的自信心。第三步，分析结果：他是小组中水平最弱的人，但是，慢慢地，他的技术变得越来越强了，也能够按时完成安排给他的工作了，人也越来越自信了，也越来越喜欢参与我们的讨论，并发表自己的看法，我们也都愿意与他一起合作了。“三段式”回答的一个最明显的好处就是条理清晰，既有描述，也有结果，有根有据，让面试官一

目了然。

回答问题的技巧，是一门大的学问。求职者完全可以在平时的生活中加以练习，提高自己与人沟通的技能，等到面试时，自然就得心应手了。

经验技巧 2 如何回答技术性问题

程序员面试中，面试官会经常询问一些技术性的问题，有的问题可能比较简单，都是历年的面试笔试真题，求职者在平时的复习中会经常遇到，应对自然不在话下。但有的题目可能比较难，来源于 Google、Microsoft 等大企业的题库或是企业自己为了招聘需要设计的题库，求职者可能从来没见过或者从来都不能完整地、独立地想到解决方案，而这些题目往往又是企业比较关注的。

如何能够回答好这些技术性问题呢？编者建议：会做的题目一定要拿满分，不会做的题目一定要拿部分分。即对于简单的题目，求职者要努力做到完全正确，毕竟这些题目，只要复习得当，完全回答正确一点问题都没有（编者的一个朋友把《编程之美》《编程珠玑》《程序员面试笔试宝典》上面的技术性题目与答案全都背得滚瓜烂熟，后来找工作无往不利）；对于难度比较大的题目，不要惊慌，也不要害怕，即使无法完全做出来，也要努力思考问题，哪怕是半成品也要写出来，至少要把自己的思路表达给面试官，让面试官知道你的想法，而不是完全回答不会或者放弃，因为面试官很多时候除了关注求职者的独立思考问题的能力以外，还会关注求职者技术能力的可塑性，观察求职者是否能够在别人的引导下去正确地解决问题，所以，对于你不会的问题，他们很有可能会循序渐进地启发你去思考，通过这个过程，让他们更加了解你。

一般而言，在回答技术性问题时，求职者大可不必胆战心惊，除非是没学过的新知识，否则，一般都可以采用以下六个步骤来分析解决。

（1）勇于提问

面试官提出的问题，有时候可能过于抽象，让求职者不知所措，或者无从下手，所以，对于面试中的疑惑，求职者要勇敢地提出来，多向面试官提问，把不明确或二义性的情况都问清楚。不用担心你的问题会让面试官烦恼，影响你的面试成绩，相反还对面试结果产生积极影响：一方面，提问可以让面试官知道你在思考，也可以给面试官一个心思缜密的好印象；另一方面，方便后续自己对问题的解答。

例如，面试官提出一个问题：设计一个高效的排序算法。求职者可能丈二和尚摸不到头脑，排序对象是链表还是数组？数据类型是整型、浮点型、字符型还是结构体类型？数据基本有序还是杂乱无序？数据量有多大，1000 以内还是百万以上个数？此时，求职者大可以将自己的疑问提出来，问题清楚了，解决方案也自然就出来了。

（2）高效设计

对于技术性问题，如何才能打动面试官？完成基本功能是必需的，仅此而已吗？显然不是，完成基本功能顶多只能算及格水平，要想达到优秀水平，至少还应该考虑更多的内容，以排序算法为例：时间是否高效？空间是否高效？数据量不大时也许没有问题，如果是海量数据呢？是否考虑了相关环节，例如数据的“增删改查”？是否考虑了代码的可扩展性、安全性、完整性以及鲁棒性？如果是网站设计，是否考虑了大规模数据访问的情况？是否需要

考虑分布式系统架构？是否考虑了开源框架的使用？

（3）伪代码先行

有时候实际代码会比较复杂，上手就写很有可能会漏洞百出、条理混乱，所以，求职者可以首先征求面试官的同意，在编写实际代码前，写一个伪代码或者画好流程图，这样做往往会让思路更加清晰明了。

切记在写伪代码前要告诉面试官，否则他们很有可能对你产生误解，认为你只会纸上谈兵，实际编码能力却不行。只有征得了他们的允许，方可先写伪代码。

（4）控制节奏

如果是算法设计题，面试官都会给求职者一个时间限制用以完成设计，一般为 20min 左右。完成得太慢，会给面试官留下能力不行的印象，但完成得太快，如果不能保证百分之百正确，也会给面试官留下毛手毛脚的印象，速度快当然是好事情，但只有速度，没有质量，速度快根本不会给面试加分。所以，编者建议，回答问题的节奏最好不要太慢，也不要太快，如果实在是完成得比较快，也不要急于提交给面试官，最好能够利用剩余的时间，认真仔细地检查一些边界情况、异常情况及极性情况等，看是否也能满足要求。

（5）规范编码

回答技术性问题时，多数都是纸上写代码，离开了编译器的帮助，求职者要想让面试官对自己的代码一看即懂，除了要字迹工整，最好能够严格遵循编码规范：函数变量命名、换行缩进、语句嵌套和代码布局等，同时，代码设计应该具有完整性，保证代码能够完成基本功能、输入边界值能够得到正确地输出、对各种不合规范的非法输入能够做出合理的错误处理，否则，写出的代码即使无比高效，面试官也不一定看得懂或者看起来非常费劲，这些对面试成功都是非常不利的。

（6）精心测试

在软件行业有一句真理：任何软件都有缺陷（bug）。但不能因为如此就纵容自己的代码，允许错误百出。尤其是在面试过程中，实现功能也许并不十分困难，困难的是在有限的时间内设计出的算法，各种异常是否都得到了有效的处理，各种边界值是否都在算法设计的范围内。

测试代码是让代码变得完备的高效方式之一，也是一名优秀程序员必备的素质之一。所以，在编写代码前，求职者最好能够了解一些基本的测试知识，做一些基本的单元测试、功能测试、边界测试以及异常测试。

在回答技术性问题时，注意在思考问题的时候，千万别一句话都不说，面试官面试的时间是有限的，他们希望在有限的时间内尽可能地去了解求职者，如果求职者坐在那里一句话不说，会让面试官觉得求职者不仅技术水平不行，思考问题能力以及沟通能力可能都存在问题。

其实，在面试时，求职者往往会存在一种思想误区，把技术性面试的结果看得太重要了。面试过程中的技术性问题，结果固然重要，但也并非最重要的内容，因为面试官看重的不仅仅是最终的结果，还包括求职者在解决问题的过程中体现出来的逻辑思维能力以及分析问题的能力。所以，求职者在与面试官的博弈中，要适当地提问，通过提问获取面试官的反馈信息，并抓住这些有用的信息进行辅助思考，从而博得面试官的欢心，进而提高面试的成功率。

经验技巧 3 如何回答非技术性问题

评价一个人的能力，除了专业能力，还有一些非专业能力，如智力、沟通能力和反应能力等，所以在 IT 企业招聘过程的笔试面试环节中，并非所有的笔试内容都是 C/C++、数据结构与算法及操作系统等专业知识，也包括其他一些非技术类的知识，如智力题、推理题和作文题等。技术水平测试可以考查一个求职者的专业素养，而非技术类测试则更加强调求职者的综合素质，包括数学分析能力、反应能力、临场应变能力、思维灵活性、文字表达能力和性格特征等内容。考查的形式多种多样，但与公务员考查相似，主要包括行测（占大多数）、性格测试（大部分都有）、应用文和开放问题等内容。

每个人都有自己的答题技巧，答题方式也各不相同，以下是一些相对比较好的答题技巧（以行测为例）：

1) 合理有效的时间管理。由于题目的难易不同，所以不要对所有题目都“绝对的公平”、都“一刀切”，要有轻重缓急，最好的做法是不按顺序回答。行测中有各种题型，如数量关系、图形推理、应用题、资料分析和文字逻辑等，而不同的人擅长的题型是不一样的，因此应该首先回答自己最擅长的问题。例如，如果对数字比较敏感，那么就先答数量关系。

2) 注意时间的把握。由于题量一般都比较大会，可以先按照总时间/题数来计算每道题的平均答题时间，如 10s，如果看到某一道题 5s 后还没思路，则马上放弃。在做行测题目的时候，以在最短的时间内拿到最多分为目标。

3) 平时多关注图表类题目，培养迅速抓住图表中各个数字要素间相互逻辑关系的能力。

4) 做题要集中精力，只有集中精力、全神贯注，才能将自己的水平最大限度地发挥出来。

5) 学会关键字查找，通过关键字查找，能够提高做题效率。

6) 提高估算能力，有很多时候，估算能够极大地提高做题速度，同时保证正确率。

除了行测以外，一些企业非常相信个人性格对入职匹配的影响，所以都会引入相关的性格测试题用于测试求职者的性格特性，看其是否适合所投递的职位。大多数情况下，只要按照自己的真实想法选择就行了，不要弄巧成拙，因为测试是为了得出正确的结果，所以大多测试题前后都有相互验证的题目。如果求职者自作聪明，选择该职位可能要求的性格选项，则很可能导致测试前后不符，这样很容易让企业发现你是个不诚实的人，从而首先予以筛选。

经验技巧 4 如何回答快速估算类问题

有些大企业的面试官，总喜欢出一些快速估算类问题。对他们而言，这些问题只是手段，不是目的，能够得到一个满意的结果固然是他们所需要的，但更重要的是通过这些题目他们可以考查求职者的快速反应能力以及逻辑思维能力。由于求职者平时准备的时候可能对此类问题有所遗漏，一时很难想起解决的方案。而且，这些题目乍一看确实是毫无头绪，无从下手，其实求职者只要从惊慌失措中冷静下来，稍加分析，也就那么回事。因为此类题目比较灵活，属于开放性试题，一般没有标准答案，只要弄清楚了回答要点，分析合理到位，具有