**1、Java 基础**

* Java Object类方法
* HashMap原理，Hash冲突，并发集合，线程安全集合及实现原理
* HashMap 和 HashTable 区别
* HashCode 作用，如何重载hashCode方法
* ArrayList与LinkList区别与联系
* GC机制
* Java反射机制，Java代理模式
* Java泛型
* Synchronized原理
* Volatile实现原理
* 方法锁、对象锁、类锁的意义和区别
* 线程同步的方法：Synchronized、lock、reentrantLock分析
* Java锁的种类: 公平锁、乐观锁、互斥锁、分段锁、偏向锁、自旋锁等
* ThreadLocal的原理和用法
* ThreadPool的用法和示例
* wait()和sleep()的区别

**2、Java 高阶**

* Java虚拟机，Java运行，Java GC机制（可达性分析法，引用计数法）
* Java对象的完整生命周期
* JVM内存模型
* 进程间通信，线程间通信
* JVM类加载机制
* Java引用类型
* 设计模式：除常用设计模式之外，特别的，反射机制，代理模式
* HTTP协议和HTTPS协议
* Socket协议，Socket实现长连接
* TCP和UDP协议
* HTTP协议中GET和POST的具体实现
* 序列化和反序列化
* 线程池的实现原理
* 数据库基础知识：多表查询、索引、数据库事务

**3、数据结构、算法**

**数据结构**

* 栈和队列
* 数组和链表，自定义一个动态数组
* Hash表，及Hash冲突的解决
* 二叉树
* B+ B-树
* 基础排序算法：重点  快排、归并排序、堆排序（大根堆、小根堆）
* 快排的优化
* 二分查找与变种二分查找
* 哈夫曼树、红黑树
* 字符串操作，字符串查找，KMP算法
* 图的BFS、DFS、prim、Dijkstra算法（高阶技能）
* 经典问题：海量数据的处理  （10亿个数中找出最大的10000个数 TOP K问题）

**算法**

* 分治算法
* 动态规划
* 贪心算法
* 分支限界法

**4、Android 基础**

* Application生命周期
* Android Activity生命周期
* Android Service、IntentService，Service和组件间通信
* Activity的onNewIntent
* Fragment的懒加载实现，参数传递与保存
* ContentProvider实例详解
* BroadcastReceiver使用总结
* Android消息机制
* Binder机制，共享内存实现原理
* Android 事件分发机制
* Android 多线程的实现：Thread、HandlerThread、AsyncTask、IntentService、RxJava
* ActivityThread工作原理
* 嵌套滑动实现原理
* RecyclerView与ListView(缓存原理，区别联系，优缺点)
* View的绘制原理，自定义View，自定义ViewGroup
* View、SurfaceView 与 TextureView
* 主线程Looper.loop为什么不会造成死循环
* ViewPager的缓存实现
* requestLayout，invalidate，postInvalidate区别与联系
* AndroidP新特性
* Android两种虚拟机
* ADB常用命令
* Asset目录与res目录的区别
* Android SQLite的使用入门

**5、Android 开发高级**

引子：Android高级工程师招聘要求：1. 熟悉Android SDK，熟悉Android UI，熟悉Android各种调试工具；2. 有丰富的Android应用架构能力，能够独立主导并架构App；3. Mobile Web 开发经验；具备各种复合技能：熟悉iOS、H5、Python、.NET等多种开发语言的优先考虑；4. 对Android性能优化，安全，软件加固，自动化测试有深刻认识;5. 博客，开源项目

**Android技术难点**

AIDL、Binder、多进程、View的绘制流程、事件分发、消息队列等。

这类知识对于定位自己为高级Android工程师的人来说是必须掌握的，同时他也是能鉴别高级和初中级工程师的一块试金石，其中binder是Android系统进程间通信最重要的手段之一，现阶段app的发展离不开多进程的运用，经常会启动例如定位、推送等需要在后台开启动的进程来来保证主进程的内存运行；所以合理的使用多进程也是十分必要的；

view的绘制是我们自定义控件的理论基础，只有掌握了view是如何绘制的才能个性化的自定义控件；

事件分发一直是Android开发的难点之一，也是必须掌握的；

关于handler机制也是android的一块难点，因为包括Asynctask、系统启动、Intentservice等底层都是通过handler来实现的，所以掌握后handler机制不仅能提高你的实战开发能力，更能让你系统的了解整个android系统运作的情况。

**Android框架层源码掌握**

Android框架层有很多东西，以下几个是高级程序员必须要掌握的：

1、Android包管理机制，核心PackageManagerService

2、Window管理，核心WindowManagerService

3、Android Activity启动和管理，核心ActivityManagerService

4、根Activity工作流程

5、Context关联类

**各种原理，经典第三方库源码系列**

* 自定义LayoutManager，RecyclerView中如何自定义LayoutManager
* VLayout实现原理，即如何自定义LayoutManager
* Glide加载原理，缓存方案，LRU算法
* Retrofit的实现与原理
* OKHttp3的使用，网络请求中的Intercept
* EventBus实现原理
* ButterKnife实现原理
* RxJava实现原理
* Dagger依赖注入
* 热修复实现原理，解决方案
* 组件化原理和解决方案

**Android进程通信以及多进程开发**

Android 多进程和Application关系

经典解决方案：

* 多进程通信解决方案：Andromeda

*https://mp.weixin.qq.com/s/PZs1wss3PizqSE8U2RGXYw*

**Android动画机制 & Android绘图原理**

经典学习资料：

* HenCoder: 给高级Android工程师的进阶手册

*https://hencoder.com/*

**Android页面恢复**

Android的页面恢复采用以下两个方法：

* onSaveInstanceState(Bundle outState)
* onRestoreInstanceState(Bundle savedInstanceState)

onSaveInstanceState: 当Activity容易被系统销毁时，会触发该方法。

具体的说

1、用户点击Home键

2、用户点击Home键，切换到其他应用程序

3、有电话来了等附加操作

**混合开发及Android WebView应用**

混合开发涉及到的知识点主要包括：

1、APP调用WebView加载url

2、掌握WebView的封装，了解所有的WebSettings配置，掌握WebViewClient、WebChromeClient

3、掌握WebView和Native双向通信机制，会自己封装双向通信中间件

对WebView的封装可参考：

*https://github.com/Justson/AgentWeb*

对通信中间件原理理解：

*https://github.com/xudjx/webprogress*

**Gradle，自动化构建，持续集成相关**

**6、Android 系统**

**Android Studio编译过程**

**其中使用到的编译工具：**

aapt、aidl、Java Compiler、dex、 zipalign

**主要步骤描述：**

1、通过aapt打包res资源文件，生成R.java、resources.arsc和res文件（二进制 & 非二进制如res/raw和pic保持原样）

2、处理.aidl文件，生成对应的Java接口文件

3、通过Java Compiler编译R.java、Java接口文件、Java源文件，生成.class文件

4、通过dex命令，将.class文件和第三方库中的.class文件处理生成classes.dex

5、通过apkbuilder工具，将aapt生成的resources.arsc和res文件、assets文件和classes.dex一起打包生成apk

6、通过Jarsigner工具，对上面的apk进行debug或release签名

7、通过zipalign工具，将签名后的apk进行对齐处理。

**App启动加载过程**

**Android虚拟机 Android App运行的沙箱原则**

**7、Android 架构**

在Android源码中最重要的三个类：ActivityManagerService／PackageManagerService／View，推荐大家周末的时候可以去阅读下这部分的源码，阅读源码能提高我们今后设计架构自己代码的能力，同时也能从底层了解整个android系统的运行原理，其他一些比如主线程的消息循环、主线程如何和AMS如何跨进程交互、SystemServer进程中的各种Service的工作方式、AsyncTask的工作原理等。这些知识也是作为一个Android高级开发工程师必须掌握的，不能整天沉溺于ui和四大组件的交互，要站在更高的角度去考虑Android的有些问题。

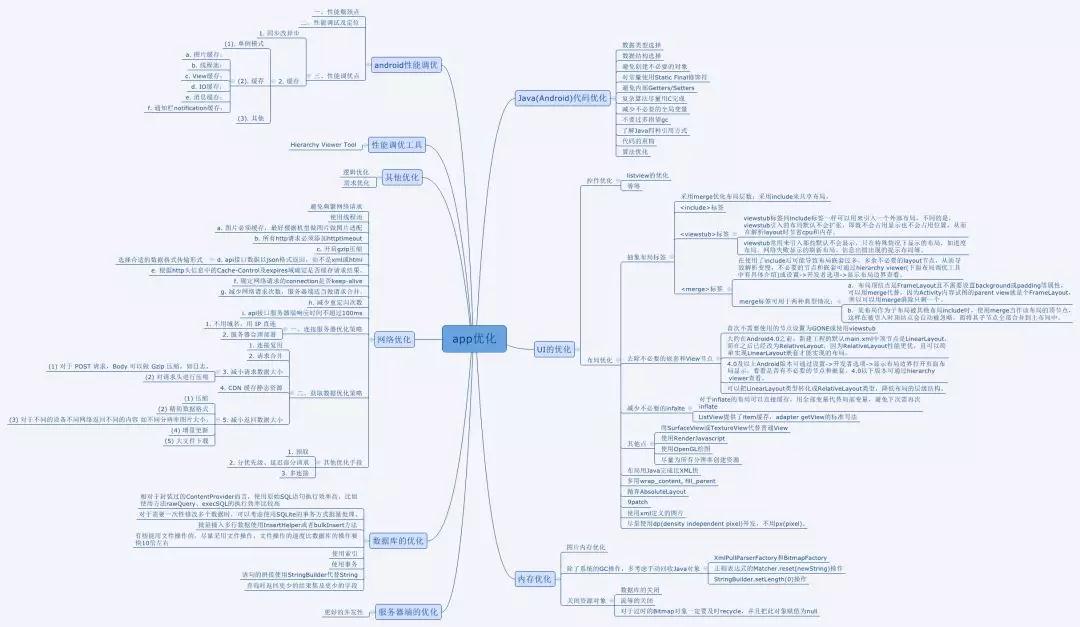
**参考资料：**

* 我对移动端架构的思考

*https://mp.weixin.qq.com/s/OEzcsPZHCVkjeUCh6hMuWg*

* MVC模式
* MVP模式
* MVVM模式
* CLEAN模式
* 组件化开发
* 跨平台开发：Flutter、ReactNative（RN未来要黄，了解一下就好）

**8、Android 优化**



**9、移动开发外围**

**服务器开发相关**

* SpringBoot技术
* Restful API开发
* 网络协议理解：TCP/IP、HTTP/HTTPS、OSI七层协议
* 授权认证协议：  OAuth2.0 等
* 基本的数据库技术
* 数据缓存技术：Memcached、Redis，Web缓存原理
* 消息队列技术
* 监控、日志分析技术

**前端开发相关**

前端开发知识很多，框架层出不穷，本质的东西却只有以下这些。

* 核心必备：HTML、CSS、JavaScript
* 入门提高：浏览器兼容性、自定义UI和动效
* 中级技能：框架层出不穷，当前以vue.js、react.js 为核心
* 协作开发技能：包管理、模块化，工具采用 npm、webpack等
* 高级技能：框架原理源码研究

**开发调试各种工具**

* 性能分析工具：Memory Monitor
* 性能追踪及方法执行分析： TraceView
* 视图分析：Hierarchy Viewer
* ApkTool- 用于反向工程Android Apk文件的工具
* Lint- Android lint工具是一个静态代码分析工具
* Dex2Jar- 使用android .dex和java .class文件的工具