-编程，始于传智

iOS面试宝典

Version 1.2

[By \*](wangzhenyang@itcast.cn)\*

iOS面试宝典-V1.2

# 内容介绍

该面试宝典不仅收录了上海iOS小组成员亲身面试遇到的问题，还收录了从传智.黑马学子那里收集过来的问题。在以后的工作中该面试宝典也会不断的更新和扩充 , 当然也希望大家能够多多奉献比较优质的面试题。

注意：该面试宝典仅供参考，无法对其所有内容的正确性和准确性做保证。由于知识水平有限加之编写时间仓促因此难免有不少bug的存在，希望大家见谅。

如果您有不错的知识或者面试题，您可以发送面试题到xx@itcast.cn，我们将不胜感激。让天下没有难学的知识，希望你我的努力能帮到更多的莘莘学子。

# C语言（★★）

## C语言语法

### 局部变量能否和全局变量重名？

　答：能，局部会屏蔽全局。要用全局变量，需要使用"::" ;局部变量可以与全局变量同名，在函数内引用这个变量时，会用到同名的局部变量，而不会用到全局变量。对于有些编译器而言，在同一个函数内可以定义多个同 名的局部变量，比如在两个循环体内都定义一个同名的局部变量，而那个局部变量的作用域就在那个循环体内。

### 如何引用一个已经定义过的全局变量？

答：extern 　可以用引用头文件的方式，也可以用extern关键字，如果用引用头文件方式来引用某个在头文件中声明的全局变理，假定你将那个编写错了，那么在编译期 间会报错，如果你用extern方式引用时，假定你犯了同样的错误，那么在编译期间不会报错，而在连接期间报错。

### 全局变量可不可以定义在可被多个.C文件包含的头文件中？为什么？

答：可以，在不同的C文件中以static形式来声明同名全局变量。 　　可以在不同的C文件中声明同名的全局变量，前提是其中只能有一个C文件中对此变量赋初值，此时连接不会出错.

### 关键字volatile有什么含意？并举出三个不同的例子？

一个定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。下面是volatile变量的几个例子：

1) 并行设备的硬件寄存器（如：状态寄存器）

2) 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)

3) 多线程应用中被几个任务共享的变量

### static 关键字的作用是什么？

（1）函数体内 static 变量的作用范围为该函数体，不同于 auto 变量，该变量的内存只被分配一次，

因此其值在下次调用时仍维持上次的值；

（2）在模块内的 static 全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；

（3）在模块内的 static 函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明

它的模块内；

（4）在类中的 static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；

（5）在类中的 static 成员函数属于整个类所拥有，这个函数不接收 this 指针，因而只能访问类的static 成员变量。

### 堆和栈的区别是什么？

管理方式：对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。

申请大小： 栈：在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统 预先规定好的，在WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示 overflow。因此，能从栈获得的空间较小。 堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地 址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

碎片问题：对于堆来讲，频繁的new/delete势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个 问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出

分配方式：堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由 alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。

分配效率：栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的 效率比较高。堆则是C/C++函数库提供的，它的机制是很复杂的

### static全局变量与普通的全局变量有什么区别？static局部变量和普通局部变量有什么区别？static函数与普通函数有什么区别？

答: 1) 全局变量(外部变量)的说明之前再冠以static 就构成了静态的全局变量。全局变量本身就是静态存储方式， 静态全局变量当然也是静态存储方式。 这两者在存储方式上并无不同。这两者的区别在于非静态全局变量的作用域是整个源程序， 当一个源程序由多个源文件组成时，非静态的全局变量在各个源文件中都是有效的。 而静态全局变量则限制了其作用域， 即只在定义该变量的源文件内有效， 在同一源程序的其它源文件中不能使用它。由于静态全局变量的作用域局限于一个源文件内，只能为该源文件内的函数公用， 因此可以避免在其它源文件中引起错误。

2) 从以上分析可以看出， 把局部变量改变为静态变量后是改变了它的存储方式即改变了它的生存期。把全局变量改变为静态变量后是改变了它的作用域，限制了它的使用范围。

 3) static函数与普通函数作用域不同,仅在本文件。只在当前源文件中使用的函数应该说明为内部函数(static)，内部函数应该在当前源文件中说明和定义。对于可在当前源文件以外使用的函数，应该在一个头文件中说明，要使用这些函数的源文件要包含这个头文件

综上所述:

static全局变量与普通的全局变量有什么区别：

static全局变量只初使化一次，防止在其他文件单元中被引用;

static局部变量和普通局部变量有什么区别：

static局部变量只被初始化一次，下一次依据上一次结果值；

static函数与普通函数有什么区别：

static函数在内存中只有一份，普通函数在每个被调用中维持一份拷贝

### 关键字const的作用分别是什么？

const int a;

int const a;

const int \*a;

int const \*a;

int \* const a;

int const \* const a;

1> 前两个的作用是一样：a 是一个常整型数

2> 第三、四个意味着 a 是一个指向常整型数的指针(整型数是不可修改的，但指针可以)

3> 第五个的意思：a 是一个指向整型数的常指针(指针指向的整型数是可以修改的，但指针是不可修改的)

4> 最后一个意味着：a 是一个指向常整型数的常指针(指针指向的整型数是不可修改的，同时指针也是不可修改的)

### struct与union的区别是什么？

设有以下说明和定义：

　 typedef union

{

long i;

int k[5];

 char c;

} DATE;

　struct data

 {

   int cat;

   DATE cow;

double dog;

} too;

　DATE max;

则语句 printf("%d",sizeof(struct data)+sizeof(max));的执行结果是：\_\_\_52\_\_\_\_

考点:区别struct与union.(一般假定在32位机器上)

答：DATE是一个union, 变量公用空间. 里面最大的变量类型是int[5], 占用20个字节. 所以它的大小是20. data是一个struct, 每个变量分开占用空间. 依次为int4 + DATE20 + double8 = 32. 所以结果是 20 + 32 = 52. 当然...在某些16位编辑器下, int可能是2字节,那么结果是 int2 + DATE10 + double8 = 20

### 关于指针的理解

1.请写出下题输出结果：

1. Main(){
2. Int a[5] = {1,2,3,4,5};
3. Int \*ptr = (int \*)(&a+1);
4. Printf(“%d,%d”,\*(a+1),\*(ptr – 1));
5. }

答：2,5

    \*(a+1)就是a[1]，\*(ptr-1)就是a[4],执行结果是2.5 ，&a+1不是首地址+1，系统会认为加一个a数组的偏 移，是偏移了一个数组的大小（本例是5个int，int \*ptr=(int \*)(&a+1); 则ptr实际 是&(a[5]),也就是a+5 原因如下：

　&a是数组指针，其类型为 int (\*)[5]; 而指针加1要根据指针类型加上一定的值，不同类型的指针+1之后增加的大小不同。a是长度为5的int数组指针，所以要加 5\*sizeof(int)所以ptr实际是a[5]，但是prt与(&a+1)类型是不一样的(这点很重要)，所以prt-1只会减去sizeof(int\*)，a,&a的地址是一样的，但意思不一样，a是数组首地址，也就是a[0]的地址，&a是对象（数组）首地址，a+1是数组下一元素的地址，即a[1],&a+1是下一个对象的地址，即a[5].

2.写出下列代码的输出内容

#include <stdio.h>

int inc(int a)

 { return(++a); }

 int multi(int\*a,int\*b,int\*c)

 { 　 return(\*c=\*a\*\*b); }

typedef int(FUNC1)(int in);

 typedef int(FUNC2) (int\*,int\*,int\*);

　 void show(FUNC2 fun,int arg1, int\*arg2)

　 {

  FUNC1 p=&inc;

　 int temp =p(arg1);

　 fun(&temp,&arg1, arg2);

　 printf("%dn",\*arg2);

 }

 main()

 {

　 int a; 　　       //局部变量a为0;

　 show(multi,10,&a);

 return 0;

　　 }

　答：110

3.以下为Windows NT下的32位C++程序，请计算sizeof的值  void Func ( char str[100] )   {     sizeof( str ) = ?   }   void \*p = malloc( 100 );  sizeof ( p ) = ?

答案：这题 很常见了,Func ( char str[100] )函数中数组名作为函数形参时，在函数体内，数组名失去了本身的内涵，仅仅只是一个指针；在失去其内涵的同时，它还失去了其常量特性，可以作自增、自减等 操作，可以被修改。Windows NT 32位平台下，指针的长度（占用内存的大小）为4字节，故sizeof( str ) 、sizeof ( p ) 都为4。

### c和obj-c如何混用?

1)obj-c的编译器处理后缀为m的文件时,可以识别obj-c和c的代码,处理mm文件可以识别obj-c,c,c++代码,但cpp文件必须只能用c/c++代码,而且cpp文件include的头文件中,也不能出现obj- c的代码,因为cpp只是cpp。 2) 在mm文件中混用cpp直接使用即可,所以obj-c混cpp不是问题 3)在cpp中混用obj- c其实就是使用obj-c编写的模块是我们想要的。如果模块以类实现,那么要按照cpp class的标准写类的定 义,头文件中不能出现obj-c的东西,包括#import cocoa的。 实现文件中,即类的实现代码中可以使用obj-c的东西,可以import,只是后缀是mm。如果模块以函数实现,那么头文件要按 c的格 式声明函数,实现文件中,c++函数内部可以用obj-c,但后缀还是mm或m。总结:只要cpp文件和cpp include的文件中不包含 obj-c的东西就可以用了,cpp混用obj-c的关键是使用接口,而不能直接使用实现代码,实际上cpp混用的是obj-c编译后的o文 件,这个东西其实是无差别的,所以可以用。obj-c的编译器支持cpp。

# OC语言（★★）

有人可能会问对于我们学iOS的同学来讲，面试还会问Objective-C基础吗？答案是会的，但是不会太多，因此我给了两颗星的重要程度。一般笔试的时候出现Objective-C基础题的概率比较大，口头面试的时候比较少，比如自己在面试的时候一些对基础知识比较看重的面试官会深究着Objective-C基础去问，比如Objective-C内存管理等等。

## Objective-C语法

### 1. 面向对象都有哪些特性以及你对这些特性的理解

 继承：继承是从已有类得到继承信息创建新类的过程。提供继承信息的类被称为父类（超类、基类）；得到继承信息的类被称为子类（派生类）。继承让变化中的软件系统有了一定的延续性，同时继承也是封装程序中可变因素的重要手段。

 封装：通常认为封装是把数据和操作数据的方法绑定起来，对数据的访问只能通过已定义的接口。面向对象的本质就是将现实世界描绘成一系列完全自治、封闭的对象。我们在类中编写的方法就是对实现细节的一种封装；我们编写一个类就是对数据和数据操作的封装。可以说，封装就是隐藏一切可隐藏的东西，只向外界提供最简单的编程接口。

 多态性：多态性是指允许不同子类型的对象对同一消息作出不同的响应。简单的说就是用同样的对象引用调用同样的方法但是做了不同的事情。多态性分为编译时的多态性和运行时的多态性。如果将对象的方法视为对象向外界提供的服务，那么运行时的多态性可以解释为：当A系统访问B系统提供的服务时，B系统有多种提供服务的方式，但一切对A系统来说都是透明的。方法重载（overload）实现的是编译时的多态性（也称为前绑定），而方法重写（override）实现的是运行时的多态性（也称为后绑定）。运行时的多态是面向对象最精髓的东西，要实现多态需要做两件事：1. 方法重写（子类继承父类并重写父类中已有的或抽象的方法）；2. 对象造型（用父类型引用引用子类型对象，这样同样的引用调用同样的方法就会根据子类对象的不同而表现出不同的行为）。

 抽象：抽象是将一类对象的共同特征总结出来构造类的过程，包括数据抽象和行为抽象两方面。抽象只关注对象有哪些属性和行为，并不关注这些行为的细节是什么。

### 2. 我们说的Objective-C是动态运行时语言是什么意思? (When we call objective c is runtime language what does it mean?)

主要是将数据类型的确定由编译时,推迟到了运行时。这个问题其实浅涉及到两个概念,运行时和多态。简单来说, 运行时机制使我们直到运行时才去决定一个对象的类别,以及调用该类别对象指定方法。多态:不同对象以自己的方式响应 相同的消息的能力叫做多态。意思就是假设生物类(life)都用有一个相同的方法-eat;那人类属于生物,猪也属于生物,都 继承了life后,实现各自的eat,但是调用是我们只需调用各自的eat方法。也就是不同的对象以自己的方式响应了相同的消 息(响应了eat这个选择器)。因此也可以说,运行时机制是多态的基础.

### 3. readwrite, readonly, assign, retain, copy, nonatomic, strong, weak 属性的作用?并区别strong(强引用)、weak(弱引用)? 什么情况使用copy,assign和retain?

readwrite 是可读可写特性;需要生成getter方法和setter方法时 ;

readonly 是只读特性 只会生成getter方法 不会生成setter方法 ,不希望属性在类外改变;

assign 是赋值特性,setter方法将传入参数赋值给实例变量;仅设置变量时;, assign用于简单数据类型,如NSInteger,double,bool;

retain 表示持有特性,setter方法将传入参数先保留,再赋值,传入参数的retaincount会+1;

copy 表示赋值特性,setter方法将传入对象复制一份;需要完全一份新的变量时;

nonatomic 非原子操作,决定编译器生成的setter getter是否是原子操作;

atomic表示多线程安全,一般使用 nonatomic。

retain 和copy用户对象,copy用于当 a指向一个对象, b也想指向同样的对象的时候, 如果用assign, a如果释放,再 调用b会crash,如果用copy 的方式,a和b各自有自己的内存,就可以解决这个问题。retain 会使计数器加1,也可以解 决assign的问题。另外:atomic和nonatomic用来决定编译器生成的getter和setter是否为原子操作。 在多线程环境下,原子操作是必要的,否则有可能引起错误的结果。

其实从这个也可以引申到内存管理的一系列问题.面试官会让你讲讲MRC和ARC的理解,随即也引申出Block的内存机制,为啥推荐Block用copy,Block会存在哪些内存问题?循环引用?

### 4. 分别描述内存管理要点、autorelease、release、NSAutoreleasePool?并说明autorelease是什 么时候被release的?简述什么时候由你负责释放对象,什么时候不由你释放?[NSAutoreleasePool release]和[NSAutoreleasePool drain]有什么区别?

内存管理要点:

Objective-C 使用引用计数机制(retainCount)来管理内存。内存每被引用一次,该内存的引用计数+1,每被释放一次引 用计数-1。当引用计数 = 0 的时候,调用该对象的 dealloc 方法,来彻底从内存中删除该对象。 alloc,allocWithZone,new(带初始化)时:该对象引用计数 +1;

retain:手动为该对象引用计数 +1;

copy:对象引用计数 +1;

mutableCopy:生成一个新对象,新对象引用计数为 1;

 release:手动为该对象引用计数 -1;

autorelease:把该对象放入自动释放池,当自动释放池释放时,其内的对象引用计数 -1。

NSAutoreleasePool: NSAutoreleasePool是通过接收对象向它发送的autorelease消息,记录该对象的release消息,当自动释放池被销毁时,会自动向池中的对象发送release消息。 autorelease 是在自动释放池被销毁,向池中的对象发送release 只能释放自己拥有的对象, 区别是:在引用计数环境下(在不使用ARC情况下),两者基本一样,在GC(垃圾回收制)环境下,release 是一个no-op(无效操作),所以无论是不是GC都使用drain

面试中内存管理,release和autorelease的含义?这里尤其要强调下autorelease,它引申出自动释放池,也能引申出Run loop!

### 自动释放池是什么,如何工作 ?

当 您向一个对象发送一个autorelease消息时，Cocoa就会将该对象的一个引用放入到最新的自动释放池。它仍然是个对象，因此自动释放池定义的作用域内的其它对象可以向它发送消息。当程序执行到作用域结束的位置时，自动释放池就会被释放，池中的所有对象也就被释放。

1>. Objective-C 是 通过一种"referring counting"(引用计数)的方式来管理内存的, 对象在开始分配内存(alloc)的时候引用计数为1,以后每当碰到有copy,retain的时候引用计数都会加1, 每当碰到release和autorelease的时候引用计数就会减1,如果此对象的计数变为了0, 就会被系统销毁.

2>. NSAutoreleasePool 就是用来做引用计数的管理工作的,这个东西一般不用自己管理.

3>. autorelease和release没什么区别,只是引用计数减1的时机不同而已,autorelease会在对象的使用真正结束的时候才做引用计数减1.

### IPhone OS有没有垃圾回收?autorelease 和垃圾回收制(gc)有什么关系?

IPhone OS 中没有垃圾回收 。autorelease只是延迟释放,gc是每隔一段时间询问程序,看是否有无指针指向的对象,若有,就将它回收。他们 两者没有什么关系。

### 简述NotificationCenter、KVC、KVO、Delegate?并说明它们之间的区别?

KVO（Key-Value- Observing）：是键值监听,键值观察机制, 当观察者为一个对象的属性进行了注册，被观察对象的isa指针被修改的时候，isa指针就会指向一个中间类，而不是真实的类。所以 isa指针其实不需要指向实例对象真实的类。所以我们的程序最好不要依赖于isa指针。在调用类的方法的时候，最好要明确对象实例的类名

KVC(Key-Value-Coding)内部的实现：是键值编码, 一个对象在调用setValue的时候，（1）首先根据方法名找到运行方法的时候所需要的环 境参数。（2）他会从自己isa指针结合环境参数，找到具体的方法实现的接口。（3）再直接查找得来的具体的方法实现。

Delegate: 代理的目的是改变或传递控制链。允许一个类在某些特定时刻通知到其他类，而不需要获取到那些类的指针。可以减少框架复杂度。消息的发送者(sender)告知接收者(receiver)某个事件将要发生，delegate同意然然后发送者响应事件，delegate机制使得接收者可以改变发送者的行为。通常发送者和接收者的关系是直接的一对多的关系。

Notification: 消息的发送者告知接收者事件已经发生或者将要发送，仅此而已，接收者并不能反过来影响发送者的行为。通常发送者和接收者的关系是间接的多对多关系。

1). 效率肯定是delegate比nsnotification高。

2). delegate方法比notification更加直接，最典型的特征是，delegate方法往往需要关注返回值，也就是delegate方法的结果。比如-windowShouldClose:，需要关心返回的是yes还是no。所以delegate方法往往包含should这个很传神的词。也就是好比你做我的delegate，我会问你我想关闭窗口你愿意吗？你需要给我一个答案，我根据你的答案来决定如何做下一步。相反的，notification最大的特色就是不关心接受者的态度，我只管把通告放出来，你接受不接受就是你的事情，同时我也不关心结果。所以notification往往用did这个词汇，比如NSWindowDidResizeNotification，那么nswindow对象放出这个notification后就什么都不管了也不会等待接受者的反应。

1）两个模块之间联系不是很紧密，就用notification传值，例如多线程之间传值用notificaiton。

2）delegate只是一种较为简单的回调，且主要用在一个模块中，例如底层功能完成了，需要把一些值传到上层去，就事先把上层的函数通过delegate传到底层，然后在底层call这个delegate，它们都在一个模块中，完成一个功能，例如说 NavgationController 从 B 界面到A 点返回按钮 (调用popViewController方法) 可以用delegate比较好。

### What is lazy loading ?

  就是懒汉模式,只在用到的时候才去初始化。也可以理解成延时加载。我觉得最好也最简单的一个列子就是tableView中图片的加载显示了。 一个延时载,避免内存过高,一个异步加载,避免线程堵塞。

### OC有多继承吗?没有的话可以用什么方法替代?

多继承即一个子类可以有多个父类,它继承了多个父类的特性。Object-c的类没有多继承,只支持单继承,如果要实现多继承的话,可以通过类别和协议的方式来实现,OC类似于多继承,是在 用protocol委托代理来实现的;可以实现多个接口,通过实现多个接口可以完成C++的多重继承;Category是类别,一 般情况用分类好,用Category去重写类的方法,仅对本Category有效,不会影响到其他类与原有类的关系。

### 分别描述类别(categories)和延展(extensions)是什么?以及两者的区别?继承和类别在实现中有何区别?为什么Category只能为对象添加方法,却不能添加成员变量?

类别: 在没有原类.m文件的基础上,给该类添加方法;

延展:一种特殊形式的类别,主要在一个类的.m文件里声明和实现延展的作用,就是给某类添加私有方法或是私有变量。 两个的区别:延展可以添加属性并且它添加的方法是必须要实现的。延展可以认为是一个私有的类目。

继承和类别在实现中的区别: 类别可以在不获悉,不改变原来代码的情况下往里面添加新的方法,只能添加,不能删除修改。并且如果类别和原来类中的方法产生名称冲突,则类别将覆盖原来的方法,因为类别具有更高的优先级。 Category只能为对象添加方法,却不能添加成员变量的原因:如果可以添加成员变量,添加的成员变量没有办法初始化

### Objective-C有私有方法么?私有变量呢?如多没有的话,有没有什么代替的方法?

objective-c类里面的方法只有两种, 静态方法和实例方法.但是可以通过把方法的声明和定义都放在.m文件中来实现一个表 面上的私有方法。有私有变量,可以通过@private来修饰,或者把声明放到.m文件中。在Objective‐C中,所有实例变量默认都是私有的, 所有实例方法默认都是公有的

### #include与#import的区别? #import与@class的区别?

#import指令是Object-C针对#include的改进版本，#import确保引用的文件只会被引用一次，这样你就不会陷入递归包含的问题中。

#import与@class二者的区别在于：

1> #import会链入该头文件的全部信息，包括实体变量和方法等；而@class只是告诉编译器，其后面声明的名称是类的名称，至于这些类是如何定义的，暂时不用考虑。

2> 在头文件中， 一般只需要知道被引用的类的名称就可以了。 不需要知道其内部的实体变量和方法，所以在头文件中一般使用@class来声明这个名称是类的名称。 而在实现类里面，因为会用到这个引用类的内部的实体变量和方法，所以需要使用#import来包含这个被引用类的头文件。

3> 在编译效率方面考虑，如果你有100个头文件都#import了同一个头文件，或者这些文件是依次引用的，如A–>B, B–>C, C–>D这样的引用关系。当最开始的那个头文件有变化的话，后面所有引用它的类都需要重新编译，如果你的类有很多的话，这将耗费大量的时间。而是用@class则不会。

4> 如果有循环依赖关系，如:A–>B, B–>A这样的相互依赖关系，如果使用#import来相互包含，那么就会出现编译错误，如果使用@class在两个类的头文件中相互声明，则不会有编译错误出现。所以，一般来说，@class是放在interface中的，只是为了在interface中引用这个类，把这个类作为一个类型来用的。 在实现这个接口的实现类中，如果需要引用这个类的实体变量或者方法之类的，还是需要import在@class中声明的类进来.

### 浅复制和深复制的区别? (Difference between shallow copy and deep copy?)

浅层复制(copy)：只复制指向对象的指针，而不复制引用对象本身。意思就是说我有个A对象，复制一份后得到A\_copy对象后，对于浅复制来说，A和A\_copy指向的是同一个内存资源，复制的只不过是是一个指针，对象本身资源还是只有一份，那如果我们对A\_copy执行了修改操作,那么发现A引用的对象同样被修改，这其实违背了我们复制拷贝的一个思想。

深层复制(mutableCopy)：复制引用对象本身。深复制就好理解了,内存中存在了两份独立对象本身, 当修改A时,A\_copy不变。

用网上一哥们通俗的话将就是：

浅复制好比你和你的影子，你完蛋，你的影子也完蛋

深复制好比你和你的克隆人，你完蛋，你的克隆人还活着。

### 类变量的@protected,@private,@public,@package声明各有什么含义?

变量的作用域不同,@protected 该类和所有子类中 的方法可以直接访问这样的变量,这是默认的; @private 该类中的方法可以访问这样的变量,子类不可以; @public除了自己和子类方法外,也可以被其他类或者其他模块中的方法访问; @package 目前尚未得出结论.

### static 关键字的作用?

1> 函数体内 static 变量的作用范围为该函数体，不同于 auto 变量，该变量的内存只被分配一次， 因此其值在下次调用时仍维持上次的值；

2> 在模块内的 static 全局变量可以被模块内所用函数访问，但不能被模块外其它函数访问；

3> 在模块内的 static 函数只可被这一模块内的其它函数调用，这个函数的使用范围被限制在声明 它的模块内；

4> 在类中的 static 成员变量属于整个类所拥有，对类的所有对象只有一份拷贝；

5> 在类中的 static 成员函数属于整个类所拥有，这个函数不接收 this 指针，因而只能访问类的static 成员变量

### 关键字volatile有什么含意?并给出三个不同的例子

一个定义为volatile的变量是说这变量可能会被意想不到地改变，这样，编译器就不会去假设这个变量的值了。精确地说就是，优化器在用到这个变量时必须每次都小心地重新读取这个变量的值，而不是使用保存在寄存器里的备份。下面是volatile变量的几个例子：

• 并行设备的硬件寄存器（如：状态寄存器）

• 一个中断服务子程序中会访问到的非自动变量(Non-automatic variables)

• 多线程应用中被几个任务共享的变量

### Objective-C与C、C+++之间的联系和区别？

Objective-C和C++都是C的面向对象的超集。

Object与C++的区别主要点：Objective-C是完全动态的，支持在运行时动态类型决议(dynamic typing)，动态绑定(dynamic binding)以及动态装载(dynamic loading)；而C++是部分动态的，编译时静态绑定，通过嵌入类(多重继承)和虚函数(虚表)来模拟实现。

Objective-C 在语言层次上支持动态消息转发，其消息发送语法为 [object function]； 而且C++ 为 object->function()。 两者的语义也不同，在 Objective-C 里是说发送消息到一个对象上，至于这个对象能不能响应消息以及是响应还是转发消息都不会 crash； 而在 C++ 里是说对象进行了某个操作，如果对象没有这个操作的话，要么编译会报错(静态绑定)，要么程序会 crash 掉的(动态绑定)。

### 目标-动作机制

目标是动作消息的接收者。一个控件，或者更为常见的是它的单元，以插座变量（参见"插座变量"部分）的形式保有其动作消息的目标。

动作是控件发送给目标的消息，或者从目标的角度看，它是目标为了响应动作而实现的方法. 程序需要某些机制来进行事件和指令的翻译。这个机制就是目标-动作机制。

### 动态绑定

在运行时确定要调用的方法 , 动态绑定将调用方法的确定也推迟到运行时。在编译时，方法的调用并不和代码绑定在一起，只有在消实发送出来之后，才确定被调用的代码。通过动态类型和动态绑 定技术，代码每次执行都可以得到不同的结果。运行时因子负责确定消息的接收者和被调用的方法。运行时的消息分发机制为动态绑定提供支持。当向一个动态类型确定了的对象发送消息时，运行环境系统会通过接收者的isa指针定位对象的类，并以此为起点确定被调用的方法，方法和消息是动态绑定的。而且，不必在Objective-C 代码中做任何工作，就可以自动获取动态绑定的好处。在每次发送消息时，特别是当消息的接收者是动态类型已经确定的对象时，动态绑定就会例行而透明地发生

### iPad开发与iPhone开发有什么不同？

开发iPad和iPhone应用使用的是相同的SDK，它们之间的区别主要体现在设备硬件和UI操作上。

(1)设备硬件：由于iPad不具备电话功能，iPad1也没有相机，所以相关的特性如电话、SMS等无法使用。

(2)UI操作:由于iPad屏幕大小为9.7英寸，而iPhone的屏幕是3.5英寸，这就决定了两者操作方式的不同。有些控件，如UIPopoverController, UISliptViewController 只能在iPad中使用，而不能在iPhone中使用；而且当需要presentModalViewController时，Window presentation style 只对iPad有效；除此之外，iPad产品开发中还需针对iPad大屏幕考虑加入多指手势以增加用户体验

### 换肤技术使用了什么技术?

### 谈谈你对runtime的了解?

### 算法和数据结构

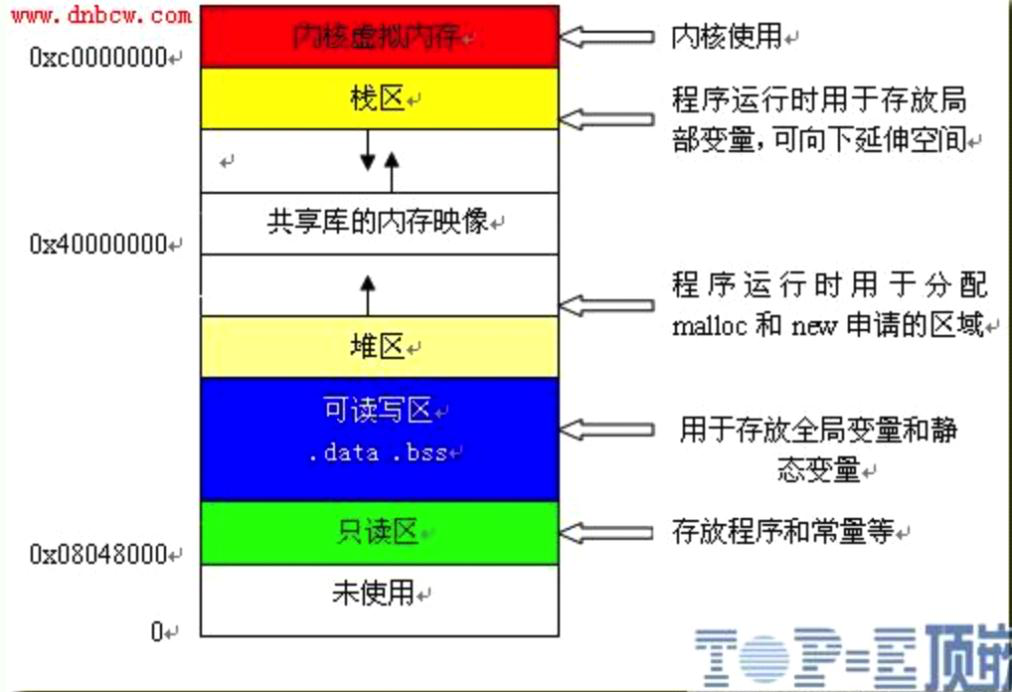
## 内存管理

### 1. 为什么要进行内存管理?

因为移动设备的内存极其有限,当一个程序达到40M时, 系统会发出内存警告. 当程序达到120M时, 程序会闪退, 影响用户体验. 为了保证程序的运行流畅, 必须进行内存管理

### 2. 内存管理的范围?

管理所有继承自NSObject的对象, 对基本数据类型无效. 是因为对象和其他数据类型在系统中存储的空间不一样, 其他局部变量主要存储在栈区(因为基本数据类型占用的存储空间是固定的, 一般存放于栈区 ), 而对象存储于堆中, 当代码块结束时, 这个代码块所涉及到的所有局部变量会自动弹栈清空,  指向对象的指针也会被回收,  这时对象就没有指针指向, 但依然存在于堆内存中, 造成内存泄露.



### 3. 内存管理的原理是什么?

1> 对象的所有权及计数

1) . 对象所有权的概念:

任何对象都有可能拥有一个或者多个所有者,  只要对象还有至少一个所有者, 它就会继续存在

2).  Cocoa所有权策略:

任何自己创建的对象,都归自己所有, 可以使用retain来获取一个对象的所有权,  可以使用”alloc”,”new”,”copy”的方法创建对象

2> 对象的引用计数器

对象里面有retainCount这个实例变量 : 来标识当前有多少个东西来引用这个对象

注: 每个OC对象内部 ,都专门有一个8个字节的存储空间来存储引用计数器, 不分系统, 不管是多少位的系统,  retainCount都是占8个字节

1). 每个对象都有自己的引用计数器

是一个整数表示对象被引用的次数,  即现在有多少东西在引用这个对象. 对象刚被创建时默认计数器值为1,因为一个对象被创建时 Person \* person = [[Person alloc] init]  至少有一个拥有者,就是他自己, 当计数器值变为0时,  对象被销毁

2) . 引用计数器的作用:

判断对象要不要回收的依据, 如果retainCount = 0 , 表示对象需要被回收 (空指针 nil除外)

3> 对象的引用计数器操作:

1). 发送retain消息: 使计数器+1, 该方法返回对象本身

2). 发送release消息: 使计数器-1,

3). 发送retainCount消息: 获得当前对象的引用计数器值

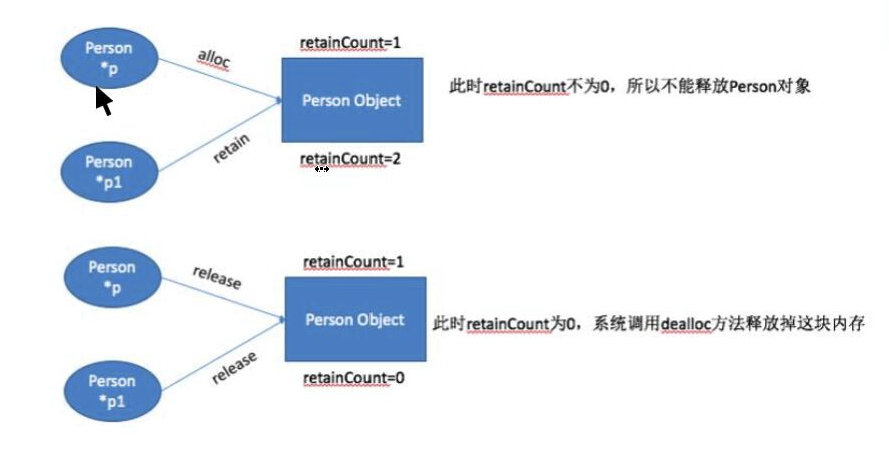
4> 对象销毁

1). 当引用计数器值为0, 那么该对象将会被销毁, 占据的内存空间将会被释放, 如果不为0,这不会被回收(除非整个程序退出)

2). 当对象被销毁时,系统会自动发送一条dealloc 消息, 一般会重写dealloc方法, 在这里释放相关资源

3). 一旦重写了dealloc方法 ,不能直接调用dealloc方法,  必须调用[super dealloc] , 将父类自资源也释放一次,  放在代码块最后调用

4). 一旦对象被回收了,那么他所占据的存储空间就不可再用, 坚持使用程序会崩溃



### 4. objc使用什么机制管理对象内存(或者内存管理方式有哪些)？(重点)

1).MRC(manual retain-release)手动内存管理

2).ARC(automatic reference counting)自动引用计数

3).Garbage collection (垃圾回收)

但是iOS不支持垃圾回收, ARC作为LLVM(编译器的架构系统,用c++编写而成的) 3.0编译器的一项特性, 在iOS5.0 (Xcode4) 版本后推出的自动内存管理, 苹果推荐使用ARC技术来管理内存, 节约时间 , 提高效率 , 减少代码量 , 降低出错几率. 开发者不需要再手动写入retain,release,autorelease三个关键字,手动管理内存, 编译器将自动在代码合适的地方插入retain,release,autorelease进行内存管理.ARC的判断准则, 只要没有强指针指向对象, 对象就会被释放.

### 5. 内存管理的原则

1> 原则:

  1). 只要还有人在使用这个对象, 那么这个对象就不会被回收

2). 只有你想使用这个对象, 那么就应该让这个对象的引用计数器加1

3). 当你不想使用这个对象时, 应该让对象的引用计数器减1

2> 谁创建, 就由谁来release

1). 如果你通过alloc, new, copy 来创建一个对象, 当你不想用这个对象的时候就必须调用release 或者autorelease 让引用计数器减1

2). 不是你创建的就不用你负责 release

3> 谁retain 谁release

只要你调用了retain ,无论这个对象如何生成,  都需要调用release

4> 总结:

有加就应该有减, 曾让某个计数器加1, 就应该让其在最后减1

### 6. 内存管理研究的对象:

1> 野指针

1). 指针变量没有进行初始化

2). 指向的空间已经被释放

 2> 内存泄露

1). 如 Person \* person = [Person new]; 在栈区的person已经被释放, 而堆区new产生的对象还没有释放, 就会造成内存泄露

2). 在MRC手动引用计数器模式下, 内存泄露分4中情况

情况一: OC对象在创建完成使用之后,  没有使用release.

情况二: 没有遵守内存管理的原则, 导致release的次数少于retain次数, 因为只要对象还在使用, 内存就不会被释放 , 导致内存泄露

情况三: 不当的使用nil, 在对象还在使用之前就将对象赋值为nil

情况四: 在函数和方法中错误的使用了retain,release, 在函数中传入的对象进行了retain

 3>.僵尸对象 : 堆中已经被释放的对象

 4>.空指针 : 指针赋值为空,nil

### 7. Objective-C的内存管理? 如何实现内存管理?

如果您通过分配和初始化（比如[[MyClass alloc] init]）的方式来创建对象，您就拥有这个对象，需要负责该对象的释放。这个规则在使用NSObject的便利方法new 时也同样适用。

如果您拷贝一个对象，您也拥有拷贝得到的对象，需要负责该对象的释放。

如果您保持一个对象，您就部分拥有这个对象，需要在不再使用时释放该对象. 反过来，如果您从其它对象那里接收到一个对象，则您不拥有该对象，也不应该释放它

对于这个问题, 也可以从iOS为什么要进行内存管理, 内存管理的范围, 内存管理的对象, 内存管理的方式和内存管理的原理进行一步步分析, 也可以简要回答, 根据具体问题而定.

### 8. 堆和栈的区别?

1> 管理方式：

对于栈来讲，是由编译器自动管理，无需我们手工控制；对于堆来说，释放工作由程序员控制，容易产生memory leak。

2> 申请大小：

栈： 在Windows下,栈是向低地址扩展的数据结构，是一块连续的内存的区域。这句话的意思是栈顶的地址和栈的最大容量是系统预先规定好的，在 WINDOWS下，栈的大小是2M（也有的说是1M，总之是一个编译时就确定的常数），如果申请的空间超过栈的剩余空间时，将提示overflow。因 此，能从栈获得的空间较小。

堆：堆是向高地址扩展的数据结构，是不连续的内存区域。这是由于系统是用链表来存储的空闲内存地址的，自然是不连续的，而链表的遍历方向是由低地址向高地址。堆的大小受限于计算机系统中有效的虚拟内存。由此可见，堆获得的空间比较灵活，也比较大。

3> 碎片问题：

对于堆来讲，频繁的new/delete势必会造成内存空间的不连续，从而造成大量的碎片，使程序效率降低。对于栈来讲，则不会存在这个问题，因为栈是先进后出的队列，他们是如此的一一对应，以至于永远都不可能有一个内存块从栈中间弹出

4> 分配方式：

堆都是动态分配的，没有静态分配的堆。栈有2种分配方式：静态分配和动态分配。静态分配是编译器完成的，比如局部变量的分配。动态分配由alloca函数进行分配，但是栈的动态分配和堆是不同的，他的动态分配是由编译器进行释放，无需我们手工实现。

5> 分配效率：

栈是机器系统提供的数据结构，计算机会在底层对栈提供支持：分配专门的寄存器存放栈的地址，压栈出栈都有专门的指令执行，这就决定了栈的效率比较高。堆则是C/C++函数库提供的，它的机制是很复杂的。

## Objective-C优点和缺点

优点:  1)Cateogies 2)Posing 3)动态识别 4)指标计算 5)弹性讯息传递  6)不是一个过度复杂的C衍生语言 7)Objective-C与C++可混合编程

缺点:  1)不支援命名空間 2)不支持运算符重载 3)不支持多重继承 4)使用动态运行时类型,所有的方法都是函数调用,所以很多编译时优化方法都用不到。(如内联函数等),性能低劣。

# Swift（★）

## Swift和Objective-C的联系

1> Swift与Objective-C共用同一套运行时环境

我们编写程序，让程序运行起来，被机器执行的代码并非全部是由我们自己来编写的。需要同时运行很多预先写好的支持性的代码，才能让我们自己的代码运行起来。程序并非单独存在的，运行时处在一定的环境当中。我总联想到很多小蚂蚁在泥土上面爬，而我自己写的程序只是其中的一只。

Swift跟Objective-C编译出的程序代码运行在同一套运行环境上面。Swift的类型可以桥接到Objective-C的类型，反之亦然。Swift编写的代码可以调用Objective-C编写的代码，反之也一样。

Objective-C之前积累下来的大量类库，实现不用改写Swift就可以直接调用。

2> 同一个工程，可以同时使用Swift和Objective-C

Objective-C在一端，Swift在另一端，两端经中间文件进行桥接。桥接文件包含Objective-C的头文件，编译时自动转成Swift可以识别的形式。Swift就可以使用Objective-C的类和它的函数。

在Swift的类中，加上@objc（类名）的字样，Objective-C也可以使用Swift编写的类。但Swift跟C++的相互调用，需要Objective-C来封装。

3> Swift骨子里大多与Objective-C一样

Objective-C出现过的绝大多数概念，比如引用记数、ARC、属性、协议、接口、初始化、扩展类、命名参数、匿名函数等，在Swift中继续有效（可能只是换了个术语）。我自己将Swift看成是Objective-C的一块大大的语法糖，其他人可能有不同感受。

Swift大多数概念与Objective-C一样，也有些概念在Objective-C找不到对应，比如泛型。Swift中将那种操作写一次就可以作用多个类型的语法叫做Generics（泛型）。

## Swift比Objective-C有什么优势？

1>, Swift 容易阅读

不再需要行尾的分号，以及 if/else 语句中围绕条件表达式的括弧。另外就是方法的调用不再互相嵌套成中括号[[[ ]]]。Swift 中的方法和函数的调用使用行业内标准的在一对括弧内使用逗号分隔的参数列表。这样做的结果就是一种带有简化了句法和语法的更加干净有表现力的语言。

2>, Swift 更易于维护

Swift 丢掉了对着两个文件的要求。 Xcode编译器可以自动计算出以来并执行增量构建。如此，将头文件 同实现文件相分离。把 Objective-C 头文件(.h) 和实现文件 (.m) 合并成了一个代码文件 (.swift)

3>, Swift 更加安全

Swift代码中的可选类型使得一个nil可选值的可能性变得非常的明确, 这意味它能在你写下一段糟糕的代码时会生成一个编译器错误. 这就建立了一种短程反馈的循环，可以让程序员带着目标去写代码. 问题在代码被写就时就可以被修复, 这大大节省了你要在修复有关来自 Objective-C 指针逻辑的bug时需要耗费的时间和金钱.

4>, Swift 代码更少

Swift 减少了重复性语句和字符串操作所需要的代码量。在 Objective-C 中, 使用文本字符串将两块信息组合起来的操作非常繁琐。Swift 采用当代编程语言的特性，比如使用“+”操作符将两个字符串加到一起

Swift中的类型系统减少了代码语句的复杂性--作为编译器可以理解的类型。比如，Objective-C要求程序员记住特殊字符标记（%s，%d，%@）并且提供了一个用逗号分隔的变量来代替每个标记。Swift支持字符串插入，这就消除了需要记住的标记和允许程序员直接插入变量到面向用户的字符串中

5>, Swift 速度更快

删除遗留下来的C语言约定大大提升了引擎盖之下Swift的性能, 进行计算密集型任务的性能上，Swift已经逼近C++的表现, 将近是OC运行速度的1.4倍.

总结: 使用 Swift，程序员只要维护原来一半量的代码文件，手动的代码同步工作为零，标点输入出错的概率也远远低于以前 -- 这样就能腾出更多的时间写高质量的代码。通过使用可选类型 —— 一种针对返回或不返回值的编译时安全机制，而返回值是同步操作、网络失效时无效的用户输入以及数据验证错误发生时普遍会遇到的问题。ARC 在 Swift 中对过程式 C 风格的代码，还有苹果公司 Cocoa 框架使用的面向对象代码都进行了统一。

## Swift的内存管理是怎样的？

Swift 使用自动引用计数（Automatic Reference Counting, ARC）来简化内存管理, 这种内存管理方式相比GC而言，对程序员的要求较高, 并且ARC比GC更容易引起编程错误，但却比GC快。尤其在性能很重要的场合。

## Swift 支持面向过程编程吗？

它采用了 Objective-C 的命名参数以及动态对象模型，可以无缝对接到现有的 Cocoa 框架，并且可以兼容 Objective-C 代码，支持面向过程编程和面向对象编程。

## 举例说明Swift里面有哪些是 Objective-C中没有的？

Swift引入了在Objective-C中没有的一些高级数据类型，例如tuples（元组），可以使你创建和传递一组数值。swift还引入了可选项类型（Optionals），用于处理变量值不存在的情况。可选项的意思有两种：一是变量是存在的，例如等于X，二是变量值根本不存在。Optionals类似于Objective-C中指向nil的指针，但是适用于所有的数据类型，而非仅仅局限于类，Optionals 相比于Objective-C中nil指针更加安全和简明，并且也是Swift诸多最强大功能的核心。

## Swift 是一门安全语言吗？

Swift是一门类型安全的语言，Optionals就是代表。Swift能帮助你在类型安全的环境下工作，如果你的代码中需要使用String类型，Swift的安全机制能阻止你错误的将Int值传递过来，这使你在开发阶段就能及时发现并修正问题。

## 为什么要在变量类型后面加个问号？

用来标记这个变量的值是可选的，一般用“！”和“？”定义可选变量的区别：用“！”定的可选变量必须保证转换能够成功，否则报错，但定义的变量可以直接使用，不会封装在option里；而用“？”号定的可选变量即使转换不成功本身也不会出错，变量值为nil，如果转换成功，要使用该变量进行计算时变量名后需要加“！”

## 什么是泛型，它们又解决了什么问题？

泛型是用来使代码能安全工作。在Swift中，泛型可以在函数数据类型和普通数据类型中使用，例如类、结构体或枚举。

泛型解决了代码复用的问题。有一种常见的情况，你有一个方法，需要一个类型的参数，你为了适应另一种类型的参数还得重新再写一遍这个方法。

比如，在下面的代码中，第二个方法是第一个方法的“克隆体”：

1. func areIntEqual(x: Int, \_ y: Int) -> Bool {
2. return x == y
3. }
4. func areStringsEqual(x: String, \_ y: String) -> Bool {
5. return x == y
6. }
7. areStringsEqual("ray", "ray") // true
8. areIntEqual(1, 1) // true

一个Objective-C开发者可能会采用NSObject来解决问题：

1. import Foundation
2. func areTheyEqual(x: NSObject, \_ y: NSObject) -> Bool {
3. return x == y
4. }
5. areTheyEqual("ray", "ray") // true
6. areTheyEqual(1, 1) // true

这段代码能达到了目的，但是编译的时候并不安全。它允许一个字符串和一个整型数据进行比较：

1. areTheyEqual(1, "ray")

程序可能不会崩溃，但是允许一个字符串和一个整型数据进行比较可能不会得到想要的结果。

采用泛型的话，你可以将上面两个方法合并为一个，并同时还保证了数据类型安全。这是实现代码：

1. func areTheyEqual<T: Equatable>(x: T, \_ y: T) -> Bool {
2. return x == y
3. }
4. areTheyEqual("ray", "ray")
5. areTheyEqual(1, 1)

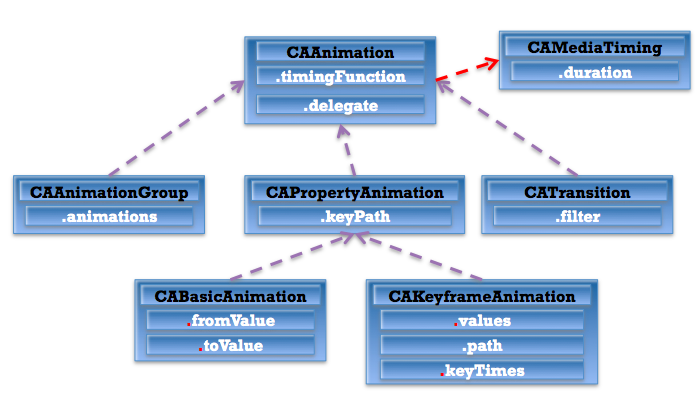
# UI基础（★★）

学习目标,UI布局-动画效果-逻辑控制-数据存储-和各端的对接-整体架构

### 怎么解决缓存池满的问题(cell)

iOS中不存在缓存池满的情况，因为通常我们ios中开发，对象都是在需要的时候才会创建，有种常用的说话叫做懒加载，还有在UITableView中一般只会创建刚开始出现在屏幕中的cell，之后都是从缓存池里取，不会在创建新对象。缓存池里最多也就一两个对象，缓存池满的这种情况一般在开发java中比较常见，java中一般把最近最少使用的对象先释放。

### CAAnimation的层级结构



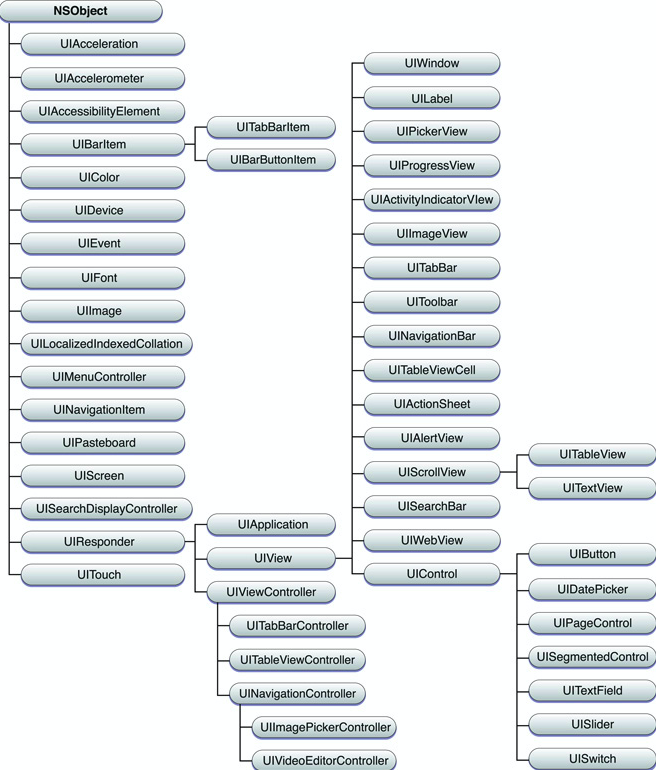
### UIButton与UITableView的层级结构

继承结构，属于内部的子控件结构

UIButton为：UIButton > UIControl > UIView > UIResponder > NSObject

UITableView为：UITableView > UIScrollView > UIView > UIResponder > NSObject

整体的结构为：



### 设置scroll view的contensize能在Viewdidload里设置么,为什么

一般情况下可以设置在viewDidLoad中，但在autolayout下，系统会在viewDidAppear之前根据subview的constraint重新计算scrollview的contentsize。 这就是为什么，在viewdidload里面手动设置了contentsize没用。因为在后面，会再重新计算一次，前面手动设置的值会被覆盖掉。

解决办法就是：

1.去除autolayout选项，自己手动设置contentsize

2.如果要使用autolayout，要么自己设置完subview的constraint，然后让系统自动根据constraint计算出contentsize。 要么就在viewDidAppear里面自己手动设置contentsize。

# UI高级（★★★）

## 一. UI高级语法

### 简述你对UIView、UIWindow和CALayer的理解

UIView: 属于UIkit.framework框架,负责渲染矩形区域的内容,为矩形区域添加动画,响应区域的触摸事件,布局和管 理一个或多个子视图

UIWindow: 属于UIKit.framework框架,是一种特殊的UIView,通常在一个程序中只会有一个UIWindow,但可以手 动创建多个UIWindow,同时加到程序里面。UIWindow在程序中主要起到三个作用:

1)作为容器,包含app所要显示的所有视图

2)传递触摸消息到程序中view和其他对象

3)与UIViewController协同工作,方便完成设备方向旋转的支持

CAlayer: 属于QuartzCore.framework,是用来绘制内容的,对内容进行动画处理依赖与UIView来进行显示,不能处 理用户事件。UIView和CALayer是相互依赖的,UIView依赖CALayer提供内容,CALayer依赖UIView一共容器显示 绘制内容。

延伸: UIViewController: 每个视图控制器都有一个自带的视图,并且负责这个视图相关的一切事务。方便管理视图中的子视图,负责model与view的通信;检测设备旋转以及内存警告;是所有视图控制类的积累,定义了控制器的基本功能。

### frame和bounds有什么不同？(Difference between frame and bounds?)

frame指的是：该view在父view坐标系统中的位置和大小。（参照点是父亲的坐标系统）

bounds指的是：该view在本身坐标系统中 的位置和大小。（参照点是本身坐标系统）

### 方法和选择器有何不同？(Difference between method and selector?)

selector是一个方法的名字，method是一个组合体，包含了名字和实现.

### 关于页面间传值的问题?

举例：页面A和页面B之间的传值，如果在B页面中有一个按钮实现点击按钮pop到A页面并改变A页面的背景颜色，然后分别说明如何使用通知, 代理, 属性, block进行传值.

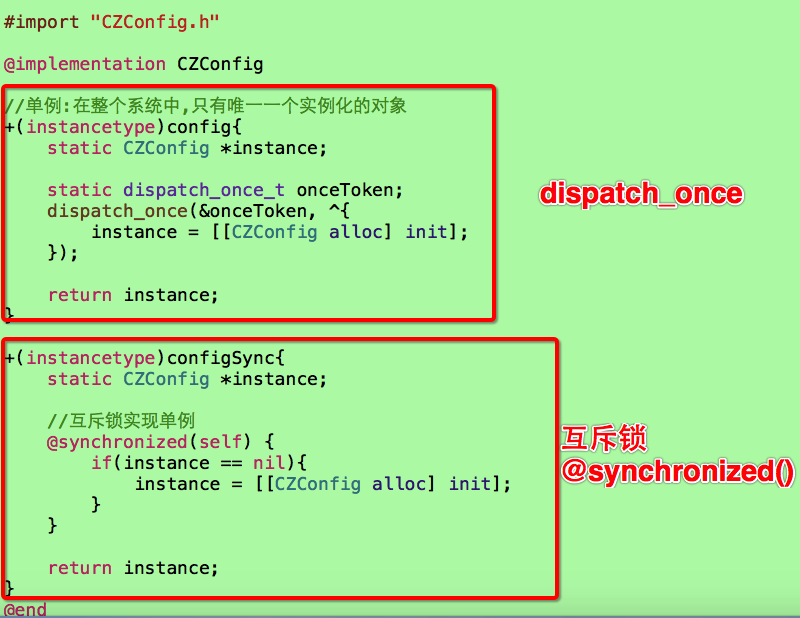
对于通知代理面试常问, 代理和通知分别在什么情况下使用? 区别? 各自优点?

### 什么是懒加载? (What is lazy loading?)

只在用到的时候才去初始化。也可以理解成延时加载。我觉得最好也最简单的一个列子就是tableView中图片的加载显示了。

一个延时载，避免内存过高，一个异步加载，避免线程堵塞。

### 手写一下单例方法(或者单例模式的设计：GCD 方式和同步锁方式的区别在哪里？unlock 呢？GCD 是怎么保证单例的？)



### 关于视图的生命周期的问题

1>.首先判断控制器是否有视图，如果没有就调用loadView方法创建：通过storyboard或者代码；

2>.随后调用viewDidLoad，可以进行下一步的初始化操作；只会被调用一次；

3>.在视图显示之前调用viewWillAppear；该函数可以多次调用；

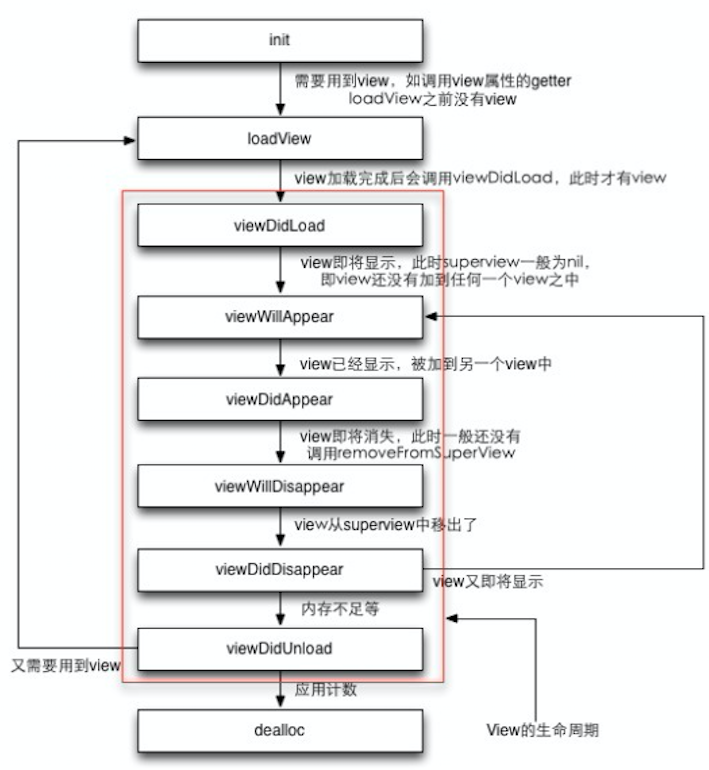
4>.视图viewDidAppear

5>.在视图显示之前调用viewWillDisappear；该函数可以多次调用；

如需要）；

6>.在布局变化前后，调用viewWill/DidLayoutSubviews处理相关信息；

7>.视图生命周期图



### 响应者链条? (What is responder chain?)

事件响应链。包括点击事件，画面刷新事件等。在视图栈内从上至下，或者从下之上传播. 可以说点事件的分发，传递以及处理。具体可以去看下touch事件这块。

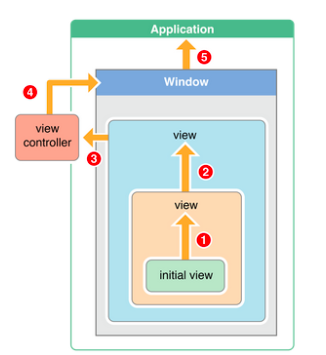
1> 首先解释响应者链的概念

UIResponder类，是UIKIT中一个用于处理事件响应的基类。窗口上的所有事件触发，都由该类响应（即事件处理入口）。所以，窗口上的 View及控制器都是派生于该类的，例如UIView、UIViewController等。调用UIResponder类提供的方法或属性，我们就可以捕捉到窗口上的所有响应事件，并进行处理。

响应者链条是由多个响应者对象连接起来的链条，其中响应者对象是能处理事件的对象，所有的View和ViewController都是响应者对象，利用响应者链条能让多个控件处理同一个触摸事件.

2> 事件传递机制

如果当前view不能处理当前事件，那么事件将会沿着响应者链(Responder Chain)进行传递，知道遇到能处理该事件的响应者(Responsder Object)。



1) 接收事件的initial view如果不能处理该事件并且她不是顶层的View，则事件会往它的父View进行传递。

2)．initial view的父View获取事件后如果仍不能处理，则继续往上传递，循环这个过程。如果顶层的View还是不能处理这个事件的话，则会将事件传递给它们的ViewController，

3). 如果ViewController也不能处理，则传递给Window(UIWindow)，此时Window不能处理的话就将事件传递UIApplication，最后如果连Application也不能处理，则废弃该事件

### C语言的函数调用和oc的消息机制有什么区别?

### loadView 和 viewDidLoad 的区别?

### ViewController 的 loadView, viewDidLoad, viewDidUnload 分别是在什么时候调用的？在自定义ViewController的时候这几个函数里面应该做什么工作？

viewDidLoad在view 从nib文件初始化时调用，loadView在controller的view为nil时调用。此方法在编程实现view时调用, view 控制器默认会注册memory warning notification,当view controller的任何view 没有用的时候，viewDidUnload会被调用，在这里实现将retain 的view release,如果是retain的IBOutlet view 属性则不要在这里release,IBOutlet会负责release 。

### 16 实现过哪些自定义控件？

### 17 导航条透明度

### 18 Cover Flow 效果演示

## 二. UITableView与UICollectionView

### 1.UITableView的重用机制?(或者如何在一个view上显示多个tableView,tableView要求不同的数据源以及不同的样式 (要求自定义cell), 如何组织各个tableView的delegate和dataSource?请说说实现思路?)

查看UITableView头文件,会找到NSMutableArray \* visiableCells,和NSMutableArray \* reusableTableCells两 个结构。visiableCells内保存当前显示的cells,reusableTableCells保存可重用的cells。TableView显示之初,reusableTableCells为空,那么 tableView dequeueReusableCellWithIdentifier:CellIdentifier返回nil。开始的cell都 是通过 [[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] 来创 建,而且cellForRowAtIndexPath只是调用最大显示cell数的次数。 比如:有100条数据,iPhone一屏最多显示10个cell。程序最开始显示TableView的情况是:

1. 用[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] 创建10次cell,并给cell指定同样的重用标识(当然,可以为不同显示类型的cell指定不同的标识)。并且10个cell全部都加 入到 visiableCells数组,reusableTableCells为空。

2. 向下拖动tableView,当cell1完全移出屏幕,并且 cell11(它也是alloc出来的,原因同上)完全显示出来的时候。 cell11加入到visiableCells,cell1移出 visiableCells,cell1加入到reusableTableCells。

3. 接着向下拖动tableView,因为reusableTableCells中已 经有值,所以,当需要显示新的cell, cellForRowAtIndexPath再次被调用的时 候, tableView dequeueReusableCellWithIdentifier:CellIdentifier,返回cell1。 cell1加入到visiableCells,cell1 移出reusableTableCells;cell2移出 visiableCells,cell2加入到reusableTableCells。之后再需要显示的Cell就可 以正常重用了

### 在一个tableView 中需要自定义多种样式的cell(两种或三种),通常你如何实现,说说思路即可?

比如:有100条数据,iPhone一屏最多显示10个cell。程序最开始显示TableView的情况是:

1. 用[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] 创建10次cell,并给cell指定同样的重用标识(当然,可以为不同显示类型的cell指定不同的标识)。并且10个cell全部都加 入到 visiableCells数组,reusableTableCells为空。

2. 向下拖动tableView,当cell1完全移出屏幕,并且 cell11(它也是alloc出来的,原因同上)完全显示出来的时候。 cell11加入到visiableCells,cell1移出 visiableCells,cell1加入到reusableTableCells。

3. 接着向下拖动tableView,因为reusableTableCells中已 经有值,所以,当需要显示新的cell, cellForRowAtIndexPath再次被调用的时 候, tableView dequeueReusableCellWithIdentifier:CellIdentifier,返回cell1。 cell1加入到visiableCells,cell1 移出reusableTableCells;cell2移出 visiableCells,cell2加入到reusableTableCells。之后再需要显示的Cell就可 以正常重用了

### 3. UITableView的性能优化? 滑动的时候有种卡的感觉是为什么？怎么解决？

然而在使用第三方应用时，却经常遇到性能上的问题，普遍表现在滚动时比较卡，特别是table cell中包含图片的情况时。 实际上针对性地优化一下就可以解决tableView滑动的时候卡顿的问题, 在iOS应用中，UITableView应该是使用率最高的视图之一了。iPod、时钟、日历、备忘录、Mail、天气、照片、电话、短信、 Safari、App Store、iTunes、Game Center⋯几乎所有自带的应用中都能看到它的身影，可见它的重要性。 然而在使用第三方应用时，却经常遇到性能上的问题，普遍表现在滚动时比较卡，特别是table cell中包含图片的情况时。

实际上只要针对性地优化一下:

1>同一时间其实只需要存在一屏幕的cell对象即可，不需要为每一行创建一个cell。

UITableView是UIScrollView的子类，因此它可以自动响应滚动事件（一般为上下滚动）。 它内部包含0到多个UITableViewCell对象，每个table cell展示各自的内容。当新cell需要被显示时，就会调用tableView:cellForRowAtIndexPath:方法来获取或创建一个 cell；而不可视时，它又会被释放。由此可见，同一时间其实只需要存在一屏幕的cell对象即可，不需要为每一行创建一个cell。 此 外，UITableView还可以分为多个sections，每个区段都可以有自己的head、foot和cells。而在定位一个cell时，就需要2 个字段了：在哪个section，以及在这个section的第几行。这在iOS SDK中是用NSIndexPath来表述的，UIKit为其添加了indexPathForRow:inSection:这个创建方法。 其他诸如编辑之类的就不提了，因为和本文无关。

介绍完原理，接下来就开始优化吧。

**使用不透明视图。**

不透明的视图可以极大地提高渲染的速度。因此如非必要，可以将table cell及其子视图的opaque属性设为YES（默认值）。

其中的特例包括背景色，它的alpha值应该为1（例如不要使用clearColor）；图像的alpha值也应该为1，或者在画图时设为不透明。

**不要重复创建不必要的table cell。**

前面说了，UITableView只需要一屏幕的UITableViewCell对象即可。因此在cell不可见时，可以将其缓存起来，而在需要时继续使用它即可。

而UITableView也提供了这种机制，只需要简单地设置一个identifier即可：

static NSString \*CellIdentifier = @"xxx"; UITableViewCell \*cell = [tableView dequeueReusableCellWithIdentifier:CellIdentifier];if (cell == nil) { cell = [[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier] autorelease]; }  值得一提的是，cell被重用时，它内部绘制的内容并不会被自动清除，因此你可能需要调用setNeedsDisplayInRect:或setNeedsDisplay方法。

此 外，在添加table cell的时候，如果不需要动画效果，最好不要使用insertRowsAtIndexPaths:withRowAnimation:方法，而是直接调 用reloadData方法。因为前者会对所有indexPaths调用tableView:cellForRowAtIndexPath:方法，即便该 cell并不需要显示（不知道是不是bug），这就可能创建大量多余的cell。勘误：只是在模拟器上测试如此，真机调试时没有这种bug。

**减少视图的数目。**

UITableViewCell包含了textLabel、detailTextLabel和imageView等view，而你还可以自定义一些视图放在它的contentView里。然而view是很大的对象，创建它会消耗较多资源，并且也影响渲染的性能。

如果你的table cell包含图片，且数目较多，使用默认的UITableViewCell会非常影响性能。奇怪的是，使用自定义的view，而非预定义的view，明显会快些。

当然，最佳的解决办法还是继承UITableViewCell，并在其drawRect:中自行绘制：

- (void)drawRect:(CGRect)rect { if (image) { [image drawAtPoint:imagePoint]; self.image = nil; } else { [placeHolder drawAtPoint:imagePoint]; } [text drawInRect:textRect withFont:font lineBreakMode:UILineBreakModeTailTruncation]; }  不过这样一来，你会发现选中一行后，这个cell就变蓝了，其中的内容就被挡住了。最简单的方法就是将cell的selectionStyle属性设为UITableViewCellSelectionStyleNone，这样就不会被高亮了。

此 外还可以创建CALayer，将内容绘制到layer上，然后对cell的contentView.layer调用addSublayer:方法。这个例 子中，layer并不会显著影响性能，但如果layer透明，或者有圆角、变形等效果，就会影响到绘制速度了。解决办法可参见后面的预渲染图像。

**不要做多余的绘制工作。**

在实现drawRect:的时候，它的rect参数就是需要绘制的区域，这个区域之外的不需要进行绘制。

例如上例中，就可以用CGRectIntersectsRect、CGRectIntersection或CGRectContainsRect判断是否需要绘制image和text，然后再调用绘制方法。

**预渲染图像。**

你会发现即使做到了上述几点，当新的图像出现时，仍然会有短暂的停顿现象。解决的办法就是在bitmap context里先将其画一遍，导出成UIImage对象，然后再绘制到屏幕，详细做法可见[《利用预渲染加速iOS设备的图像显示》](http://www.keakon.net/2011/07/26/%E5%88%A9%E7%94%A8%E9%A2%84%E6%B8%B2%E6%9F%93%E5%8A%A0%E9%80%9FiOS%E8%AE%BE%E5%A4%87%E7%9A%84%E5%9B%BE%E5%83%8F%E6%98%BE%E7%A4%BA)。

**不要阻塞主线程。**

做到前几点后，你的table view滚动时应该足够流畅了，不过你仍可能让用户感到不爽。常见的现象就是在更新数据时，整个界面卡住不动，完全不响应用户请求。

出现这种现象的原因就是主线程执行了耗时很长的函数或方法，在其执行完毕前，无法绘制屏幕和响应用户请求。其中最常见的就是网络请求了，它通常都需要花费数秒的时间，而你不应该让用户等待那么久。

解 决办法就是使用多线程，让子线程去执行这些函数或方法。这里面还有一个学问，当下载线程数超过2时，会显著影响主线程的性能。因此在使用 ASIHTTPRequest时，可以用一个NSOperationQueue来维护下载请求，并将其 maxConcurrentOperationCount设为2。而NSURLRequest则可以配合[GCD](http://developer.apple.com/library/ios/#documentation/Performance/Reference/GCD_libdispatch_Ref/Reference/reference.html)来实现，或者使用NSURLConnection的setDelegateQueue:方法。

当然，在不需要响应用户请求时，也可以增加下载线程数，以加快下载速度：

- (void)scrollViewDidEndDragging:(UIScrollView \*)scrollView willDecelerate:(BOOL)decelerate { if (!decelerate) { queue.maxConcurrentOperationCount = 5; } } - (void)scrollViewDidEndDecelerating:(UIScrollView \*)scrollView { queue.maxConcurrentOperationCount = 5; } - (void)scrollViewWillBeginDragging:(UIScrollView \*)scrollView { queue.maxConcurrentOperationCount = 2; }  此外，自动载入更新数据对用户来说也很友好，这减少了用户等待下载的时间。例如每次载入50条信息，那就可以在滚动到倒数第10条以内时，加载更多信息：

- (void)tableView:(UITableView \*)tableView willDisplayCell:(UITableViewCell \*)cell forRowAtIndexPath:(NSIndexPath \*)indexPath { if (count - indexPath.row < 10 && !updating) { updating = YES; [self update]; } }// update方法获取到结果后，设置updating为NO  还有一点要注意的就是当图片下载完成后，如果cell是可见的，还需要更新图像：

NSArray \*indexPaths = [self.tableView indexPathsForVisibleRows];for (NSIndexPath \*visibleIndexPath in indexPaths) { if (indexPath == visibleIndexPath) { MyTableViewCell \*cell = (MyTableViewCell \*)[self.tableView cellForRowAtIndexPath:indexPath]; cell.image = image; [cell setNeedsDisplayInRect:imageRect]; break; } }// 也可不遍历，直接与头尾相比较，看是否在中间即可。  最后还是前面所说过的insertRowsAtIndexPaths:withRowAnimation:方法，插入新行需要在主线程执行，而一次插入很多行的话（例如50行），会长时间阻塞主线程。而换成reloadData方法的话，瞬间就处理完了。l

### 4 tableview的cell里如何嵌套collection view？

思路同网易新闻类似, 用自定义的继承自UITableViewCell的类, 在initWithFrame的构造方法中, 初始化自定义的继承自UICollectionView的类

### 5 下拉和上拉的原理?

上拉和下拉的原理可以参照新浪微博的上拉和下拉刷新,

以tableView的上拉刷新为例:

1> 为了进行无缝阅读, 通过tableView的代理方法, willDisplayCell判断是否是最后一行,

2> 如果是最后一行, 在显示最后一行的同时, 判断当前是否存在上拉刷新

3> 如果当前没有上拉刷新, 就进行加载数据, 启动”橘花”

以tableView的下拉刷新为例:

1> 判断当前的上拉刷新视图是否动画

2> 如果没有动画, 就不是上拉刷新

3> 然后下拉刷新加载数据

4> 加载完毕数据关闭刷新

### 关于cell的复用问题? cell的复用机制?

TableView的重用机制，为了做到显示和数据分离，IOS tableView的实现并且不是为每个数据项创建一个tableCell。而是只创建屏幕可显示最大个数的cell，然后重复使用这些cell，对cell做单独的显示配置，来达到既不影响显示效果，又能充分节约内容的目的。下面简要分析一下它的实现原理。重用实现分析

　　查看UITableView头文件，会找到NSMutableArray\*  visiableCells，和NSMutableDictnery\* reusableTableCells两个结构。visiableCells内保存当前显示的cells，reusableTableCells保存可重用的cells。

　TableView显示之初，reusableTableCells为空，那么tableView dequeueReusableCellWithIdentifier:CellIdentifier返回nil。开始的cell都是通过[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier]来创建，而且cellForRowAtIndexPath只是调用最大显示cell数的次数。

　　比如：有100条数据，iPhone一屏最多显示10个cell。程序最开始显示TableView的情况是：

　　1>. 用[[UITableViewCell alloc] initWithStyle:UITableViewCellStyleDefault reuseIdentifier:CellIdentifier]创建10次cell，并给cell指定同样的重用标识(当然，可以为不同显示类型的cell指定不同的标识)。并且10个cell全部都加入到visiableCells数组，reusableTableCells为空。

　　2>. 向下拖动tableView，当cell1完全移出屏幕，并且cell11(它也是alloc出来的，原因同上)完全显示出来的时候。cell11加入到visiableCells，cell1移出visiableCells，cell1加入到reusableTableCells。

　　3>. 接着向下拖动tableView，因为reusableTableCells中已经有值，所以，当需要显示新的cell，cellForRowAtIndexPath再次被调用的时候，tableView dequeueReusableCellWithIdentifier:CellIdentifier，返回cell1。cell1加入到visiableCells，cell1移出reusableTableCells；cell2移出visiableCells，cell2加入到reusableTableCells。之后再需要显示的Cell就可以正常重用了。

所以整个过程并不难理解，但需要注意正是因为这样的原因：配置Cell的时候一定要注意，对取出的重用的cell做重新赋值，不要遗留老数据。

### 8. 如何实现cell的动态的行高？

如果希望每条数据显示自身的行高, 必须设置两个属性, 1.预估行高, 2.自定义行高

设置预估行高

tableView.estimatedRowHeight = 200

设置定义行高

tableView.rowHeight = UITableViewAutomaticDimension

如果要让自定义行高有效, 必须让容器视图有一个自下而上 的约束

## 三. 关于UIWebView的问题:

### 1. 谈谈webView的作用

iOS开发中webview和native code的配合上的一些经验和技巧 , 首先提两句两者的优劣｡webview与运维成本低， 更新几乎不依赖App的版本；但在交互和性能上与跟native code有很大差距。native code与之对应。

注，我这里不说HTML5，因为我认为，HTML5确实给web带入了一个新时代。这个时代是什么，web app。也就是说，只有脱离native的这个前提，在浏览器的环境下，HTML5的意义才能显现，而我们讨论iOS App的时候，HTML5显然没什么意义。

不管是用webview还是native code，我有两个原则：

1>，用户体验不打折

2>，运维成本低

注，为什么不提开发成本。因为做web开发和iOS开发根本就是两回事。当然，web开发发展了这么多年，对于某些功能实现是要比native app快。但多数情况，同一个功能，对于iOS开发者和web开发者，用各自擅长的方式开发成本都最低，所以说某个功能开发成本低，往往是一个伪命题。

刚刚说了，webview的优势在于更新不依赖版本，那么在一款App中，只有会频繁更新的界面考虑webview才有意义。那么哪些界面会频繁更新，这就要因App而异了，我只说两年来，我接触到的一些。

首页。首页资源可谓必争之地，内容一天一换是正常现象，一天几换也不稀奇。而如果仅仅是内容的更换，非要上个webview就显得有些激进了。而事实上首页的变化千奇百怪，逢年过节变个脸，特殊情况挂个公告，偶尔还要特批强推一把某个业务，等等。 此前，我在设计App首页的时候，把首页配置设计的非常复杂。App端要处理n种情况，n各参数，server端要记住n种规则，直到一天，我崩溃了，把首页完全换成webview，才豁然开朗。

活动页。做互联网都知道，活动，是一个最常见的运营手段。特点是，周期短，功能少，但基本不能复用。这些特点都标识了活动不适合做native，要用webview实现。 即使有人告诉你说，我的活动是一个长期活动而且形式不变，也不要相信他。因为在第二期，第三期，第四期他会分别加上一些非常诡异，却有很合理的小变更，而这些变更是你在那个版本根本无法实现的。

试水的新功能。这种界面，往往设计不成熟，需要在运行过程中不断收集用户反馈，更新升级，甚至决定去留。所以，只有webview才能hold住如此不稳定的功能。 切记在一个功能还没有确定之前，不要大张旗鼓单位开发native code，要知道，你写的这些代码，三天后就要改一遍，而且要发布上线。

富文本内容。这个不用多说了吧，按照HTML的常用标签做一个webtext可不是小工程。而且富文本的变化太多了，一点无法匹配，都会导致整个界面巨丑。

OK，上边说了我认为最该使用webview的4个界面，分别带有不同的特点，但webview的交互是个短板，因此webview在一个App中，只能作为界面，不允许在界面中出现动作。而一个webview的界面如何跟native code结合起来呢，我的答案是，超链接。在webview上点击超链接，会调用webview delegate的shouldload方法，自这里拦截请求，进行处理。由于webview的链接都是URL，因此我建议，把整个App的界面都用URL管理起来。从320框架对VC的管理中获得灵感，构建一个新的应用，总是先制定一套协议，封装一个方法，每一次VC的切换都通过URL。如此一来，server与App的交互就简单了许多，webview也如是。

最后就是长相问题，webview很难长成native的view。而我的方案是，长不成也要装成。在一些情况下，禁用webview滚动，使用滚动框架（iScroll不错）去实现。webview上下留出200pixel的空白背景，y从-200开始。否则大家知道，webview上下会有阴影的背景，不藏起来会很丑。等等，还有很多其他的方法去伪装webview，是要视情景而用。

## 四.关于动画

### 1. 谈谈你对Core Graphic 绘图的了解?

CoreGraphics也称为Quartz 2D 是UIKit下的主要绘图系统，频繁的用于绘制自定义视图。Core Graphics是高度集成于UIView和其他UIKit部分的。Core Graphics数据结构和函数可以通过前缀CG来识别。

视图可以通过子视图、图层或实现drawRect：方法来表现内容，如果说实现了drawRect：方法，那么最好就不要混用其他方法了，如图层和子视图。自定义绘图大部分是由UIKit或者Core Graphics来实现的。

2D绘图一般可以拆分成以下几个操作: 线条 , 路径 , 文本 , 图片 , 渐变

由于像素是依赖于目标的，所以2D绘图并不能操作单独的像素，我们可以从上下文（Context）读取它。

绘图就好比在画布上拿着画笔机械的进行画画，通过制定不同的参数来进行不同的绘制。

<http://www.tuicool.com/articles/jIJzMf>

<http://blog.csdn.net/mangosnow/article/details/37054765>

### 2. Core Animation(核心动画)?

CoreAnimation也就是核心动画, 是一组非常强大的动画处理API, 可以使用少量的代码做出绚丽的效果, 是直接作用在CALayer上的, 并非UIView, 并且Core Animation的动画执行过程都是在后台操作, 不会阻塞主线程.

所有动画都是作用在CALayer上的, 当把动画添加到Layer上, 是不直接修改它的属性, Core Animation维护了两个平行layer的层次结构, 模型层树可以看到Layer的状态, 表示层树则是动画正在表现的值的近似.

Core Animation的使用步骤:

1> 使用它需要先添加QuartzCore.framework框架和引入主头文件<QuartzCore/QuartzCore.h>(iOS7.0+ 不需要)

2>. 初始化一个CAAnimation对象，并设置一些动画相关属性

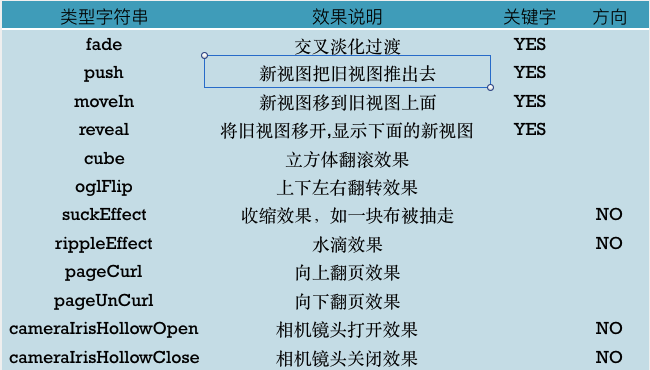
3>. 通过调用CALayer的addAnimation:forKey:方法增加CAAnimation对象到CALayer中，这样就能开始执行动画了

4>. 通过调用CALayer的removeAnimationForKey:方法可以停止CALayer中的动画

### 转场动画?

CATransition-转场动画, 作为CAAnimation的子类，用于做转场动画，能够为层提供移出屏幕和移入屏幕的动画效果。iOS比Mac OS X的转场动画效果少一点. UINavigationController就是通过CATransition实现了将控制器的视图推入屏幕的动画效果.

如下是转场动画的过渡效果:



使用UIView的动画函数, 实现转场动画

1> 单视图:

+(void)transitionWithView:(UIView\*)view

duration:(NSTimeInterval)duration options:(UIViewAnimationOptions)options animations:(void (^)(void))animations completion:(void (^)(BOOL finished))completion;

参数说明：

duration：动画的持续时间

view：需要进行转场动画的视图

options：转场动画的类型

animations：将改变视图属性的代码放在这个block中

completion：动画结束后，会自动调用这个block

2> 双视图:

+(void)transitionFromView:(UIView\*)fromView

toView:(UIView\*)toView

duration:(NSTimeInterval)duration options:(UIViewAnimationOptions)options completion:(void (^)(BOOL finished))completion;

参数说明：

duration：动画的持续时间

options：转场动画的类型

animations：将改变视图属性的代码放在这个block中

completion：动画结束后，会自动调用这个block

### 一个动画怎么实现?

以转场动画为例:

1> 创建CATransition对象

    CATransition \*animation = [CATransition animation];

2> 设置运动时间(即动画时间)

    animation.duration = DURATION;

 3> 设置运动type(类型)

    animation.type = type;

    if (subtype != nil) {

   4> 设置子类 (和type配合使用, 指定运动的方向)

        animation.subtype = subtype;

    }

    5> 设置运动速度(动画的运动轨迹，用于变化起点和终点之间的插值计算,形象点说它决定了动画运行的节奏,比如是均匀变化(相同时间变化量相同)还是先快后慢,先慢后快还是先慢再快再慢)

    animation.timingFunction = UIViewAnimationOptionCurveEaseInOut;

6> 将动画添加到view的Layer层

    [view.layer addAnimation:animation forKey:@"animation"];

动画类型如下:

typedef enum : NSUInteger {

    Fade = 1,                   //淡入淡出

    Push,                       //推挤

    Reveal,                     //揭开

    MoveIn,                     //覆盖

    Cube,                       //立方体

    SuckEffect,                 //吮吸

    OglFlip,                    //翻转

    RippleEffect,               //波纹

    PageCurl,                   //翻页

    PageUnCurl,                 //反翻页

    CameraIrisHollowOpen,       //开镜头

    CameraIrisHollowClose,      //关镜头

    CurlDown,                   //下翻页

    CurlUp,                     //上翻页

    FlipFromLeft,               //左翻转

    FlipFromRight,              //右翻转

} AnimationType;

### CADisplayLink

CADisplayLink是一种以屏幕刷新频率触发的时钟机制，每秒钟执行大约60次左右

CADisplayLink是一个计时器，可以使绘图代码与视图的刷新频率保持同步，而NSTimer无法确保计时器实际被触发的准确时间

使用方法：

定义CADisplayLink并制定触发调用方法

将显示链接添加到主运行循环队列

# 多线程（★★）

### 1、unix上进程怎么通信？

UNIX主要支持三种通信方式：

1. 基本通信：主要用来协调进程间的同步和互斥

(1)锁文件通信

通信的双方通过查找特定目录下特定类型的文件(称锁文件)来完成进程间 对临界资源访问时的互斥；例如进程p1访问一个临界资源，首先查看是否有一个特定类型文件，若有，则等待一段时间再查找锁文件。

(2)记录锁文件

2. 管道通信：适应大批量的数据传递

3. IPC    ：适应大批量的数据传递

### 2、列举几种进程的同步机制、进程的通信途径、死锁及死锁的处理方法。

答案：  原子操作 信号量机制    自旋锁    管程，会合，分布式系统

进程之间通信的途径

答案：共享存储系统消息传递系统管道：以文件系统为基础

进程死锁的原因

答案：资源竞争及进程推进顺序非法

死锁的4个必要条 件

答案：互斥、请求保持、不可剥夺、环路

死锁的处理

答案：鸵鸟策略、预防策略、避免策略、检测与解除死锁

### 3、谈谈你对多线程开发的理解？ios中有几种实现多线程的方法？

好处：

1.使用线程可以把占据时间长的[程序](http://baike.baidu.com/view/17674.htm" \t "_blank)中的[任务](http://baike.baidu.com/view/135914.htm" \t "_blank)放到[后台](http://baike.baidu.com/view/179243.htm" \t "_blank)去处理

2.用户界面可以更加吸引人，这样比如用户点击了一个按钮去触发某些事件的处理，可以弹出一个进度条来显示处理的进度

3.[程序](http://baike.baidu.com/view/17674.htm" \t "_blank)的运行效率可能提高

4·在一些等待的[任务](http://baike.baidu.com/view/135914.htm" \t "_blank)实现上如用户输入、文件读写和网络收发数据等，线程就比较有用了。

缺点：

1.如果有大量的线程,会影响性能,因为[操作系统](http://baike.baidu.com/view/880.htm" \t "_blank)需要在它们之间切换。

2.更多的线程需要更多的[内存](http://baike.baidu.com/view/1082.htm" \t "_blank)空间。

3.线程的中止需要考虑其对[程序](http://baike.baidu.com/view/17674.htm" \t "_blank)运行的影响。

4.通常块模型数据是在多个线程间共享的，需要防止线程死锁[情况](http://baike.baidu.com/view/780206.htm" \t "_blank)的发生。

实现多线程的方法：

NSObject类方法

NSThread

NSOperation

GCD

### 4、线程与进程的区别和联系?

线程是进程的基本单位 进程和线程都是由操作系统所体会的程序运行的基本单元,系统利用该基本单元实现系统对应用的并发性。 进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间,一个进程崩溃后,在保护模式下 不会对其它进程产生影响,而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量,但线程之间没有单独的地址空间,一个线程死掉就等于整个进程死掉,所以多进程的程序要比多线程的程序健壮,但在进程切换时,耗费资源较大,效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作,只能用线程,不能用进程。

### 5、简述多线程的作用以及什么地方会用到多线程? OC实现多线程的方法有哪些?

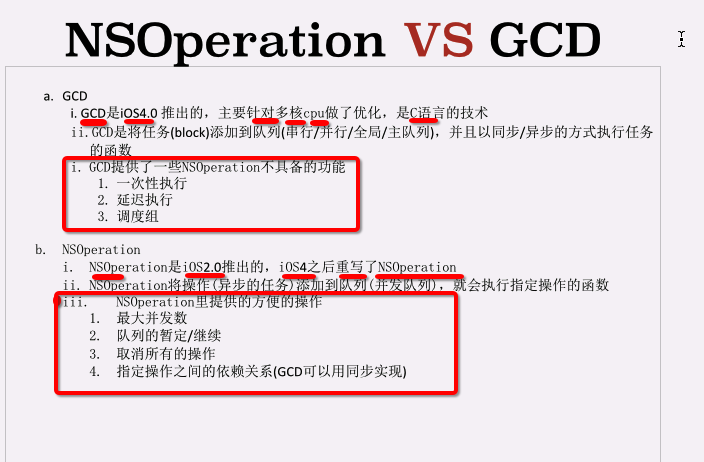
(1)、多线程的作用:可以解决负载均衡问题,充分利用cpu资源 。为了提高CPU的使用率,采用多线程的方式去同时完成几件事情而互不干扰,

(2)、大多情况下,要用到多线程的主要是需要处理大量的IO操作时或处理的情况需要花大量的时间等等,比如:读写文件、视频图像的采集、处理、显示、保存等。

(3)、iOS中多线程的方法主要有三种: 1、NSThread。2、NSOperation。3、GCD。

iOS中多线程有哪几种?各自之间有什么区别?这个问题,毋庸置疑,面试必问!!!

1).NSOperation与GCD的区别:



GCD与NSThread的区别:

1). NSThread 通过 @selector 指定要执行的方法，代码分散

2).GCD 通过 block 指定要执行的代码，代码集中, 所有的代码写在一起的，让代码更加简单，易于阅读和维护

3).使用 GCD 不需要管理线程的创建/销毁/复用的过程！程序员不用关心线程的生命周期

4).如果要开多个线程 NSThread 必须实例化多个线程对象

5).NSThread 靠 NSObject 的分类方法实现的线程间通讯，

### 6、谈谈多线程安全问题的几种解决方案?何为线程同步,如何实现的?分线程回调主线程方法是什么,有什么作用?

解决方案: 使用锁: 锁是线程编程同步工具的基础。锁可以让你很容易保护代码中一大块区域以便你可以确保代码的正确性。使用POSIX互斥锁; 使用NSLock类; 使用@synchronized指令等。

回到主线程的方法: dispatch\_async(dispatch\_get\_main\_queue(), ^{ });

作用: 主线程是显示UI界面,子线程多数是进行数据处理

### 7、详细谈谈GCD

推出的时间 iOS4 目的是用来取代NSThread（ios2.0推出）的，是 C语言框架，它能够自动利用更多CPU的核数，并且会自动管理线程的生命周期。

    CGD的两个核心概念：任务， 队列

     任务：记为在block中执行的代码。

     队列：用来存放任务的。

     注意事项：

    队列 != 线程!

    队列中存放的任务最后都要由线程来执行!

    队列的原则:先进先出,后进后出(FIFO/ First In First Out)

    队列又分为四种种：1 串行队列  2 并发队列 3 主队列 4 全局队列

    串行队列： 任务一个接一个的执行。

    并发队列： 队列中的任务并发执行。

    主队列： 跟主线程相关的队列，主队列里面的内容都会在主线程中执行（我们一般在主线程中刷新UI）。

    全局队列： 一个特殊的并发队列。

    并发队列与全局队列的区别：

    1. 并发队列有名称,可以跟踪错误。全局队列没有

    2. 在ARC中两个队列不需要考虑释放内存,但是在MRC中并发队列是创建出来的需要release操作，而全局队列只有一个不需要。

    3. 一般在开发过程中我们使用全局队列。

执行任务的两个函数

   '同步'执行任务:

  dispatch\_sync(<#dispatch\_queue\_t queue#>, <#^(void)block#>)

  '异步'执行任务:

  dispatch\_async(dispatch\_queue\_t queue, <#^(void)block#>)

  "同步"和"异步"的区别:

  "同步": 只能在'当前'线程中执行任务,不具备开启新线程的能力.

  "异步": 可以在'新'的线程中执行任务,具备开启新线程的能力.

 各个队列的执行效果：

 组合模式

       1. 串行队列同步执行，既在当前线程中顺序执行

       2. 串行队列异步执行，开辟一条新的线程，在该线程中顺序执行

       3. 并行队列同步执行，不开辟线程，在当前线程中顺序执行

       4. 并行队列异步执行，开辟多个新的线程，并且线程会重用，无序执行

       5. 主队列异步执行，不开辟新的线程，顺序执行

       6. 主队列同步执行，会造成死锁（'主线程'和'主队列'相互等待,卡住主线程）

线程间通讯：经典案例：子线程进行耗时操作（例如下载更新等）主线程进行UI刷新。

1.经典用法(子线程下载(耗时操作),主线程刷新UI):

    dispatch\_async(dispatch\_get\_global\_queue(0, 0), ^{

         // 执行耗时的异步操作...

  dispatch\_async(dispatch\_get\_main\_queue(), ^{

        // 回到主线程，执行UI刷新操作

             });

       });

    延迟操作

   <1> 调用 NSObject 方法:

    [self performSelector:@selector(run) withObject:nil afterDelay:2.0];

    // 2秒后再调用self的run方法

   <2> GCD 函数实现延时执行:

    dispatch\_after(dispatch\_time(DISPATCH\_TIME\_NOW, (int64\_t)(2.0 \* NSEC\_PER\_SEC)), dispatch\_get\_main\_queue(), ^{

  // 2秒后执行这里的代码... 在哪个线程执行，跟队列类型有关

    });

 5、队列组的使用：

       项目需求:

    首先:分别异步执行两个耗时操作;

    其次:等两次耗时操作都执行完毕后,再回到主线程执行操作.

    使用队列组(dispatch\_group\_t)快速,高效的实现上述需求.

    dispatch\_group\_t group = dispatch\_group\_create(); // 队列组

    dispatch\_queue\_t queue = dispatch\_get\_global\_queue(0, 0); // 全局并发队列

    dispatch\_group\_async(group, queue, ^{         // 异步执行操作1

        // longTime1

    });

    dispatch\_group\_async(group, queue, ^{         // 异步执行操作2

        // longTime2

  });

dispatch\_group\_notify(group, dispatch\_get\_main\_queue(), ^{

// 在主线程刷新数据

    // reload Data

  });

### 8、关于NSOperation的问题:

NSOperation: 抽象类,不能直接使用,需要使用其子类.(类似的类还有核心动画)

两个常用子类: NSInvocationOperation(调用) 和 NSBlockOperation(块);

两者没有本质区别,后者使用 Block 的形式组织代码,使用相对方便.

NSInvocationOperation在调用start方法后，不会开启新的线程只会在当前线程中执行。

NSBlockOperation 在调用start方法后，如果封装的操作数>1会开辟多条线程执行 =1 只会在当前线程执行.

NSOperationQueue 创建的操作队列默认为全局队列，队列中的操作执行顺序是无序的，如果需要让它有序执行需要添加依赖关系。

   // 操作op3依赖于操作op2;

   [op3 addDependency:op2];

    // 操作op2依赖于操作op1;

   [op2 addDependency:op1];

同时可以设置最大并发数

NSOperationQueue NSOperation支持 取消暂停的操作 但是正在进行的的操作并不能取消，这些操作一旦取消不可恢复.

### 9、为什么要取消/恢复队列呢？

一般在接收到内存警告后取消队列中的操作。

为了保证scorllView在滚动的时候流畅 通常在滚动开始时，暂停队列中的所有操作，滚动结束后，恢复操作。

### 10、说说NSOperation 和 GCD 的区别?

1. GCD是底层的C语言构成的API，而NSOperationQueue及相关对象是Objc的对象。在GCD中，在队列中执行的是由block构成的任务，这是一个轻量级的数据结构；而Operation作为一个对象，为我们提供了更多的选择；

    2. 在NSOperationQueue中，我们可以随时取消已经设定要准备执行的任务(当然，已经开始的任务就无法阻止了)，而GCD没法停止已经加入queue的block(其实是有的，但需要许多复杂的代码)；

    3. NSOperation能够方便地设置依赖关系，我们可以让一个Operation依赖于另一个Operation，这样的话尽管两个Operation处于同一个并行队列中，但前者会直到后者执行完毕后再执行；

    4. 我们能将KVO应用在NSOperation中，可以监听一个Operation是否完成或取消，这样子能比GCD更加有效地掌控我们执行的后台任务；

    5. 在NSOperation中，我们能够设置NSOperation的priority优先级，能够使同一个并行队列中的任务区分先后地执行，而在GCD中，我们只能区分不同任务队列的优先级，如果要区分block任务的优先级，也需要大量的复杂代码；

    6. 我们能够对NSOperation进行继承，在这之上添加成员变量与成员方法，提高整个代码的复用度，这比简单地将block任务排入执行队列更有自由度，能够在其之上添加更多自定制的功能。

7. GCD 是严格的队列，先进先出 FIFO；NSOperation可以改动 优先级（或者说服务质量）改变执行顺序

8. NSOperation的高级：最大并发数，控制线程个数，优化了线程的暂停、继续、取消功能（GCD实现起来太难，可以用 KVO ），依赖关系，可以让异步任务同步执行.

### 11、多线程的实现方式有哪些？都有什么区别？

### 12、多线程在开发中怎么用的？

### 13、关于RunLoop的问题:

### 14、RoonLoop与多线程的关系？

### 15、线程与进程的区别和联系?

线程是进程的基本单位 进程和线程都是由操作系统所体会的程序运行的基本单元,系统利用该基本单元实现系统对应用的并发性。 进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间,一个进程崩溃后,在保护模式下 不会对其它进程产生影响,而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量,但线程之间没有单独的 地址空间,一个线程死掉就等于整个进程死掉,所以多进程的程序要比多线程的程序健壮,但在进程切换时,耗费资源较 大,效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作,只能用线程,不能用进程。

### 16、简述多线程的作用以及什么地方会用到多线程?OC实现多线程的方法有哪些?谈谈多线程安全问题的几种解决方案?何为线程同步,如何实现的?分线程回调主线程方法是什么,有什么作用?

getter方法 -(NSString\*)name

{return \_name if(\_name!=name) }

(1)、多线程的作用:可以解决负载均衡问题,充分利用cpu资源 。为了提高CPU的使用率,采用多线程的方式去同时完 成几件事情而互不干扰,

(2)、大多情况下,要用到多线程的主要是需要处理大量的IO操作时或处理的情况需要花大量的时间等等,比如:读写文 件、视频图像的采集、处理、显示、保存等。

(3)、ios有三种主要方法:1、NSThread。2、NSOperation。3、GCD。

(4)解决方案:使用锁:锁是线程编程同步工具的基础。锁可以让你很容易保护代码中一大块区域以便你可以确保代码的正 确性。使用POSIX互斥锁;使用NSLock类;使用@synchronized指令等。

(5)回到主线程的方法: dispatch\_async(dispatch\_get\_main\_queue(), ^{ });

作用:主线程是显示UI界面,子线程多数是进行数据处理。

# 网络（★★）

### 1、HTTP的通信的 发送请求、接收响应 包含哪些内容？OC中是怎样实现的？

请求：

一个请求包含以下内容：

   请求行：包含了请求方法、请求资源路径、HTTP协议版本

GET /MJServer/resources/images/1.jpg HTTP/1.1

请求头：包含了对客户端的环境描述、客户端请求的主机地址等信息

Host: 192.168.1.105:8080   // 客户端想访问的服务器主机地址

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.9) Firefox/30.0

// 客户端的类型，客户端的软件环境

Accept: text/html, \*/\*   // 客户端所能接收的数据类型

Accept-Language: zh-cn   // 客户端的语言环境

Accept-Encoding: gzip   // 客户端支持的数据压缩格式

请求体：客户端发给服务器的具体数据，比如文件数据

OC中请求用NSURLRequest

NSURLRequest

    发送给服务器的请求包含：

    请求行： 包含了请求方法、请求资源路径、HTTP协议版本

           请求头：  对客户端的环境描述、客户端请求的主机地址等信息

           请求体：  客户端发给服务器的具体数据

           默认超时时常：60s

响应：

一个响应包括：

  状态行：包含了HTTP协议版本、状态码、状态英文名称

HTTP/1.1 200 OK

响应头：包含了对服务器的描述、对返回数据的描述

Server: Apache-Coyote/1.1   // 服务器的类型

Content-Type: image/jpeg   // 返回数据的类型

Content-Length: 56811   // 返回数据的长度

Date: Mon, 23 Jun 2014 12:54:52 GMT  // 响应的时间

实体内容：服务器返回给客户端的具体数据，比如文件数据

   OC中响应用NSURLRespose

          NSURLRespones

          返回给客户端的回应包含：

          状态行 ： 包含了HTTP协议版本、状态码、状态英文名称

          响应头 ： 包含了对服务器的描述、对返回数据的描述

          实体内容：服务器返回给客户端的具体二进制数据

          常用属性：  expectedContentLength （下载时返回文件的长度）

                        suggestedFilename（建议保存的文件名）

### 2、http 的post与get区别与联系，实践中如何选择它们？

（1）get是从服务器上获取数据，post是向服务器传送数据。

（2）在客户端，Get方式在通过URL提交数据，数据在URL中可以看到；POST方式，数据放置HTML HEADER内提交。

（3）对于get方式，服务器端用Request.QueryString获取变量的值，对于post方式，服务器端用Request.Form获取提交的数据。

（4）GET方式提交的数据最多只能有1024字节，而POST则没有此限制。

（5）安全性问题。正如在（1）中提到，使用 Get 的时候，参数会显示在地址栏上，而 Post 不会。所以，如果这些数据是中文数据而且是非敏感数据，那么使用 get；如果用户输入的数据不是中文字符而且包含敏感数据，那么还是使用 post为好。 Http定义了与服务器交互的不同方法，最基本的方法有4种，分别是GET，POST，PUT，DELETE。URL全称是资源描述符，我们可以这样认为：一个URL地址，它用于描述一个网络上的资源，而HTTP中的GET，POST，PUT，DELETE就对应着对这个资源的查，改，增，删4个操作。GET一般用于获取/查询资源信息，而POST一般用于更新资源信息。

### 3、说说关于UDP/TCP的区别？

UDP（用户数据报协议）

只管发送，不确认对方是否接收到

将数据及源和目的封装成数据包中，不需要建立连接

每个数据报的大小限制在64K之内

因为无需连接，因此是不可靠协议

不需要建立连接，速度快

应用场景：多媒体教室／网络流媒体

TCP（传输控制协议）

建立连接，形成传输数据的通道

在连接中进行大数据传输（数据大小不收限制）

通过三次握手完成连接，是可靠协议，安全送达

必须建立连接，效率会稍低

### 4、什么是三次握手与四次握手？

三次握手实现的过程：

第一次握手：建立连接时，客户端发送同步序列编号到服务器，并进入发送状态，等待服务器确认

第二次：服务器收到同步序列编号，并确认同时自己也发送一个同步序列编号+确认标志，此时服务器进入接收状态

第三次：客户端收到服务器发送的包，并向服务器发送确认标志，随后链接成功。

注意：是在链接成功后在进行数据传输。

四次握手：

第一次： 客户端向服务器发送一个带有结束标记的报文。

第二次： 服务器收到报文后，向客户端发送一个确认序号，同时通知自己相应的应用程序：对方要求关闭连接第三次： 服务器向客户端发送一个带有结束标记的报文。

第四次： 客户端收到报文后，向服务器发送一个确认序号。链接关闭。

### 5、分析json、xml的区别?json、xml解析方式的底层是如何处理的?

Json与xml的区别:

(1)可读性方面:基本相同,xml的可读性比较好

(2)可扩展性方面:都具有很好的扩展性

(3)编码难度方面:相对而言:JSON的编码比较容易

(4)解码难度:json的解码难度基本为零,xml需要考虑子节点和父节点

(5)数据体积方面:json相对于xml来讲,数据体积小,传递的速度跟快些

(6)数据交互方面:json与JavaScript的交互更加方面,更容易解析处理,更好的数据交互

(7)数据描述方面:xml对数据描述性比较好

(8)传输速度方面:json的速度远远快于xml

JSON底层原理:

遍历字符串中的字符,最终根据格式规定的特殊字符,比如{}号,[]号, : 号 等进行区分,{}号是一个字典 的开始,[]号是一个数组的开始, : 号是字典的键和值的分水岭,最终乃是将json数据转化为字典,字典中值可能是字典,数组,或字符串而已。

XML底层原理:

XML解析常用的解析方法有两种:DOM解析和SAX解析。DOM 采用建立树形结构的方式访问 XML 文档,而 SAX 采用的事件模型。 DOM 解析把 XML 文档转化为一个包含其内容的树,并可以对树进行遍历。使用 DOM 解析器的时候需 要处理整个 XML 文档,所以对性能和内存的要求比较高。SAX在解析 XML 文档的时候可以触发一系列的事件,当发现给定的tag 的时候,它可以激活一个回调方法,告诉该方法制定的标签已经找到。SAX 对内存的要求通常会比较低,因为它让开发人员自己来决 定所要处理的tag。特别是当开发人员只需要处理文档中所包含的部分数据时,SAX 这种扩展能力得到了更好的体现。

延伸:SAX与DOM的区别及应用场景:

1> SAX处理的优点非常类似于流媒体的优点。分析能够立即开始,而不是等待所有的数据被处理。而且由于应用程序只是 在读取数据时检查数据,因此不需要将数据存储在内存中。这对于大型文档来说是个巨大的优点。事实上,应用程序甚至不 必解析整个文档;它可以在某个条件得到 满足时停止解析。一般来说,SAX 还比它的替代者 DOM 快许多。另一方面,由于应用程序没有以任何方式存储数据,使用 SAX 来更改数据或在数据流中往后移是不可能的。

 2> DOM 以及广义的基于树的处理具有几个优点。首先,由于树在内存中是持久的,因此可以修改它以便应用程序能对数据和结构作出更改。它还可以在任何时候在树中上下 导航,而不是像 SAX 那样是一次性的处理。DOM 使用起来也要简单 得多。另一方面,在内存中构造这样的树涉及大量的开销。大型文件完全占用系统内存容量的情况并不鲜见。此外,创建一 棵 DOM 树可能是一个缓慢的过程。

3> 选择 DOM 还是选择 SAX,这取决于下面几个因素: 应用程序的目的:如果打算对数据作出更改并将它输出为 XML,那么在大多数情况下,DOM 是适当的选择。并不是说使 用 SAX 就不能更改数据,但是该过程要复杂得多,因为您必须对数据的一份拷贝而不是对数据本身作出更改。 数据容量: 对于大型文件,SAX 是更好的选择。数据将如何使用:如果只有数据中的少量部分会被使用,那么使用 SAX 来将该部分数据提取到应用程序中可能更好。 另一方面,如果您知道自己以后会回头引用已处理过的大量信息,那么 SAX 也许不是恰当的选择。 对速度的需要:SAX 实现通常要比 DOM 实现更快。 SAX 和 DOM 不是相互排斥的,记住这点很重要。您可以使用 DOM 来创建 SAX 事件流,也可以使用 SAX 来创建 DOM 树。事实上,用于创建 DOM 树的大多数解析器实际上都使用 SAX 来完成这个任务!

### 6、http和scoket通信的区别?socket连接相关库,TCP,UDP的连接方法,HTTP的几种常用方式?

面试常问, 是否了解Socket网络编程?是否明白TCP/IP HTTP UDP等协议知识?

http和scoket通信的区别:

http是客户端用http协议进行请求,发送请求时候需要封装http请求头,并绑定请求的数据,服务器一般有web服务器配合(当然也非绝对)。 http请求方式为客户端主动发起请求,服务器才能给响应,一次请求完毕后则断开连接,以节省资源。服务器不能主动给客户端响应(除非采取http长连接技术)。iphone主要使用类是NSUrlConnection。

scoket是客户端跟服务器直接使用socket“套接字”进行连接,并没有规定连接后断开,所以客户端和服务器可以保持连接通道,双方都可以主动发送数据。一般在游戏开发或股票开发这种要求即时性很强并且保持发送数据量比较大的场合使用。主要使用类是CFSocketRef。

UDP: 是用户数据报协议: 主要用在实时性要求高以及对质量相对较弱的地方,但面对现在高质量的线路不是容易丢包除非是一些拥塞条件下, 如流媒体

TCP: 是传输控制协议:是面连接的,那么运行环境必然要求其可靠性不可丢包有良好的拥塞控制机制如http ftp telnet 等

http的常用方式:get,post

# 9设计模式（★★）

### 1. 什么是设计模式

设计模式是为特定场景下的问题而定制的解决方案。特定场景指问题所在的重复出现的场景，问题指特定环境下你想达成的目标。同样的问题在不同的环境下会有不同的限制和挑战。定制的解决方案是指在特定环境下克服了问题的限制条件而达成目标的一种设计。

### 2. 设计模式的分类

设计模式分为三种类型，共23种。

创建型模式：单例模式、[抽象工厂模式](http://baike.baidu.com/view/1580269.htm)、建造者模式、工厂模式、原型模式。

结构型模式：适配器模式、桥接模式、装饰模式、组合模式、外观模式、[享元模式](http://baike.baidu.com/view/6143765.htm)、代理模式。

行为型模式：模版方法模式、命令模式、[迭代器模式](http://baike.baidu.com/view/9791023.htm)、[观察者模式](http://baike.baidu.com/view/1854779.htm)、中介者模式、备忘录模式、解释器模式（[Interpreter模式](http://baike.baidu.com/view/3928378.htm)）、状态模式、策略模式、职责链模式(责任链模式)、访问者模式。

### 3. 类工厂方法是什么？

 类工厂方法的实现是为了向客户提供方便，它们将分配和初始化合在一个步骤中， 返回被创建的对象，并 进行自动释放处理。这些方法的形式是+ (type)className...（其中 className不包括任何前缀）。

工厂方法可能不仅仅为了方便使用。它们不但可以将分配和初始化合在一起，还可以 为初始化过程提供对 象的分配信息。

类工厂方法的另一个目的是使类（比如NSWorkspace）提供单件实例。虽 然init...方法可以确认一 个类在每次程序运行过程只存在一个实例，但它需要首先分配一个“生的”实例，然后还必须释放该实例。 工厂 方法则可以避免为可能没有用的对象盲目分配内存。

### 4. 单件实例是什么？

Foundation 和 Application Kit 框架中的一些类只允许创建单件对象，即这些类在当前进程中的唯一实例。举例来说，NSFileManager 和NSWorkspace 类在使用时都是基于进程进行单件对象的实例化。当向这些类请求实例的时候，它们会向您传递单一实例的一个引用，如果该实例还不存在，则首先进行实例的分配 和初始化。 单件对象充当控制中心的角色，负责指引或协调类的各种服务。如果类在概念上只有一个实例（比如 NSWorkspace），就应该产生 一个单件实例，而不是多个实例；如果将来某一天可能有多个实例，您可 以使用单件实例机制，而不是工厂方法或函数。

### 5. 动态绑定—在运行时确定要调用的方法

动态绑定将调用方法的确定也推迟到运行时。在编译时，方法的调用并不和代码绑定 在一起，只有在消实发送出来之后，才确定被调用的代码。通过动态类型和动态绑定技术，您的代码每次执行都可以得到不同的结果。运行时因子负责确定消息的接 收者和被调用的方法。 运行时的消息分发机制为动态绑定提供支持。当您向一个动态类型确定了的对象发送消息时，运行环境系统会通过接收者的isa指针定位对象的类，并以此为起点 确定被调用的方法，方法和消息是动态绑定的。而且，您不必在Objective-C 代码中做任何工作，就可以自动获取动态绑定的好处。您在每次发送消息时，

 特别是当消息的接收者是动态类型已经确定的对象时，动态绑定就会例行而 透明地发生。

### 6.谈谈你对MVC的理解?为什么要用MVC?在Cocoa中MVC是怎么实现的?你还熟悉其他的OC设计模式或别的设计模式吗?

MVC就是Model-View-Controller的缩写,M指的是业务模型,V指的是用户页面,C指的是控制器。MVC是架构模式,是讲M和 V的代码分离,从而使同那个一个程序可以使用不同的表现形式。 M:表示数据和业务规则,V是用户看到的并与之交互的页面,C是接受用户的输入并调用M和V取完成用户需求的

单例,代理,观察者,工厂模式等 单例模式:说白了就是一个类不通过alloc方式创建对象,而是用一个静态方法返回这个类的对象。系统只需要拥有一个的 全局对象,这样有利于我们协调系统整体的行为; 代理模式:代理模式给某一个对象提供一个代理对象,并由代理对象控制对源对象的引用.比如一个工厂生产了产品,并不想 直接卖给用户,而是搞了很多代理商,用户可以直接找代理商买东西,代理商从工厂进货.常见的如QQ的自动回复就属于代 理拦截,代理模式在iphone中得到广泛应用.

### 7.说一说代理模式是什么？

### 8.说一说单例模式是什么？

（一次性）单例的使用：

   1.单例简介:

    作用:

        保证程序在运行过程中,一个类只有一个实例对象.这个实例对象容易被外界访问!

        控制实例对象个数(只有一个),节约系统资源.

    使用场合:

        在整个应用程序中,共享一份资源(这份资源只需要创建初始化一次).

    举例:

        打印机/视图窗口/一些网络工具类等等

单例分为两种：懒汉式（用到的时候在加载）、恶汉式（程序已启动就加载）

单例实现的方法：互斥锁    dispatch\_once

单例需要重写的方法：

模仿苹果源生的写法创建单例单例的方法名都是share+xxxx的一个形式，这也是大家在使用单利时默认的方法但是为了防止单例被用alloc init的方法创建出来需要重写

+ (id)allocWithZone:(struct \_NSZone \*)zone（该方法在调用alloc init方法时会自定调用，保证单例只开辟一个空间）

同时为了防止使用copy创建单例需要重写

+(id)copyWithZone:(struct \_NSZone \*)zone

最后提供一个外界方法创建唯一的实利

+(instance)share+xxxx;

MRC 中增加如下方法的实现:

    - (instancetype)retain { return self; }

    - (NSUInteger)retainCount { return 1; }

    - (oneway void)release {}

    - (instancetype)autorelease { return self; }

### 9.说一说观察者模式是什么？

观察者模式: 当一个物体发生变化时,会通知所有观察这个物体的观察者让其做出反应。实现起来无非就是把所有观察者的 对象给这个物体,当这个物体的发生改变,就会调用遍历所有观察者的对象调用观察者的方法从而达到通知观察者的目的;

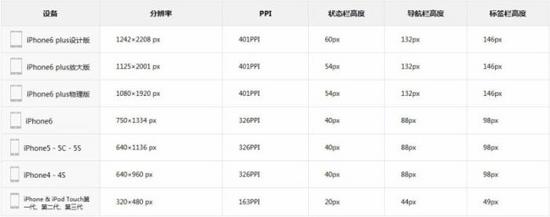
### 10.你了解MVVM设计模式吗？

2.1. 你是怎么理解 MVVM的？MVVM 中的 View 和 ViewModel 之间是怎么个关系？

# 10屏幕适配（★★）

### 1. 常使用的适配方式是什么？平时是怎样实现适配的？

关于iOS尺寸规范如下:



根据UI设计师，标注“距屏幕边缘xx像素，距顶部xx像素，字号多少”就可以了，然后交给我们，如果使用XIB或者Storyboard 进行autolayout会自动拉伸布局, 如果纯代码进行自动布局, 可以使用苹果原生的自动布局, 可视化布局 VFL. 也可以使用第三方框架, 例如Snapkit, masonry.

### 2. 说说控件适配和屏幕适配?(以上四个开放题,下面的回答均可参考一二)

根据屏幕类型判断, 可以根据设备类型，写一些if...else，或者switch语句, 判断机型可以使用screen的height（不能使用width，因为4和5的width是一样的，都是320），也可以使用API里的宏，都差不多。我个人感觉，if...else似乎是不可避免的，虽然有auto layout，但是有一些大的布局改动，或者字体大小，不用判断似乎是无法解决的, 比如说，为了达到最佳显示效果，我们在大的屏幕上使用CollectionView，而在4S上使用TableView，用自动布局应该是没有办法做到的。或者根据屏幕的大小，切换字体大小，好像也只能通过if...else来实现

Masonry是我们实现屏幕适配的重要手段之一，本质上是界面约束的语法糖。基本上，我们的做法是：大的页面关系，用计算完成；每个小块里面的相对位置关系，用Masonry来做。在有些场景下，Masonry有非常大的优势。比如说: 垂直方向与另一个View对齐，左边距离上一个元素的右边5，右边距离父View右边5, 类似这种布局，用frame来写会复杂很多，如果再考虑屏幕适配，需要非常多代码。这类的需求，Masonry堪称神器。不过使用中发现，用Masonry布局的View，我们通常会init，或者initWithFrame:CGRectZero。这个View直到经过Masonry处理以后，它的origin和size才能确定，如果在此之前就用到它的origin和size，就会有问题, 整体替换UIView, 对于适配后变化不大的页面，把if...else写在UIView里，但是有个别页面，完全要根据设备显示不同的View。这种情况比较适合在Controller里做判断，然后load不同的View

一定要注意问公司的代码规范.

### iOS 9的适配?

1>.tableview有时全部刷新无效。必须采用局部刷新；

2>.不允许在didFinishLaunchingWithOptions之后还没有设置window的rootViewController

3>. 直接使用[UIApplication shareApplication]设置StatusBarStyle 会出现警告。必须在控制器里通过

-(UIStatusBarStyle)preferredStatusBarStylei{

return UIStatusBarStyleLightContent; }

4>. UIR scheme 也就是应用跳转的时候必须在info.plist中添加, LSApplicationQueriesSchemes   Array添加要跳转的的scheme

5>. 字体改变。建议固定长度的文字也使用sizetofit或者提起计算

6>. 企业级应用的时候需要手动到   设置－> 通用－> 描述文件里面自行添加信任。

类似与安装不是苹果官方下载的软件。要手动信任。

7>. Bitcode(就是允许苹果对我门的软件进行瘦身)，编译报错的原因是如果我门添加了默写第三方的库不支持bitcode.

8>. 常用的http与https。需要手动的添加信任的关系。

### 如何进行iOS6、7的适配?

通过判断版本来控制，来执行响应的代码

功能适配：保证同一个功能在6、7上都能用

UI适配：保证各自的显示风格

// iOS版本为7.0以上（包含7.0）

#define iOS7 ([[UIDevice currentDevice].systemVersion doubleValue]>=7.0)

### 5. iOS 9的新特性?

1> .Xcode 7 免证书真机调试。教程：<http://www.cnblogs.com/wahy/p/5004623.html>

  2>.如果不适配iOS9，就不能偷偷在后台定位（不带蓝条，见图）！好消息：将允许出现这种场景：同一App中的多个location manager：一些只能在前台定位，另一些可在后台定位，并可随时开启或者关闭特定location manager的后台定位。如果没有请求后台定位的权限，也是可以在后台定位的，不过会带蓝条。

  3>.UIAlertView过期。用UIAlertController 可以代替其功能。

  4>.tableView刷新不出来问题。虽然现在的iOS9已经推送正式版了，但是iOS9使用时还是会感觉到App比以前更加卡顿了，tableView拖动时卡顿显示的最为明显。 并且之前遇到一个bug，原本好的项目用xcode7一编译，tableView刷新出了问题 ，[tableView reloadData]无效 有一行cell明明改变了但是刷新不出来。 感觉可能是这个方法和某种新加的特性冲突了，猜测可能是reloadData的操作被推迟到下一个RunLoop执行最终失效。

解决的方法是，注释[tableView reloadData]，改用局部刷新，问题居然就解决了。

# 11数据存储（★★）

### IOS中常用的数据存储方式有哪些？

答：1.数据存储有四种方案，NSUserDefault,KeyChain,File,DB.

2.其中File有三种方式：plist,Archiver,Stream

3.DB包括core Data和FMDB

### 2、说一说你对sqlite的认识

SQLite是目前主流的嵌入式关系型数据库，其最主要的特点就是轻量级、跨平台，当前很多嵌入式操作系统都将其作为数据库首选。虽然SQLite是一款轻型数据库，但是其功能也绝不亚于很多大型关系数据库。学习数据库就要学习其相关的定义、操作、查询语言，也就是大家日常说得SQL语句。和其他数据库相比，SQLite中的SQL语法并没有太大的差别，因此这里对于SQL语句的内容不会过多赘述，大家可以参考SQLite中其他SQL相关的内容，这里还是重点讲解iOS中如何使用SQLite构建应用程序。先看一下SQLite数据库的几个特点：

1.基于C语言开发的轻型数据库

2.在iOS中需要使用C语言语法进行数据库操作、访问（无法使用ObjC直接访问，因为libqlite3框架基于C语言编写）

3.SQLite中采用的是动态数据类型，即使创建时定义了一种类型，在实际操作时也可以存储其他类型，但是推荐建库时使用合适的类型（特别是应用需要考虑跨平台的情况时）

4.建立连接后通常不需要关闭连接（尽管可以手动关闭）

在iOS中操作SQLite数据库可以分为以下几步（注意先在项目中导入libsqlite3框架）：

1.打开数据库，利用sqlite3\_open()打开数据库会指定一个数据库文件保存路径，如果文件存在则直接打开，否则创建并打开。打开数据库会得到一个sqlite3类型的对象，后面需要借助这个对象进行其他操作。

2.执行SQL语句，执行SQL语句又包括有返回值的语句和无返回值语句。

3.对于无返回值的语句（如增加、删除、修改等）直接通过sqlite3\_exec()函数执行；

4.对于有返回值的语句则首先通过sqlite3\_prepare\_v2()进行sql语句评估（语法检测），然后通过sqlite3\_step()依次取出查询结果的每一行数据，对于每行数据都可以通过对应的sqlite3\_column\_类型()方法获得对应列的数据，如此反复循环直到遍历完成。当然，最后需要释放句柄。

### 3、说一说你对FMDB的认识

FMDB是一个处理数据存储的第三方框架，框架是对sqlite的封装，整个框架非常轻量级但又不失灵活性，而且更加面向对象。FMDB有如下几个特性：

1.FMDB既然是对于libsqlite3框架的封装，自然使用起来也是类似的，使用前也要打开一个数据库，这个数据库文件存在则直接打开否则会创建并打开。这里FMDB引入了一个MFDatabase对象来表示数据库，打开数据库和后面的数据库操作全部依赖此对象。

2.对于数据库的操作跟前面KCDbManager的封装是类似的，在FMDB中FMDatabase类提供了两个方法executeUpdate:和executeQuery:分别用于执行无返回结果的查询和有返回结果的查询。当然这两个方法有很多的重载这里就不详细解释了。唯一需要指出的是，如果调用有格式化参数的sql语句时，格式化符号使用“?”而不是“%@”、等。

3.我们知道直接使用libsqlite3进行数据库操作其实是线程不安全的，如果遇到多个线程同时操作一个表的时候可能会发生意想不到的结果。为了解决这个问题建议在多线程中使用FMDatabaseQueue对象，相比FMDatabase而言，它是线程安全的。

4.将事务放到FMDB中去说并不是因为只有FMDB才支持事务，而是因为FMDB将其封装成了几个方法来调用，不用自己写对应的sql而已。其实在在使用libsqlite3操作数据库时也是原生支持事务的（因为这里的事务是基于数据库的，FMDB还是使用的SQLite数据库），只要在执行sql语句前加上“begin transaction;”执行完之后执行“commit transaction;”或者“rollback transaction;”进行提交或回滚即可。另外在Core Data中大家也可以发现，所有的增、删、改操作之后必须调用上下文的保存方法，其实本身就提供了事务的支持，只要不调用保存方法，之前所有的操作是不会提交的。在FMDB中FMDatabase有beginTransaction、commit、rollback三个方法进行开启事务、提交事务和回滚事务。

### 4、说一说你对Core Data的认识

Core Data使用起来相对直接使用SQLite3的API而言更加的面向对象，操作过程通常分为以下几个步骤：

1.创建管理上下文

创建管理上下可以细分为：加载模型文件->指定数据存储路径->创建对应数据类型的存储->创建管理对象上下方并指定存储。

经过这几个步骤之后可以得到管理对象上下文NSManagedObjectContext，以后所有的数据操作都由此对象负责。同时如果是第一次创建上下文，Core Data会自动创建存储文件（例如这里使用SQLite3存储），并且根据模型对象创建对应的表结构。

2.查询数据

对于有条件的查询，在Core Data中是通过谓词来实现的。首先创建一个请求，然后设置请求条件，最后调用上下文执行请求的方法。

3.插入数据

插入数据需要调用实体描述对象NSEntityDescription返回一个实体对象，然后设置对象属性，最后保存当前上下文即可。这里需要注意，增、删、改操作完最后必须调用管理对象上下文的保存方法，否则操作不会执行。

4.删除数据

删除数据可以直接调用管理对象上下文的deleteObject方法，删除完保存上下文即可。注意，删除数据前必须先查询到对应对象。

5.修改数据

修改数据首先也是取出对应的实体对象，然后通过修改对象的属性，最后保存上下文。

### 5、OC中有哪些数据存储方式,各有什么区别?

OC中有四种数据存储方式:

1).NSUserDefaults,用于存储配置信息

2).SQLite,用于存储查询需求较多的数据

3).CoreData,用于规划应用中的对象

4).使用基本对象类型定制的个性化缓存方案. NSUserDefaults:对象中储存了系统中用户的配置信息,开发者可以通过这个实例对象对这些已有的信息进行修改,也 可以按照自己的需求创建新的配置项。 SQLite擅长处理的数据类型其实与NSUserDefaults差不多,也是基础类型的小数据,只是从组织形式上不同。开发者可 以以关系型数据库的方式组织数据,使用SQL DML来管理数据。一般来说应用中的格式化的文本类数据可以存放在数据库 中,尤其是类似聊天记录、Timeline等这些具有条件查询和排序需求的数据。 CoreData是一个管理方案,它的持久化可以通过SQLite、XML或二进制文件储存。它可以把整个应用中的对象建模并进 行自动化的管理。从归档文件还原模型时CoreData并不是一次性把整个模型中的所有数据都载入内存,而是根据运行时状 态,把被调用到的对象实例载入内存。框架会自动控制这个过程,从而达到控制内存消耗,避免浪费。 无论从设计原理还是使用方法上看,CoreData都比较复杂。因此,如果仅仅是考虑缓存数据这个需求,CoreData绝对不 是一个优选方案。CoreData的使用场景在于:整个应用使用CoreData规划,把应用内的数据通过CoreData建模,完全 基于CoreData架构应用。 使用基本对象类型定制的个性化缓存方案:从需求出发分析缓存数据有哪些要求:按Key查找,快速读取,写入不影响正常 操作,不浪费内存,支持归档。这些都是基本需求,那么再进一步或许还需要固定缓存项数量,支持队列缓存,缓存过期等。

数据存储这一块, 面试常问, 你常用哪一种数据存储? 什么是序列化? sqlite是直接用它还是用封装了它的第三方库? 尤其是会问sqlite和core data的区别?

### 5、IOS平台怎么做数据的持久化?coredata和sqlite有无必然联系?coredata是一个关系型数据库吗?

iOS中可以有四种持久化数据的方式: 属性列表、对象归档、SQLite3和Core Data coredata可以使你以图形界面的方式快速的定义app的数据模型,同时在你的代码中容易获取到它。coredata提供了基础结构去处 理常用的功能,例如保存,恢复,撤销和重做,允许你在app中继续创建新的任务。在使用coredata的时候,你不用安装额外的数据 库系统,因为coredata使用内置的sqlite数据库。coredata将你app的模型层放入到一组定义在内存中的数据对象。coredata会 追踪这些对象的改变,同时可以根据需要做相应的改变,例如用户执行撤销命令。当coredata在对你app数据的改变进行保存的时 候,core data会把这些数据归档,并永久性保存。 mac os x中sqlite库,它是一个轻量级功能强大的关系数据引擎,也很容易嵌入到应用程序。可以在多个平台使用,sqlite是一个轻 量级的嵌入式sql数据库编程。与coredata框架不同的是,sqlite是使用程序式的,sql的主要的API来直接操作数据表。 Core Data不是一个关系型数据库,也不是关系型数据库管理系统(RDBMS)。虽然Core Dta支持SQLite作为一种存储类型, 但它不能使用任意的SQLite数据库。Core Data在使用的过程种自己创建这个数据库。Core Data支持对一、对多的关系。

### 6、如果后期需要增加数据库中的字段怎么实现，如果不使用CoreData呢？

编写SQL语句来操作原来表中的字段

* 增加表字段

ALTER TABLE 表名 ADD COLUMN 字段名 字段类型;

* 删除表字段

ALTER TABLE 表名 DROP COLUMN 字段名;

. 修改表字段

ALTER TABLE 表名 RENAME COLUMN 旧字段名 TO 新字段名;

### 7、SQLite数据存储是怎么用？

添加SQLite动态库：

导入主头文件：#import <sqlite3.h>

利用C语言函数创建\打开数据库，编写SQL语句

### 8、简单描述下客户端的缓存机制？

1>缓存可以分为：内存数据缓存、数据库缓存、文件缓存

2>每次想获取数据的时候

3>先检测内存中有无缓存

4>再检测本地有无缓存(数据库\文件)

5>最终发送网络请求

6>将服务器返回的网络数据进行缓存（内存、数据库、文件）， 以便下次读取

### 9、你实现过多线程的Core Data么？NSPersistentStoreCoordinator，NSManagedObjectContext和NSManagedObject中的哪些需要在线程中创建或者传递？你是用什么样的策略来实现的？

1>CoreData是对SQLite数据库的封装

2>CoreData中的NSManagedObjectContext在多线程中不安全

### 3>如果想要多线程访问CoreData的话，最好的方法是一个线程一个NSManagedObjectContext

4>每个NSManagedObjectContext对象实例都可以使用同一个NSPersistentStoreCoordinator实例，这是因为NSManagedObjectContext会在便用NSPersistentStoreCoordinator前上锁

### 10、core  data数据迁移

博客地址: <http://blog.csdn.net/jasonblog/article/details/17842535>

### 11、FMDB的使用和对多张表的处理

博客地址: <http://blog.csdn.net/wscqqlucy/article/details/8464398>

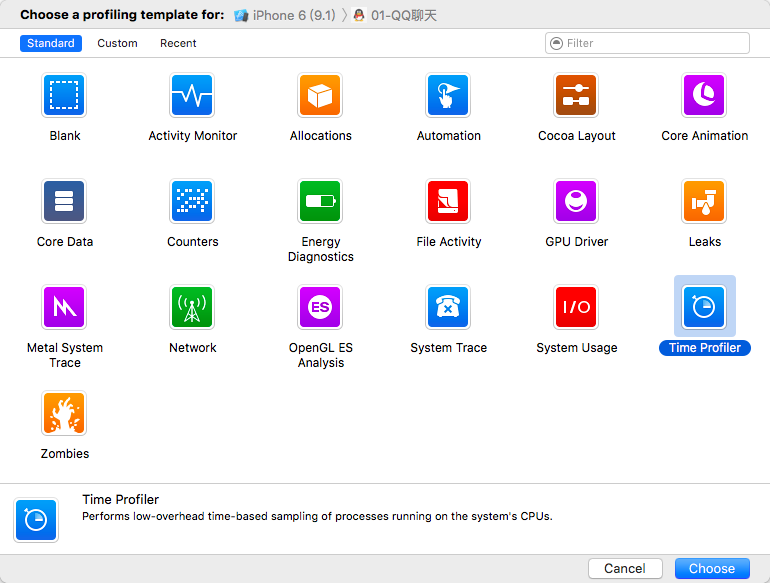
### 12、说说数据库的左连接和右连接的区别

数据库左连接和右连接的区别：主表不一样通过左连接和右连接，最小条数为3（记录条数较小的记录数），最大条数为12（3×4）技术博客的地址 : <http://www.2cto.com/database/201407/317367.html>

# 12 iOS性能优化（★★）

### 1. 平时你是如何对代码进行性能优化的？

利用性能分析工具检测，包括静态 Analyze 工具，以及运行时 Profile 工具，通过Xcode工具栏中Product->Profile可以启动,启动后界面如下:



比如测试程序启动运行时间，当点击Time Profiler应用程序开始运行后.就能获取到整个应用程序运行消耗时间分布和百分比.为了保证数据分析在统一使用场景真实需要注意一定要使用真机,因为此时模拟器是运行在Mac上，而Mac上的CPU往往比iOS设备要快。

为了防止一个应用占用过多的系统资源，开发iOS的苹果工程师门设计了一个“看门狗”的机制。在不同的场景下，“看门狗”会监测应用的性能。如果超出了该场景所规定的运行时间，“看门狗”就会强制终结这个应用的进程。开发者们在crashlog里面，会看到诸如0x8badf00d这样的错误代码。

### 2. 优化Table View

为了保证table view平滑滚动，确保你采取了以下的措施:

* 正确使用`reuseIdentifier`来重用cells
* 尽量使所有的view opaque，包括cell自身
* 如果cell内现实的内容来自web，使用异步加载，缓存请求结果
* 减少subviews的数量
* 尽量不适用`cellForRowAtIndexPath:`，如果你需要用到它，只用一次然后缓存结果
* 使用`rowHeight`, `sectionFooterHeight` 和 `sectionHeaderHeight`来设定固定的高，不要请求delegate

### 3. UIImage加载图片

A：imagedNamed初始化

B：imageWithContentsOfFile初始化

imageNamed默认加载图片成功后会内存中缓存图片,这个方法用一个指定的名字在系统缓存中查找并返回一个图片对象.如果缓存中没有找到相应的图片对象,则从指定地方加载图片然后缓存对象，并返回这个图片对象.

而imageWithContentsOfFile则仅只加载图片,不缓存.

大量使用imageNamed方式会在不需要缓存的地方额外增加开销CPU的时间来做这件事.当应用程序需要加载一张比较大的图片并且使用一次性，那么其实是没有必要去缓存这个图片的，用imageWithContentsOfFile是最为经济的方式,这样不会因为UIImage元素较多情况下，CPU会被逐个分散在不必要缓存上浪费过多时间.

使用场景需要编程时，应该根据实际应用场景加以区分，UIimage虽小，但使用元素较多问题会有所凸显.

1. 不要在 viewWillAppear 中做费时的操作

viewWillAppear: 在 view 显示之前被调用，出于效率考虑，在这个方法中不要处理复杂费时的事情；只应该在这个方法设置 view 的显示属性之类的简单事情，比如背景色，字体等。要不然，用户会明显感觉到 view 显示迟钝。

1. 在正确的地方使用reuseIdentifier

table view用 `tableView:cellForRowAtIndexPath:` 为rows分配cells的时候，它的数据应该重用自UITableViewCell。 一个table view维持一个队列的数据可重用的UITableViewCell对象。

1. 尽量把views设置为透明

如果你有透明的Views你应该设置它们的opaque属性为YES。原因是这会使系统用一个最优的方式渲染这些views。这个简单的属性在IB或者代码里都可以设定。

1. 避免过于庞大的XIB

尽量简单的为每个Controller配置一个单独的XIB，尽可能把一个View Controller的view层次结构分散到单独的XIB中去, 当你加载一个引用了图片或者声音资源的nib时，nib加载代码会把图片和声音文件写进内存。在OS X中，图片和声音资源被缓存在named cache中以便将来用到时获取

1. 不要阻塞主线程

永远不要使主线程承担过多。因为UIKit在主线程上做所有工作，渲染，管理触摸反应，回应输入等都需要在它上面完成,大部分阻碍主进程的情形是你的app在做一些牵涉到读写外部资源的I/O操作，比如存储或者网络。

1. dispatch\_async(dispatch\_get\_global\_queue(DISPATCH\_QUEUE\_PRIORITY\_DEFAULT, 0), ^{
2. // 选择一个子线程来执行耗时操作
3. dispatch\_async(dispatch\_get\_main\_queue(), ^{
4. // 返回主线程更新UI
5. });
6. });
7. 在Image Views中调整图片大小

如果要在`UIImageView`中显示一个来自bundle的图片，你应保证图片的大小和UIImageView的大小相同。在运行中缩放图片是很耗费资源的 .

# 13第三方框架（★★）

面试常问,你经常使用一些第三方库吗?比如AFNetworking SDWebimage JsonModel等,有没有去了解过里面的实现原理?

## 第三方框架

### 1. 关于cocoaPods的问题:

#### 1.2. 使用过 CocoaPods 吗？它是什么？ CocoaPods的原理?

CocoaPod是一个第三方库的管理工具，用来管理项目中的第三方框架。

在终端中进入（cd命令）你项目所在目录，然后在当前目录下，利用vim创建Podfile，运行： $ vim Podfile

然后在Podfile文件中输入以下文字：

platform :ios, '9.2'

pod "SDWebImage", "~> 2.0”

然后保存退出。vim环境下，保存退出命令是 :wq

这时候，你会发现你的项目目录中，出现一个名字为Podfile的文件，而且文件内容就是你刚刚输入的内容。

这时候，你就可以利用CocoPods下载AFNetworking类库了,运行以下命令：$ pod install

#### 1.3. 用cocopod管理第三方框架的时候我想改版本，怎么办到？

可以直接或者终端打开Podfile, 修改Podfile文件中第三方框架的版本

### 2. 关于SDWebImage的问题:

#### 2.1. SDWebImage的原理实现机制如何解决TableView卡的问题?

SDWebImage内部实现过程

1> 入口 setImageWithURL:placeholderImage:options: 会先把 placeholderImage 显示，然后 SDWebImageManager 根据 URL 开始处理图片。

2> 进入 SDWebImageManager-downloadWithURL:delegate:options:userInfo:，交给 SDImageCache 从缓存查找图片是否已经下载 queryDiskCacheForKey:delegate:userInfo:.

3> 先从内存图片缓存查找是否有图片，如果内存中已经有图片缓存，SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo: 到 SDWebImageManager。

4> SDWebImageManagerDelegate 回调 webImageManager:didFinishWithImage: 到 UIImageView+WebCache 等前端展示图片。

5> 如果内存缓存中没有，生成 NSInvocationOperation 添加到队列开始从硬盘查找图片是否已经缓存。

6> 根据 URLKey 在硬盘缓存目录下尝试读取图片文件。这一步是在 NSOperation 进行的操作，所以回主线程进行结果回调 notifyDelegate:。

7> 如果上一操作从硬盘读取到了图片，将图片添加到内存缓存中（如果空闲内存过小，会先清空内存缓存）。SDImageCacheDelegate 回调 imageCache:didFindImage:forKey:userInfo:。进而回调展示图片。

8>如果从硬盘缓存目录读取不到图片，说明所有缓存都不存在该图片，需要下载图片, 回调 imageCache:didNotFindImageForKey:userInfo:。

9> 共享或重新生成一个下载器 SDWebImageDownloader 开始下载图片。

10> 图片下载由 NSURLConnection 来做，实现相关 delegate 来判断图片下载中、下载完成和下载失败。

11> connection:didReceiveData: 中利用 ImageIO 做了按图片下载进度加载效果。

12> connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder 做图片解码处理。

13> 图片解码处理在一个 NSOperationQueue 完成，不会拖慢主线程 UI。如果有需要对下载的图片进行二次处理，最好也在这里完成，效率会好很多。

14> 在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成，imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader。

15> imageDownloader:didFinishWithImage: 回调给 SDWebImageManager 告知图片下载完成。

16> 通知所有的 downloadDelegates 下载完成，回调给需要的地方展示图片。

17> 将图片保存到 SDImageCache 中，内存缓存和硬盘缓存同时保存。写文件到硬盘也在以单独 NSInvocationOperation 完成，避免拖慢主线程。

18> SDImageCache 在初始化的时候会注册一些消息通知，在内存警告或退到后台的时候清理内存图片缓存，应用结束的时候清理过期图片。

19> SDWI 也提供了 UIButton+WebCache 和 MKAnnotationView+WebCache，方便使用。

20> SDWebImagePrefetcher 可以预先下载图片，方便后续使用。

如何解决tableView卡 , 通过设置最大并发数, 设置当前页的cell, 而不是把所有cell一次性设置完, 以及数据图片的三级缓存, 直接保存在内存中和沙盒缓存中进行读取. 降低网络请求的次数, 不仅节约用户流量. 也会保证tableView滑动的流畅性

#### 2.2. SDWebImage怎样实现图片的缓存机制的？

图片的缓存, 内存缓存, 沙盒缓存, 操作缓存, 以tableViewController为例:

每次cell需要显示,都需要重新调用- (UITableViewCell \*)tableView:(UITableView \*)tableView cellForRowAtIndexPath:(NSIndexPath \*)indexPath { }方法

每次调用tableView显示行的数据源方法时, 如果需要从网络加载图片, 就需要将加载图片这样的耗时操作放在子线程上执行, 从网络上下载的图片可以以键值对的形式保存在定义的可变字典中 ,将每张图片的唯一的路径作为键,将从网络下载下来的图片作为值, 保存在内存缓存中, 这样每次滑动tableView cell重用时就直接判断内存缓存中有没有需要的图片, 如果有就不需要再次下载,在没有出现内存警告或者程序员手动清理内存缓存时, 就直接从内存缓存中获取图片.

为了每次退出程序,再次进入程序时, 不浪费用户的流量, 需要将第一次进入程序时加载的图片保存在本地沙盒缓存文件中, 在沙盒中保存的图片数据没有被改变之前, 下次开启程序就直接从沙盒的缓存文件中读取需要显示的图片, 并将沙盒缓存文件夹(Cache)中保存的图片保存到内存缓存中, 这样用户每次滑动tableView cell重用时直接从内存缓存中读取而不是从沙盒中读取, 节约时间.

#### 2.3. AFN的几个版本和SDWebImage的几个版本?

1> AFN的几个版本:

如果要兼容4.3或者MAC OS X 10.6，需要用最新发布的0.10.x版本

AFNetworking 0.10.x<https://github.com/AFNetworking/AFNetworking/tree/0.10.x>

如果你想要兼容IOS5或MAC OS X 10.7，那你需要用最新发布的1.x版本

[AFNetworking 1.x的下载地址https://github.com/AFNetworking/AFNetworking/tree/1.x](https://github.com/AFNetworking/AFNetworking/tree/1.x)

AFNetworking 2.0或者之后的版本需要xcode5.0版本并且只能为IOS6或更高的手机系统上运行，如果开发MAC程序，那么2.0版本只能在MAC OS X 10.8或者更高的版本上运行。

[AFNetworking 2.0的下载地址https://github.com/AFNetworking/AFNetworking](https://github.com/AFNetworking/AFNetworking)

下载2.5版本：<http://afnetworking.com/>

AFNetworking 3.0正逢 iOS 9 发布，NSUrlConnection 弃用，改用 NSURLSession，AFNetworking 也针对性发布了 3.0 版本

2> SDWebImage的几个版本 :

SDWebImage是一个开源第三方库，最简单的使用方式是以UIImageView类目的方式提供下载网络图片的方法。除此之外，还可以使用它作为图片异步下载器、图片自动缓存、支持gif动态图等，它会保证相同的url图片资源只下载一次，永远不会锁住主线程，同时支持gcd和arc、arm64。总之，使用SDWebImage下载网络图片可以提高各种性能。

注意：SDWebImage3.0和2.0是两个分开的仓库，低于iOS 5.1.1的只能使用2.0版本。

3> 如何集成SDWebImage:

将SDWebImage集成到项目中的方法有三个：

（1）用cocoapod下载管理

传送门： <http://blog.csdn.net/awaylin113/article/details/39347893>

（2）下载framework

A、下载并解压最新版本的framework： [down page](https://github.com/rs/SDWebImage/releases)

B、以添加文件的方法将SDWebImage.framework添加到工程里，勾选copy items

C、以添加框架的方法将ImageIO.framework的框架添加到工程里。

D、打开“build settings”->"linking"->"other linker flags",填写“-ObjC”

如果你的工程里面有用到Parse、RestKit、opencv2等扩展库，就用“

-force\_load SDWebImage.framework/Versions/Current/SDWebImage”替代“-ObjC”.

如果你的工程有用到Parse、RestKit、opencv2等扩展库，而且你使用cocoapods下载SDWebImage，

那么用“-force\_load $(TARGET\_BUILD\_DIR)/libPods.a”替代“-ObjC”.

E、在使用到SDWebImage的地方导入头文件：

#import <SDWebImage/UIImageView+WebCache.h>

（3）下载源码

### 3. 关于AFN的问题:

#### 3.1. 你用过哪些第三方框架？使用 AFNetworking做过断点续传吗？

使用过一些第三方框架, 例如AFN,SDWebimage,FMDB, MBProgressHUD, Masonry, SnapKit

* 断点续传的主要思路:

1> 检查服务器文件信息

2> 检查本地文件

如果比服务器文件小, 断点续传, 利用 HTTP 请求头的 Range 实现断点续传

如果比服务器文件大，重新下载

如果和服务器文件一样，下载完成

#### 3.2. 使用第三方框架的时候有没有仔细研读过它们的源码(AFNetworking)？

AFNetworking主要是对NSURLSession和NSURLCollection的封装,其中主要有以下类:

AFHTTPRequestOperationManager :内部封装的是 NSUrlConnection,负责发送网络请求,使用最多的一个类.

AFHTTPSessionManager :内部封装是 NSUrlSession ,负责发送网络请求,使用最多的一个类.

AFHTTPRequestOperationManager 和 AFHTTPSessionManager :定义的 API(方法名称)是一模一样,没有任何区别.

AFNetworkReachabilityManager :实时监测网络状态的工具类.当前的网络环境发生改变之后,这个工具类就可以检测到.

AFSecurityPolicy :网络安全的工具类, 主要是针对 HTTPS 服务.

AFURLRequestSerialization :序列化工具类,基类.上传的数据转换成JSON格式(AFJSONRequestSerializer).使用不多.

AFURLResponseSerialization :反序列化工具类;基类.使用比较多:

AFJSONResponseSerializer; JSON解析器,默认的解析器.

AFHTTPResponseSerializer; 万能解析器; JSON和XML之外的数据类型,直接返回二进制数据.对服务器返回的数据不做任何处理.

AFXMLParserResponseSerializer; XML解析器;

#### 3.3. AFN默认超时时长是多少啊?

AFN的默认超时时长是60s.

### 4. 关于ASI的问题:

#### 4.1. 使用过 ASI 吗？它跟 AFN 的实现原理是怎么样的？

1> AFN基于NSURL，ASI基于底层的CFNetwork框架，因此ASI的性能优于AFN

2> AFN采取block的方式处理请求，ASI最初采取delegate的方式处理请求，后面也增加了block的方式

3> AFN只封装了一些常用功能，满足基本需求，直接忽略了很多扩展功能，比如没有封装同步请求；ASI提供的功能较多，预留了各种接口和工具供开发者自行扩展

4> AFN直接解析服务器返回的JSON、XML等数据，而ASI比较原始，返回的是NSData二进制数据

### 5. 关于NSURLSession与NSURLCollection的问题:

#### 5.1. NSURLConnection的几个常用的代理?

NSURLConnectionDownloadDelegate :能够实现监听下载进度!但是下载之后,找不到下载好的文件!

NSURLConnectionDataDelegate 是针对数据下载提供的方法!需要注意的是,需要自己实现监听进度的业务逻辑!

利用 NSURLConnection 的异步回调进行文件下载:

1> 如果是小文件下载,问题不大! 可以直接使用异步回调进行下载

2> 如果使用异步回调的方法进行大文件下载,则会出现内存暴涨的情况!

内存暴涨的原因: 大文件下载之后,默认是放在内存中的,所以下载的文件越大,越耗费内存.

存在的缺点: 使用异步回调实现文件,无法监听下载进度!并且对于大文件下载,会造成内存暴涨!

基于以上两点,一般,在进行文件下载的时候,使用代理回调监听下载进度!并且在下载文件的时候,手动管理内存!

#### 5.2. NSURLConnection&NSURLSession的区别?

虽然 NSURLConnection 在 iOS 9.0 中已经被废弃，但是作为资深的 iOS 程序员，必须要了解 NSURLConnection 的细节,

1> NSURLSession: 用于替代 NSURLConnection

2> 支持后台运行的网络任务

3> 暂停、停止、重启网络任务，不再需要 NSOperation 封装

4> 请求可以使用同样的配置容器

5> 不同的 session 可以使用不同的私有存储

6> block 和代理可以同时起作用

7> 直接从文件系统上传、下载

## 第三方SDK

### 1. 举出5个以上你所熟悉的iOS sdk库有哪些和第三方库有哪些?

友盟（包括第三方登录和分享），高德地图，百度地图,AFN,SDWebimage,FMDB, MBProgressHUD等等

1> iOS-sdk:

Foundation.framework,

CoreGraphics.framework,

UIKit.framework,

MediaPlayer.framework,

CoreAudio.framework

2> 第三方库:

1.Json编码解码;

2.ASIHTTPRequest等相关协议封装;

3.EGORefreshTableHeaderView下拉刷新代码;

4.AsyncImageView 异步加载图片并缓存;

5.SDWebImage 简化网络图片处理

3> 框架分类:

音频和视频

         Core Audio

         OpenAL

         Media Library

        AVFoundation

数据管理

        Core Data

        SQLite

图片和动画

        Core Animation

        OpenGL ES

        Quartz 2D

网络

        Bonjour

        WebKit

        BSD Sockets

用户应用

        Address Book

        Core Location

        Map Kit

        Store Kit

### 2. 关于第三方SDK的问题

#### 2.1. 常使用的第三方框架有哪些？

友盟（包括第三方登录和分享），高德地图，百度地图,AFN,SDWebimage,FMDB, MBProgressHUD等等

#### 2.2. 第三方API是怎么用的？

大公司的开放API,github上面别人发布的框架用第三方API, 在官方文档都有说明，按照官方文档的一步一步做参考官方提供的示例程序 先自己创建一个工程试试，等熟悉了，在使用到项目中

### 3. 关于地图的问题:

#### 3.1. 最常使用的地图是什么？

百度地图, 高德地图

#### 3.2. 集成地图时都使用了哪些技术？

使用地理编码和反编码进行地图定位，查找等功能，大头针的使用，路线查找.

#### 3.3. 集成地图有什么用?

可以给用户更好的体验，使用苹果原生地图在位置定位的时候需要跟后台的数据进行转换，容易出错，因为一般后台用的都是百度地图，两者之间的坐标不同，就算转换后也容易出现偏差

#### 3.4.地图导航不能用了怎么办

提示用户打开导航定位功能

### 4. 关于支付的问题:

#### 4.1. 常用的支付方式用那些？怎样实现支付宝的集成和使用？

下载SDK，申请账号、交费、加入客服群，按照SDK进行集成 百度钱包、微信支付、支付宝、银联

#### 4.2. 简单给出购物车的实现思路？

<http://www.cocoachina.com/ios/20150908/13289.html>

#### 你在这个项目中订单是怎样实现网上付款的？

IAP的商品从消费性质上分为四种：

1.消耗型商品（Consumable）,比如游戏道具，子弹，药品等等。由于这类商品可以被消耗，所以支持重复购买。苹果应用商店不保存此类商品的购买记录，如果要保存则需要开发者同步到自己的服务器上。

2.非消耗型商品（Non-Consumable），比如游戏关卡，隐藏地图等等。这类商品只要购买一次便可以了，苹果应用商店里每一个用户对非消耗型商品的购买都有记录，可以在不同的设备上恢复购买状态，这个恢复的过程叫做Restore。

3.自动重置型订阅（Auto-Renewable Subscriptions），比如电子杂志，读物等。消费者购买这类商品时会从列表中选择一个有效期限，卖家在定义商品的时候从一群固定的选项 中选择添加一个有效期，比如7天，一个月，两个月。过了有效期之后，商品的购买状态会被自动重置成未购买，要想继续获得内容则需要再次订阅。这种类型的商品和非消耗型商品一样，会在苹果商店内保存购买记录。

4.非自动重置型订阅（Non-Renewing Subscription），比如用户订阅电子杂志和读物报刊时需要从自定义的期限列表中选择期限，而不是苹果提供的固定选项，比如9天，一个半月或任意时间。在这种情况下，苹果商店无法根据期限来控制订阅的到期行为，所以一切都需要开发商自己编写相应的逻辑来实现。

在创建产品类型的选择决定着客户端/server的处理流程，为此一定要在选择类型时考虑所创建的产品特性，选择时，如果对于自动重置型订阅，特别需要注意，此类型的商品的必须依据苹果设置的使用期限进行创建，如一个月、三个月、半年等，不能创建任意时间段，同时，在此类型中，经过在沙盒测试发现，当产品到期时，苹果服务器会自动进行续订操作，因此，客户端需根据监听处理苹果的续订事件进行续订结果提示用户；

<http://www.aiuxian.com/article/p-2515575.html>

### 5. 越狱手机的IAP问题

由于越狱手机可能安装了黑客的恶意程序，监听网络数据，支付凭证中并不包含任何用户的apple id信息，所以我们的app和服务器无法知道这个凭证是谁买的，如果恶意程序截获苹果服务器的有效支付凭证，但恶意程序将假的支付凭证发给后台server导致原支付的账号验证失败，而此时恶意程序将截获的有效支付凭证对应到另外的支付账号上，就会导致该恶意程序设置的账号通过正确的支付凭证而获取server的认证。

所以，对于越狱的手机可禁用IAP支付，采用第三方支付平台进行支付的方式。

### 内购使用了什么技术?

IAP流程:

用户点击内购按钮 🡪 用户确认购买内容 🡪 用户通过App Atore账户验证 🡪 苹果服务器验证用户请求 🡪 苹果服务器从用户帐号扣款 🡪 苹果向用户返回购买成功信息 🡪 软件接收并显示用户购买信息.

### <http://www.zhihu.com/question/20997664>

### 7. 关于短信验证的问题:

#### 7.1. SMS验证的时候用的是自己公司的服务器，还是后台，还是sms的服务器

#### 7.2. 分析一下使用手机获取验证码注册账号的实现逻辑(给了一个示例图)，发送到手机的验证码超过60秒钟后重新发送

定义一个label属性，赋值为59秒 在定义一个count 设置一个timer 每次减少一秒 把count-- 再把count的值拼接到label上 当count == 0 的时候 在显示验证码输入

### 8. 关于二维码

#### 8.1. 不联网扫描二维码支付?

### 9. 关于蓝牙

#### 9.1. 一个黑屋子里，没网没蓝牙，两个手机怎么传图片?

### 10. 关于推送

#### 10.1. 什么是推送消息？(What is push notification?)

推送通知简单点就是客户端获取资源的一种手段。普通情况下，都是客户端主动的pull。推送则是服务器端主动push。

第一步:UIApplication向APNS注册push notification服务

应用程序 要支持 推送服务(在网页里配置) (1)https://developer.apple.com/devcenter/ios/index.action (2)登录 苹果开发者账号(注意是收费账号,$99或$299) 3)下载push证书(主要是给程序签名,push服务只有收费开发者才具备。所以需要签名验证),如果没有 push证书,创建一个push证书(App ID->钥匙串程序生成request->push证书)注意事项:App ID的 Bundle ID必须和程序plist文件里的Bundle identifier一致。App ID一旦生成,将不可修改。 (4)把证书安装到钥匙串里(双击证书文件) (5)生成 编译程序 用的描述文件(网页里进行) 2、向APNS注册push服务(UIApplication的registerForRemoteNotificationTypes:方法)

 第二步 获取APNS分配的DeviceToken(64位16进制串) - (void)application:(UIApplication \*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*)deviceToken

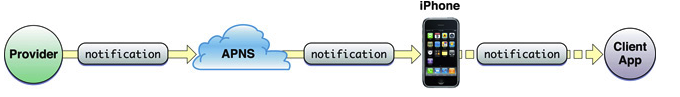
 第三步 把DeviceToken发送给自己的后台服务器,服务器记录每台设备的DeviceToken以便日后推送信息给客 户端。(需要有一个网络接口,让客户端发送DeviceToken)

第四步 服务器推送信息给客户端

1>、服务器除了需要有客户端的DeviceToken之外,还要有push证书,对push的内容进行签名。(苹果为了防 止 恶意向客户端(比如DeviceToken泄露了)发送消息,每次推送消息,都需要证书进行签名,从而避免黑客恶 意攻击用户手机。) 2>、如果你的服务器是java写的,可以直接使用钥匙串导出的p12文件(证书和密钥一起导出)。如果你的服务器 是php写的,因为php语言不支持p12文件类型,需要转换为pem文件。 3、将p12转换为pem文件:终端 先找到你p12所在的目录 openssl pkcs12 -in CertificateName.p12 - outCertificateName.pem -nodes 4、服务器发送信息给APNS,APNS自动将信息推送给客户端

第五步 客户端处理收到的信息 - (void)application:(UIApplication \*)application didReceiveRemoteNotification:(NSDictionary \*)userInfo 注意事项 1、测试版的push证书仅仅用于开发期间测试,发布版的程序需要生成一个发布版的push证书。 2、测试版APNS的ssl地址和发布版的ssl地址不同 3、测试版的DeviceToken和发布版的DeviceToken也不同 4、php文件要喝pem文件放在同一目录。 5、除了alert sound和badge之外,json串里还可以包含自定义信息。 6、推送的信息最大255字节 7、推送的信息受网络影响较大,有可能造成延迟甚至丢失,重要信息的传递不应该使用push通知,应该有专门的 后台接口。8、借助push推送,两个客户端可以实现即时通信,工程里面存放我们的p12文件,客户端自己组织json串,发 送请求到APNS。

#### 10.2. Provider是指某个iPhone软件的Push服务器，这篇文章我将使用.net作为Provider。 APNS是Apple Push Notification Service（Apple Push服务器）的缩写，是苹果的服务器。

上图可以分为三个阶段。

第一阶段：.net应用程序把要发送的消息、目的iPhone的标识打包，发给APNS。

第二阶段：APNS在自身的已注册Push服务的iPhone列表中，查找有相应标识的iPhone，并把消息发到iPhone。

第三阶段：iPhone把发来的消息传递给相应的应用程序， 并且按照设定弹出Push通知。

http://blog.csdn.net/zhuqilin0/article/details/6527113    //消息推送机制

#### 10.3、把程序自己关掉和程序进入后台,远程推送的区别

1>. 关掉后不执行任何代码, 不能处理事件

2>. 应用程序进入后台状态不久后转入挂起状态。在这种状态下，应用程序不执行任何代码，并有可能在任意时候从内存中删除。只有当用户再次运行此应用，应用才会从挂起状态唤醒，代码得以继续执行

3>.或者进入后台时开启多任务状态，保留在内存中，这样就可以执行系统允许的动作

4>.远程推送是由远程服务器上的程序发送到APNS,再由APNS把消息推送至设备上的程序,当应用程序收到推送的消息会自动调用特定的方法执行事先写好的代码

#### 五、本地通知和远程推送通知对基本概念和用法？

 本地通知和远程推送通知都可以向不在前台运行的应用发送消息,这种消息既可能是即将发生的事件,也可能是服务器的新数据.不管是本地通知还是远程通知,他们在程序界面的显示效果相同,都可能显示为一段警告信息或应用程序图标上的微章.

 本地通知和远程推送通知的基本目的都是让应用程序能够通知用户某些事情, 而且不需要应用程序在前台运行.二者的区别在于本地通知由本应用负责调用,只能从当前设备上的iOS发出, 而远程通知由远程服务器上的程序发送到APNS,再由APNS把消息推送至设备上的程序

## 关于即时通讯的问题:

### 1. 关于即时通讯的问题:

#### 1.1. 通信的时候是通过什么把信息发给服务器的

#### 1.2 服务端提供一个pdf文件怎么能直接展示在界面上

#### 1.3. 经常使用的即时通讯有哪些？你擅长的有哪些？你不擅长什么？

#### 1.4. 怎样实现信息的共享？

#### 1.5. 介绍一下即时通讯? 即时通讯大数据如何处理?

用put上传到文件服务器，然后发带url的自定义格式的给对方，对方接收到之后下载

#### 1.6. 对socket的理解?

### 2. 关于融云环信的问题:

#### 2.1. 使用过融云吗？使用过友盟吗？

使用过友盟的社会化分享，集成了qq空间、qq好友、微信、朋友圈、微博分享和数据统计，统计流量来源、内容使用、用户属性和行为数据。可以熟练的自定义分享的界面 <http://www.rongcloud.cn/docs/open_source.html>

### 3. 关于xmpp的问题:

#### 4.1. xmpp做过没有?

XMPP(Extensible Messaging and Presence Protocol,前称)是一种以XML为基础的开放式实时通信协议,是 经由互联网工程工作小组(IETF)通过的互联网标准。简单的说,XMPP就是一种协议,一种规定。就是说,在网络上传 东西,要建立连接,TCP/IP连接,建立后再传东西,而XMPP就是规定你传的东西的格式。XMPP是基于XML的协议。

优点:

 开放: XMPP协议是自由、开放、公开的,并且易于了解。 而且在客户端 、 服务器 、 组件 、 源码库等方面,都已经各自有多种实现。

标准: 互联网工程工作小组( IETF )已经将Jabber的核心XML流协议以XMPP之名,正式列为认可的实时通信及Presence技术。 而XMPP的技术规格已被定义在RFC 3920及RFC 3921 。 任何IM供应商在遵循XMPP协议下,都可与Google Talk实现连接。

证实可用: 第一个Jabber(现在XMPP)技术是Jeremie Miller在1998年开发的,现在已经相当稳定;数以百计的开发者为XMPP技术而努 力。 今日的互联网上有数以万计的XMPP服务器运作着,并有数以百万计的人们使用XMPP实时传讯软件。

分散式: XMPP网络的架构和电子邮件十分相像;XMPP核心协议通信方式是先创建一个stream,XMPP以TCP传递XML数据流,没有 中央主服务器。 任何人都可以运行自己的XMPP服务器,使个人及组织能够掌控他们的实时传讯体验。  安全: 任何XMPP协议的服务器可以独立于公众XMPP网络(例如在企业内部网络中),而使用SASL及TLS等技术的可靠安全性,已自 带于核心XMPP技术规格中。

可扩展: XML 命名空间的威力可使任何人在核心协议的基础上建造定制化的功能;为了维持通透性,常见的扩展由XMPP标准基金会 。

弹性佳: XMPP除了可用在实时通信的应用程序,还能用在网络管理、内容供稿、协同工具、文件共享、游戏、远程系统监控等。

多样性: 用XMPP协议来建造及布署实时应用程序及服务的公司及开放源代码计划分布在各种领域;用XMPP技术开发软件,资源及支持的 来源是多样的,使得使你不会陷于被“绑架”的困境。

缺点

数据负载太重: 随着通常超过70%的XMPP协议的服务器的数据流量的存在和近60%的被重复转发,XMPP协议目前拥有一个大型架空中存在的 数据提供给多个收件人。 新的议定书正在研究,以减轻这一问题。 没有二进制数据: XMPP协议的方式被编码为一个单一的长的XML文件,因此无法提供修改二进制数据。 因此, 文件传输协议一样使用外部的 HTTP。 如果不可避免,XMPP协议还提供了带编码的文件传输的所有数据使用的Base64 。 至于其他二进制数据加密会话 (encrypted conversations)或图形图标(graphic icons)以嵌入式使用相同的方法。

### 4. iOS开发系列--通讯录、蓝牙、内购、GameCenter、iCloud、Passbook系统服务开发汇总

<http://blog.csdn.net/cos_sin_tan/article/details/50432300>

# 14音频视频（★）

# 15其他问题（★）

### 1.你做iphone开发时候,有哪些传值方式,view和view之间是如何传值的?

block, target-action ,代理,属性

### 2. XML是什么? XML与HTML的区别?

XML的简单使其易于在任何应用程序中读写数据，这使XML很快成为数据交换的唯一公共语言，虽然不同的应用软件也支持其它的数据交换格式，但不久之后他们都将支持XML，那就意味着程序可以更容易的与Windows,Mac OS,Linux以及其他平台下产生的信息结合，然后可以很容易加载XML数据到程序中并分析他，并以XML格式输出结果。

　　XML去掉了之前令许多开发人员头疼的SGML（标准通用标记语言）的随意语法。在XML中，采用了如下的语法：

　　1>  任何的起始标签都必须有一个结束标签。

　　2>  可以采用另一种简化语法，可以在一个标签中同时表示起始和结束标签。这种语法是在大于符号之前紧跟一个斜线（/），例如<tag/ >。XML解析器会将其翻译成<tag></tag>。

　　3>  标签必须按合适的顺序进行嵌套，所以结束标签必须按镜像顺序匹配起始标签，例如this is asamplestring。这好比是将起始和结束标签看作是数学中的左右括号：在没有关闭所有的内部括号之前，是不能关闭外面的括号的。

　　4>  所有的特性都必须有值。

　　5>  所有的特性都必须在值的周围加上双引号。

XML与HTML的设计区别是：XML的核心是数据，其重点是数据的内容。而HTML 被设计用来显示数据，其重点是数据的显示。

XML和HTML语法区别：HTML的标记不是所有的都需要成对出现，XML则要求所有的标记必须成对出现；HTML标记不区分大小写，XML则 大小敏感,即区分大小写。

### 3. 谈谈安卓与苹果的优缺点

苹果系统优点是左右流畅，软件多，界面华丽，图标统一，很美观；缺点是系统封闭，不允许用户过多的个性化设置，而且只能在苹果手机上用。安卓系统优点是开放，可以自己扩展的东西很多，支持的硬件也多，各个价位的手机都有；缺点就是软件太杂乱，兼容性有问题，图标混乱不美观。iOS的确比android流畅，这仅仅体现在较大软件切换时，其他差不多流畅，iOS并不能做到完全后台，如果它完全后台估计也不会比安卓流畅多少。反之，如果安卓只是注重单个运行，流畅度也会大大提升，iOS系统更新没有android那么频繁，爱体验的人会选安卓，那些怕烦的会选iOS。iOS的硬件需求选不及android，以至于android机会相对iOS较热，较费电额。

|  |  |  |
| --- | --- | --- |
| |  | | --- | | android机优缺点  优点: 1>.可以自定义桌面 以及各类插件 2>.部分手机支持FM收音机 3>.各种ROM刷机 正所谓萝卜白菜各有所爱 众多ROM里 总有适合你的 4>.实体按键比iphone多 有些操作会很方便  例子“我看电子书的时候 有一个按键可以设置为翻页 按一下翻一页” 5>.随意换电池，也可以用大容量电池【换特殊背壳才行】 6>.机型多，1000左右的也有，四五千的也有 选择多  缺点: 1.机型众多导致的应用不通用，每个机器都要找自己专用的 2.机型升级过快导致【拼硬件】的现象，用户体验较差 3.各种ROM良莠不齐，有的狂吃电量之类的缺陷 4.一些老机器被Google抛弃 没有官方升级下面说iphone的优缺点 优点 1>.简单易懂，容易操作 2>.应用程序多，质量好，种类齐全 3>.硬件和软件高度匹配，硬件能很好地运转软件，软件能很好地发挥硬件的机能 4>.配件多 各种壳子 镜头  5>.用户体验好 缺点 1>.【中国区】售后服务不完善 导致维修一下就好好几百 2>.软件收费【不过一般的10块左右 贵的大概60以内吧 少吃顿必胜客 大M 就好了】 3>.实体按键少 | |  | |

### 4. OC这门语言，和c c++比，你觉得哪个效率高，高到哪里了?

object\_c比C++稍慢，是因为object\_c的一些语言特性，区别主要有以下一些方面。

单一继承：Objective-C不支持多重继承，（同Java和Smalltalk），而C++语言支持多重继承。

动态：Objective-C是动态定型（dynamicaly typed）所以它的类库比C++要容易操作。Objective-C 在运行时可以允许根据字符串名字来访问方法和类，还可以动态连接和添加类。C++ 跟从面向对象编程里的Simula67(一种早期OO语言)学派，而Objecive-C属于Smalltalk学派。 在C++里，对象的静态类型决定你是否可以发送消息给它，而对Objecive-C来说，由动态类型来决定。Simula 67学派更安全，因为大部分错误可以在编译时查出。而Smalltalk学派更灵活，比如一些Smalltalk看来无误的程序拿到Simualr 67那里就无法通过。从很多方面来看，C++和Objective-C的差别，与其说时技术上的，不如说是思维方式上的

### 5. 苹果的安全机制有哪些

1> 没经过用户同意，你不能随便获取用户信息。

2> 所有的程序都在沙盒里运行，B程序不能进入A程序的运行范围。

3> 如果跟钱有关，比如说支付宝，这些底层的实现都是保密的，只提供接口供开发者调用，这样的话安全性得到保障。

4> 如果要防止代码被反编译，可以将自己的代码中的.m文件封装成静态库（.a文件）或者是framework文件，只提供给其它人.h文件。这样就保证了个人代码的安全性。

5> 网络登录的话跟用户名跟密码相关要发送POST请求，如果是GET请求的话密码会直接在URL中显示。然后同时要对帐号密码采用加密技术，加一句：我们公司用的是MD5，但是现在MD5有一个专门的网站来破解，为了防止这个，可以采用加盐技术。

### 6. 当存储大块数据是怎么做？

你有很多选择，比如：

使用`NSUerDefaults`

使用XML, JSON, 或者 plist

使用NSCoding存档

使用类似SQLite的本地SQL数据库

使用 Core Data

1>, NSUserDefaults的问题是什么？虽然它很nice也很便捷，但是它只适用于小数据，比如一些简单的布尔型的设置选项，再大点你就要考虑其它方式了

2>, XML这种结构化档案呢？总体来说，你需要读取整个文件到内存里去解析，这样是很不经济的。使用SAX又是一个很麻烦的事情。

3>, NSCoding？不幸的是，它也需要读写文件，所以也有以上问题。

4>, 在这种应用场景下，使用SQLite 或者 Core Data比较好。使用这些技术你用特定的查询语句就能只加载你需要的对象。在性能层面来讲，SQLite和Core Data是很相似的。他们的不同在于具体使用方法。Core Data代表一个对象的graph model，但SQLite就是一个DBMS。Apple在一般情况下建议使用Core Data，但是如果你有理由不使用它，那么就去使用更加底层的SQLite吧。如果你使用SQLite，你可以用FMDB(https://[GitHub](http://blog.jobbole.com/6492/).com/ccgus/fmdb)这个库来简化SQLite的操作，这样你就不用花很多经历了解SQLite的C API了

### 7. 如何对APP进行性能优化

1>, 在正确的地方使用reuseIdentifier

table view用 `tableView:cellForRowAtIndexPath:` 为rows分配cells的时候，它的数据应该重用自UITableViewCell。 一个table view维持一个队列的数据可重用的UITableViewCell对象。

2>, 尽量把views设置为透明

如果你有透明的Views你应该设置它们的opaque属性为YES。原因是这会使系统用一个最优的方式渲染这些views。这个简单的属性在IB或者代码里都可以设定。

3>, 避免过于庞大的XIB

尽量简单的为每个Controller配置一个单独的XIB，尽可能把一个View Controller的view层次结构分散到单独的XIB中去, 当你加载一个引用了图片或者声音资源的nib时，nib加载代码会把图片和声音文件写进内存。在OS X中，图片和声音资源被缓存在named cache中以便将来用到时获取.

4>, 不要阻塞主线程

永远不要使主线程承担过多。因为UIKit在主线程上做所有工作，渲染，管理触摸反应，回应输入等都需要在它上面完成

大部分阻碍主进程的情形是你的app在做一些牵涉到读写外部资源的I/O操作，比如存储或者网络。

5>, 在Image Views中调整图片大小

如果要在`UIImageView`中显示一个来自bundle的图片，你应保证图片的大小和UIImageView的大小相同。在运行中缩放图片是很耗费资源的 .

### 8. iOS APP审核被拒的原因

1>、程序有重大bug，程序不能启动，或者中途退出。

2>、绕过苹果的付费渠道，我们之前游戏里的用兑换码兑换金币。

3>、游戏里有实物奖励的话，一定要说清楚，奖励由本公司负责，和苹果没有关系。

4>、用到苹果的标志。（应用的设计和Apple的Logo风格太像了也会被拒）

5>、网络功能不能正常访问。

6>、图标不能点击，不能点击的图标要置灰，或者直接隐藏。

7>、没有设置default页，启动画面为黑屏，有一定概率被拒绝。

8>、一个应用在线，但你想在发一个豪华版之类的，再开一个应用也会被拒绝。

9>、用了著名游戏的关键字，比如说“愤怒的小鸟”之类的。

10>、技术支持地址写的微博地址，于是被拒绝了。原因是：不能将需要登陆才能访问的网址作为技术支持地址。

11>、年龄设置太低,说是有成人内容,被拒.修改内容后上线.

12>、同一软件多个版本只是针对不同的国家和内置的语言不同.前面几个上线,后面几个被拒.让改成一个程序做程序内下载资源.

13>、原因是我们的一个软件只允许联通用户验证真实身份并发布信息。apple要求要么放弃验证，要么允许移动和电信用户验证。

# 16项目常见问题（★）

### 1.公司人员构成

不同的公司人员构成差别很大，下面给个真实案例说明。

真实案例1：北京某给政府做即时通信的公司，中等规模。

公司大致分为开发部（有10几个人），产品组（2个产品经理，1个美工），测试组（3个人，兼客服）和人事行政部（3个）。

开发部又以项目分组：密讯组、加密电话组、原网组、rom组（前四个都是安卓）和网络组。每组设一个项目负责人，APP组员一般情况下是2-3个人维护开发。根据各个项目的开发情况，总监会灵活调配开发人员。

真实案例2：北京某科技公司，给广电总局和各个电视台做后台系统，总共人员280人。

公司分销售部、财务部、事业一部、事业二部、事业三部、基础研发部。每个事业部都有三个组，分别为Java开发组，C/C++开发组，测试组。每个组大概5到8个人。

### 2.开发周期

以真实项目密讯（即时通讯）为例：

一般3~5个月一个大版本（目前6.0：重大性能优化，加密算法的改变等），原网组几个APP也大致这个时间。

平时的维护无特殊情况是一周一个小版本（例如：6.0.1，两周交替，一周bug修复，一周小功能添加，重大bug修复或者添加重要功能，直接升6.1.0）。

### 3.项目中遇到的难题

以真实项目密讯（即时通讯）为例：

1>.对于初接触信息加密的新手来说，即时通讯所涉及的加密方案比较复杂，理清思路是开发前期比较麻烦的一块。（接着人家要问，展开回答）

2>.云文件的模块，服务器方面维护麻烦，客户端优化不足，前期用户体验不好，后来直接丢给阿里云服务器了。（关于断点续传，第三方平台，文件迁移等展开）

3>.长连接不太稳定，一直在优化，现在虽然差强人意也还凑合。（优化的过程可以展开）

4>.UI方面，密讯的聊天界面是精华（具体实现可以展开）

补充：

开发工具是Xcode，用的版本管理工具是Git，用户服务器是自己的，文件服务器用的是阿里云。

总结：

大家找项目的时候大致可以根据这样的思路去找。项目里用到什么技术，怎么实现，都有哪些重难点，从这几个方面入手进行准备。

面试的时候适当的去引导到你比较擅长的模块，然后进行展开延伸。这里就可以体现自己项目技术的难点，解决方案和自己的收获。

遇到不是很清楚的问题说出自己的理解，完全不懂的问题要坦诚别瞎说，可以跟面试官交流，尽量在面试中多学点东西。

### 4.项目中最大的收获

### 5.项目是如何上线的

以真实项目密讯（即时通讯）为例：每周三提测，每周四晚经测试组许可，打包发给网络部发布。（因为用户群体办公需要，若新版有测试未发现的bug，周五我们可以发布紧急修复版）。

### 6.项目是如何盈利的

以真实项目密讯（即时通讯）为例：就我所知，公司的盈利主要有两块：一个是政府立项，具体金额未知，另一个是加密电话组（驻扎奇虎360公司，目前该项目在奇酷手机上投入使用），一年是5百万，连续三年。另：我司卖给政府的定制手机（刷个rom，装一些我们的APP）是一台1万，手机原价2千。

### 7.绘制项目架构图

这里目前只给出公司项目包结构图：

以真实项目密讯（即时通讯）为例：

### 8.项目开发流程

以真实项目密讯（即时通讯）为例：

我进公司时，密讯已经迭代到4.0。现在跟大家说说从5.0-6.0这个大版本迭代的整个过程。（底层加密算法的替换，UI的大幅更改，相当于重做了一遍，有一定的借鉴价值）

1.技术总监开会，负责底层加密模块的同事给我们讲新算法（椭圆加密算法，这整个是项目的技术核心），根据项目的功能模块需求敲定框架。

2.分工：

开发部分：加密底层一人（用C语言做加密算法的so类库），JNI一人（对底层库的封装，测试接口，写接口文档），服务器一人（技术总监，erLang语言写的），APP两人（负责APP的实现，接口功能的具体测试）。

产品部分：产品经理负责设计功能的业务逻辑和交互，美工配合产品出图

测试部分：每开发一模块的功能就进行测试

3.三线并行：

①：（底层实现）加密底层封装 — 接口调试 — 写接口文档

②：服务器开发

③：框架重构 — UI实现 — 调通接口的功能接入实现 - 提测

### 9.你在项目中的角色

以真实项目密讯（即时通讯）为例：

我是密讯组的组员，平时主要负责维护密讯，也被抽调到原网组到新项目部分模块开发。

主要职责：

1.参与信息加密方案的制定

2.负责登录，联系人，云文件等模块的开发和维护

3.后续新功能的开发和bug的修复

### 10.你负责项目中的哪些模块(同上)

### 11.讲讲你负责模块的具体实现(同上)

### 12.项目中都用到了哪些第三发框架

以真实项目密讯（即时通讯）为例：

### 13.有没有自己写过框架

### 14.业余时间你是如何提高自己（学习）的

### 15.有没有自己的技术blog

### 16.你的职业规划

### 17.为什么离职

### 18.为什么选择我们公司

### 19.说说你们项目的亮点和不足

### 20.你们的项目是如何保持风格一致的

### 21.项目架构是如何搭建的

### 22.屏幕适配是如何解决的

### 23.都看过哪些源码

### 24.项目版本是如何升级的

### 25.用的什么版本控制工具

### 26.你能独立开发吗

### 27.App跟服务器是如何交互的

### 28.需求文档写过吗

### 29.接口文档写过吗

以真实项目密讯（即时通讯）为例：这里只给出部分接口文档截图。

### 30.云服务器都用过哪些

### 31.第三方平台都用过哪些

# 17面试实战记录（★★）

（本章节的内容记录了本人于2015年5月份在上海的面试的真实情况。在这里分享给大家，希望对大家有帮助。该部分内容网络连接：<http://bbs.itheima.com/thread-196394-1-1.html>）

## 挑战公司No.1：上海创梯信息科技有限公司

**公司地址：**上海杨浦区昆明路1209号尚凯大厦副楼504室



**面试时间：**5月12日 10:00 AM.

**面试结果：**顺利砍下Android技术总监，15K offer:victory:

**面试过程：**

10:00 到公司，前台MM给了张面试人员登记表，10分钟搞定表格。

10:30 由于公司BOSS正在面试其他人，因此，又等了几分钟。

10:40 在BOSS办公室与BOSS斗智斗勇，聊了有接近1个小时。

**疯狂的笔试+面试记录（前方内容“高能”:funk:，请准备高度精神集中前行）：**

**1. 笔试**

   由于BOSS正在打造公司新业务，刚刚开始组建公司技术团队，公司没有人懂IT技术，故本次面试没有笔试题，额。。。

**2. 面试过程问答精选：**

**旁白：**进入办公室后，我淡定等待，约莫几分钟后，一位35岁左右，看起来非常老道的BOSS走了进来。几句寒暄之后，问了我一些关于我的学校、专业、工作经验、接触的iOS项目等普通流水线式的没有营养问题，轻松搞定！接着，BOSS终于切入到了重点：

**BOSS问：**其实我们想做一个快递业务的APP，类似于滴滴打车。用户如果想要寄快递，只需要打开APP，查找附近都有哪些快递员，然后直接跟附近的快递员联系。如此，既能方便用户，又能提高快递员的收入。

**我（阳哥）答：**我有一个问题想要问一下，咱们公司（注意，一定要说“咱们”，让别人感觉你已经融入他们的公司团队了）不像顺丰，不是典型的物流公司，为什么我们要做快递类的APP呢（主动沟通交流，有问题就问，这样面试官才能跟你聊起来！）？

**BOSS答：**你知道的那些物流公司现在都是各自为政。例如，你用顺丰快递，你可以下载个顺丰的APP。但是，你以后可能还要用中通、申通、圆通等等这些物流公司，难道你要下载十几个APP吗？而我们要做通用的快递类APP！

**BOSS问：**你知道一个APP重要的是什么吗？

**我（阳哥）答：**用户体验！

**BOSS说：**不对，是流量，也就是用户数量（好吧，阳哥作为技术屌丝，关注多的就是APP用户体验，所以，不敢去反驳他，先顺着他的意思来）。

**BOSS问：**如果让你去做一个类似于滴滴打车的APP，你认为自己做的出来吗？

**我（阳哥）答：**像滴滴打车这样的APP，不仅仅只是客户端的问题。它不是由几个简单的客户端程序员就能做出来的，实际上，它应该是由一个技术团队完成的大型项目。您目前能够看得见的仅仅是用户客户端，看不见的是庞大的后台系统。说详细点就是：滴滴打车这样的项目应该分3部分：服务器端，用户客户端，出租车司机客户端。服务部端考虑的是数据的存储，业务的调度处理，以及与各个客户端的数据交互。而用户客户端我个人理解主要是一个UI的显示和与用户数据交互的功能。比如，用户将自己的当前坐标发送到服务器端（当前坐标可以通过百度地图、高德地图获取），服务器根据用户的坐标查找附近的所有空闲状态的出租车，然后将用户的用车需求推送到出租车司机客户端端。出租车司机接收到信息后，如果愿意接这笔单子，就向服务器发送同意信息。服务器就是作为一个中介将用户与出租车司机联系起来。大概逻辑就是这样一个过程，中间的技术细节比较多。总之，一个人做滴滴打车是不太现实的（大家可以看出来，这一段，阳哥是傻瓜式教学，没有说的太复杂，好让BOSS能够听明白。不然，技术说的太深，BOSS就不知道说什么了，还怎么聊的开心。面试的最终目的就是让对方觉得看你很顺眼，这是关键！）。

（顺便啰嗦一下，我们做为APP开发人员，不管是服务器端还是移动端都应该有所了解。如果仅仅会移动端开发技术而不懂服务端知识，长久看来，确实会很影响iOS程序员向更高层次发展。）

**BOSS问：**那么，如果我让你负责客户端的开发呢？

**我（阳哥）答：**客户端iOS开发并不难，比如支付宝是一个很强大的金融系统。但是，你让我做它的iOS端开发，我感觉我没问题。毕竟，iOS客户端重点是信息的展示和与用户的交互。所以一个人做客户端是没有问题的。但是，目前市场上的企业很少有一个人做一个客户端项目的。市场竞争如此激烈，一个人做耗时太长，可能你还没有做出来，产品已经被淘汰了。因此，应该多招几个iOS程序员，这样团队就算有人员流失也不会影响整个公司的发展（展示自己对于项目组人数把控的个人看法）。

**旁白：**几轮PK下来，BOSS基本上对我已经很信服了。

**BOSS说：**其实，在面试你之前，我已经招聘到了两个移动开发的程序员，一个是Android程序员（6K），一个是iOS程序员（8K），你可以看一下他们两个的面试登记信息和他们的简历。

**我（阳哥）问：**根据这两份简历可以看出来，这两个人的技术有些一般（不是装逼，他们的简历写的确实有点LOW）。

**BOSS问：**是的，我看的出来，你的经验要比他们丰富地多。所以，我准备让你出任我们公司的技术总监，带着他们两个做客户端开发，你认为你有信心做好吗？

**我（阳哥）答：**这个职位我以前没有做过，所以心里没有多大的底（阳哥认为这样的问题确实比较考验人，你直接回答“可以，没问题”，就会显得你浮夸、不谨慎、办事不牢靠。如果你直接回答“不能”，又显得你不敢担当，没有上进和挑战的精神。因此回答这样的问题必须两者兼顾，既要谦虚谨慎又要表现出富有挑战精神）。

**BOSS说：**我知道你没有做过，但是，一个人挑战任何一样新工作之前都是没有做过的。你有没有想过一旦你能够胜任我们的公司的技术总监，你以后的身价将和现在完全不同，所以，对自己要有一些信心（BOSS故意画大饼，就说明你有戏了！）。当然，这是一个管理岗位，需要什么样的技术人员你可以招。

**我（阳哥）说：**好吧，其实我希望挑战一下，但是我不一定能够达到您对我那么高的期望（始终保持谦虚的态度很重要！你的态度决定别人对待你的态度！）。

**BOSS问：**你尽力就好，请问你的期望月薪是多少？

**我（阳哥）答：**15k（第一次面试，我心里也不太清楚上海的市场，所以，试水一下）。

**BOSS说：**好的，没问题（竟然没有砍价:L，额，土豪公司）。

**旁白：**随后，阳哥又和BOSS聊了一些其他内容，由于BOSS马上要开会，就告诉阳哥如果你同接受我们的邀请，那么我希望我们明天下午可以好好聊聊。。。。之后，一堆寒暄，不列出来了。虽然，第一家面试就这样结束了，没有技术上太多的PK。但是，却使阳哥在精神上却得到了巨大的鼓励，面试第二家上海公司有了更大的信心！

**面试吐槽：**

       阳哥的上海处女面就这样丢掉了！遗憾的是BOSS不太懂技术，没有碰撞出技术的火花。但是，让人幸运的是他提供给我了一个技术总监的岗位。整个过程可以看到，面试的时候，面试官一方面会明中考察我们的技术。另一方面，在暗中实际上也在考察我们的沟通能力、思维逻辑能力、管理能力等这些软实力。所以，黑马的每一位达人，在黑马拼命码代码的同事，别忘了多和你的小伙伴分享技术，不要忘记中午的时候抓住每一次的公开演讲机会。有时候，一个人综合实力远大于单纯的技术竞争力！加油吧，骚年们！

## **挑战公司No.2：**上海复深蓝信息技术有限公司

**公司地址：**上海市徐汇区漕河泾开发区虹梅路2007号1号楼



**面试时间：**5月12日 14:00 PM.

**面试结果：**顺利砍下Android工程师，17K offer+1K补助 = 18K 月薪:victory:

**面试过程：**

14:00 到公司，前台MM给了一张面试登记表和一张iOS笔试试题，然后一位和蔼的大叔（技术大拿）将我带公司接待处，让我开始笔试。

14:20 笔试完成，笔试很简单，做完后把试题交给前台，前台帮我联系面试官。

14:25 跟一个年龄大概在25~30岁之间的年轻技术官进行了接近1个小时的面试（通过交谈阳哥感觉面试官应该是项目经理或者开发组组长）。

15:20 人事面试，人事面试后提前跟我说CTO会对我进行一个电话复试。

第二天晚上大概19:00，公司CTO对我进行了技术复试，总共持续了31分钟。电话复试题要比初试难的多了，姜还是老的辣。我把复试的对话录音保存下来了，作为以后上海黑马就业指导课程的案例讲解（内容何止“高能”，简直就是高达！）。

**疯狂的笔试+面试记录（前方内容“残酷”:funk:，请准备受虐心态前行）：**

**1. 笔试**

     笔试时出现了一个小插曲，和大家分享一下。阳哥笔试所在的接待区是一个半开放式的区域，共有3张桌子，每张桌子上都有一支签字笔。我先用第一张桌子，发现笔是坏的（郁闷），然后换到第二张桌子，发现笔还是坏的（心想：尼玛，这么霉，顺便汗一下），最后只能换到第三张桌子，你猜怎么滴，对，笔还是坏的（想要骂娘了，FUCK！）。没辙了，翻遍我的背包，自己也没带笔。难道企业是用这种方式考察一个应聘者的吗？这怎么办？

    我想去前台借吧，前台肯定有（不要去人事那里借，前台毕竟只是前台，不会管你太多的细节，人事可就不一样了，他们会认为一个面试者来比试不带笔是极不严肃的表现）。热心的前台MM很乐意地把自己的笔借给了我（心里一阵暖啊！这家必须拿下！）。

**为了方便大家更清晰的看到笔试题目，阳哥手录笔试题和答案，看你能做对多少？！**

Q：Android的四大组件有哪些？

A：Activity、Service、ContentProvider、BroadcastReceiver。

Q：请描述下Activity的生命周期？

A：onCreate、onStart、onResume、onPause、onStop、onDestroy、onRestart。

Q：如何将一个Activity设置成窗口模式？

A：将Activity的样式设置成：android:theme="@android:style/Theme.Dialog

Q：如何退出Activity？如何安全退出已调用多个Activity的Application？

A：调用Activity的finish方法可以退出当前Activity。可以自定义一个Application，在Application中声明一个成员变量ArrayList用于存放打开的Activity，当退出时遍历ArrayList，依次调用Activity的finish方法。

Q：请介绍下Android的五种布局。

A：LinearLayout、RelativeLayout、FrameLayout、RelativeLayout、TableLagout

Q：请介绍下Android的数据存储方式。

A：SharedPreference、XML、SQLite、文件系统

Q：DDMS和TraceView的区别。

A：DDMS 的全称是Dalvik Debug Monitor Service，是 Android 开发环境中的Dalvik虚拟机调试监控服务。TraceView是程序性能分析器

Q：说说Activity、Intent、Service以及他们之间的关系。

A：Activity负责界面的显示和用户的交互，Intent封装了数据，可以实现Activity之间以及Activity和Service之间数据的传递。Service运行在后台进程，一般我们会让给其运行一些后台任务，Activity通过StartService（Intent）或者BindService（Intent）可以启动Service。

Q：请介绍一下ContentProvider是如何实现数据共享的。

A：我们可以定义一个类继承ContentProvider，然后覆写该类的insert、delete、update等方法，在这些方法里访问数据库等资源。同时将我们ContentProvider注册在AndroidManifest文件中，其他应用需要使用的时候只需获取ContentResolver，然后通过ContentResolver访问即可。

**2. ROUND 1：PK项目经理技术面试问答精选**

**项目经理问：**Activity都有哪些生命周期？

**我（阳哥）答：**这个问题其实在笔试题中我已经给出答案了（此时，阳哥表示非常疑惑~）。

**项目经理说：**我知道，你再说一遍。

**旁白：**当时阳哥我真没搞明白，为何笔试上的题目他还是问我一遍，而且笔试上的好几道题他都重复问了一遍。现在想想应该是他怀疑我笔试的时候作弊了，比如我可以百度什么的。好吧，我不就是笔试的时候出去跟前台妹子借了一支笔，然后又把笔试题给拍了个照片而已嘛。而且，我那3G的联通定制机安装的移动4G的卡，只能享受2G的网速，我打开个百度都得几分钟。算了，不能和面试官较劲！我忍！

我（阳哥）答：Activity有以下生命周期回调方法，比如常用的有onCreate、onStart、onResume、onPause、onStop、onDestroy、onRestart。默认情况下，如果我们不给Activity设置横竖屏配置信息的话，在横竖屏切换时会将一个Activity销毁掉然后重新创建。

**项目经理问：**Fragment你用过吗？

**我（阳哥）答：**这个当然用过呀。现在的应用中基本都有Fragment的应用，Fragment比较小巧灵活。

**项目经理问：**那Fragment跟Activity之间是如何实现值传递的？

**旁白：**其实项目经理在问我会不会Fragment的时候，我已经预料到接下来会问我Fragment跟Activity直接的值传递问题。对于前面的问题，如果我说不会，就不会有下面的问题，但是如果说不会的话，那么项目经理很可能因为这一个问题而把我PASS掉，因为Fragment是Android中一个非常重要的知识，这个是必须会的（黑马的Android基础课程中有一天就是讲Fragment的）。如果连这个都不会，那么再好的面试技巧也挽救不了同学们的！

**我（阳哥）答：**Activity可以先获取FragmentSupportManager或者FragmentManager，前者是v4包下的，向下兼容因此用的比较多。然后这些Manager通过Fragment的tag或者id调用findFragmentByTag("tag")，findFragmentById("id")找到我们需要的Fragment对象，然后通过调用Fragment对象的方法来进行值的传递。

**项目经理问：**Android中都有哪些组件需要在清单文件中注册？

**旁白：**四大组件是学习Android的必会知识点，也是黑马Android基础中的重点内容，因此这个问题同学们其实应该是很好回答的！

**我（阳哥）答：**一般来说四大组件都需要在AndroidManifest.xml中进行注册，不过其中Activity、Service、ContentResolver是必须注册的，而BroadCastReceiver可以在清单文件中注册，也可以不注册，这也分别叫做静态注册和动态注册。

**旁白：**在Android中一般通过XML文件注册的组件，我们叫静态注册，而通过代码注册或者创建的组件我们叫做动态注册。动态注册和静态注册这两个名称听起来很高大山，其实理解起来so easy滴！

**项目经理问：**你自己有做过自定义控件吗？

**我（阳哥）答：**自定义控件做过，比如我们项目中的SlideMenu，LazyViewPager，Pull2RefreshListView，VerticalSeekbar,RandomLayout等都是自定义控件。

**项目经理问：**那你说说View的绘制过程？

**我（阳哥）答：**View绘制是从根节点（Activity是DecorView）开始，他是一个自上而下的过程。View的绘制经历三个过程：Measure、Layout、Draw。

**旁白：**黑马的老版本课程体系中有2天的自定义控件，在这两天课程中同学们能够学会SlideMenu，Pull2RefreshListView，优酷菜单等自定义控件，目前的黑马课程又对自定义控制进行了加强，添加了多种QQ5.0新特性内容，当然难度其实也不小。

**项目经理问：**ListView你们应该有用过吧？

**我（阳哥）答：**这个在我们的项目中应用的很多，其实在如今所有流行的App中，ListView都有一个大量的应用，说ListView是一个使用率高的控件都不为过。

**项目经理问：**ListView的优化你们是怎么做的？

**我（阳哥）答：**ListView的优化有多种多样的策略。在我们的项目中主要做了如下优化。1、重用ConvertView，2、给ConvertView绑定ViewHolder，3、分页加载数据，4、使用缓存。前两个是通用的解决方案，后两个是针对我们业务的个性化解决方案。我们的数据来自服务端，如果服务端有1000条数据的话，我们客户端不可能傻瓜式的一次性用ListView把这些数据全部加载进来，因此我们就用分页加载数据，每次加载20页，当用户请求更多的时候再获取更多数据，网络的访问就算网速再快也多多少少会有一定的延迟，因此我们的网络请求是异步处理的，同时从网络加载来的数据使用了2级缓存来处理，第一级是内存级别的缓存，第二级是本地文件的缓存。当ListView加载数据的时候首先从内存中找，如果找不到再去本地文件中找，只有都找不到的情况下才去请求网络。

旁白：ListView的优化是黑马课程一个重要的知识点，因此大家上课的时候这个必须得学会，不然在以后的面试中肯定会栽跟头的。

**3. ROUND 2：第二轮PK CTO（非人类）技术面试问答精选**

**旁白：**第二轮技术复试是在第二天晚上7点时开始的，当时，阳哥我刚从外面面试完回到家中。跟我聊的是他们公司的CTO，通过聊天也能感受到他的技术非同寻常，前几个问题问我时感觉多方有种咄咄逼人的气势（貌似面试的人太多了，对于被面试人员都是不屑的口气）。不过几轮技术PK后，发现原来CTO对人类也可以这么温柔和气的（尼玛，技术好才能被人看得起啊！心底话！）。

**CTO问：**说说你对泛型的了解？

**我（阳哥）答：**泛型是jdk5.0版本出来的新特性，他的引入主要有两个好处，一是提高了数据类型的安全性，可以将运行时异常提高到编译时期，比如ArrayList类就是一个支持泛型的类，这样我们给ArrayList声明成什么泛型，那么他只能添加什么类型的数据。第二，也是我个人认为意义远远大于第一个的就是他实现了我们代码的抽取，大大简化了代码的抽取，提高了开发效率。比如我们对数据的操作，如果我们有Person、Department、Device三个实体，每个实体都对应数据库中的一张表，每个实体都有增删改查方法，这些方法基本都是通用的，因此我们可以抽取出一个BaseDao<T>，里面提供CRUD方法，这样我们操作谁只需要将我之前提到的三个类作为泛型值传递进去就OK了。而数据的安全性，其实程序员本身通过主观意识是完全可以避免的，何况某些情况下，我们还真的想在ArrayList中既添加String类型的数据又添加Integer类型的数据。

**CTO问：**你知道Java的继承机制吗？

**我（阳哥）答：**知道呀，这个问题很简单呀！java是单继承多实现呀。

**CTO问：**那你知道java为何这样设计吗？

**旁白：**从上面的问题也可以看出越是资历老的程序猿越喜欢刨根问底，因此如果同学们面试的时候遇到一个年龄稍微大点的程序员，那么一定要提前做好思想准备了，他可能先问你一个看似很简单的问题，然后再追问一个很深的思想或者原理。

**我（阳哥）答：**为何Java这样设计，其实这也是我一直的一个小疑惑。不过我是这样理解的。我只能用反证法，如果一个类继承了类A和类B，A和B都有一个C方法，那么当我们用这个子类对象调用C方法的时候，jvm就晕了，因为他不能确定你到底是调用A类的C方法还是调用了B类的C方法。而多实现就不会出现这样的问题，假设A和B都是接口，都有C方法，那么问题就能解决了，因为接口里的方法仅仅是个方法的声明，并没有实现，子类实现了A和B接口只需要实现一个C方法就OK了，这样调用子类的C方法时，Java不至于神志不清。从另外一个方面考虑的话应该就是Java是严格的面向对象思想的语言，一个孩子只能有一个亲爸爸。

**CTO问：**Java的异常体系你知道吗？

**我（阳哥）答：**知道呀，顶层是Throwable接口，往下分了两大类，一个RunntimeException另一个是普通的Exception。

**CTO问：**那你知道这两类异常的区别吗？

**我（阳哥）答：**当然知道，java的命名是见名知意的。从名字上我们也知道RunntimeException就是运行时异常，在运行的时候才能被jvm发现导致程序的终止，而普通Exception必须进行try、catch处理，或者在方法上用throws声明。

**CTO问：**那你的期望薪资是多少？

**我（阳哥）答：**我期望的薪资已经给贵公司人事说过了，是17k。

**CTO问：**你这么年轻，就想要到17k呀！

**我（阳哥）问：**对的，我是还年轻，高中同样是学习3年，有的考上了重点大学，有的却只考上了个大专院校，甚至落榜，不同的人学习能力是完全不一样的，甚至可以用天壤之别来形容，因此如果只简单的用时间来衡量一个人的价值显然就是不太合理的，比尔盖茨跟我这样大年龄的时候已经是亿万富翁了，而我还在找17、18k薪水的工作（前面的问题已经回答的这么漂亮了，谈薪水的时候一定要表现出绝对的自信！）。

**CTO问：**你说的对，不过我还得问你几个问题，你说你们项目中有用到图片吗？

**我（阳哥）答：**这个当然有呀，我们新闻客户端基本上每条新闻都有图片，只有图文并茂的新闻才会有人看。

**CTO问：**那你说你们遇到OOM异常吗？

**我（阳哥）说：**这个前期的时候我们的APP确实遇到过这样的问题，不过现在新的版本早就吧这些问题给解决了。

**CTO问：**那你们是怎么解决的？

**我（阳哥）答：**OOM异常是Android中经常遇到的一个问题，程序员稍微不注意可能就导致其产生。因为Android的每一个应用都是一个Davlik虚拟机，该虚拟机的默认堆内存只有16M，远远无法跟我们的PC机比较，因此和容易导致OOM（Out Of Memory）异常的产生。导致这样的异常主要有以下原因：1、加载大图片或数量过多的图片，因为图片是超级耗内存的，2、操作数据库的时候Cursor忘记关闭，3、资源未释放，比如io流，file等，4、内存泄露。我们用用的OOM主要是加载图片导致的。因为后面的三种原因都是可以通过约束程序员的编码规范来进行预防，或者使用性能分析工具来检查。

**CTO问：**好的，那你们的图片是怎么处理的？

**旁白：**随后这就是CTO面试应聘者的一个习惯，他会追着一个知识点往死里问，直到问到系统的底层，或者他能理解的底。

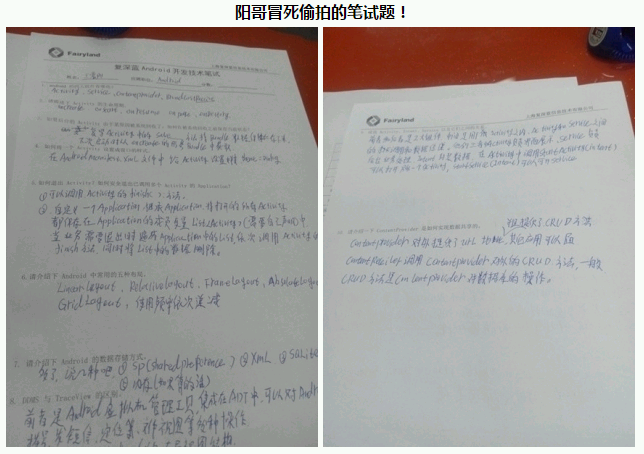
**我（阳哥）答：**图片的处理主要用两种方式。我们的应用中有两处用到了图片，一个是ListView中展示的图片缩略图，这种情况的特点是数量大，但是单个图片内存小，只有几kb，另外一种是大图片，就是用户通过手机拍摄的图片，然后通过http的post提交的方式提交到服务器上。然后在客户端将这个大图片也展示出来。对于第一种情形，我们是通过三种技术手段来解决问题的，一是图片的缓存策略，二是ListView的优化，其实在上面我已经讲过，三是WeakRefrence(弱引用)的使用。对于第二种情形，我们主要是首先通过BitmapFactory.Options参数获取图片的宽和高，然后再根据我们ImageView的宽高对图片进行一个很大比例压缩。

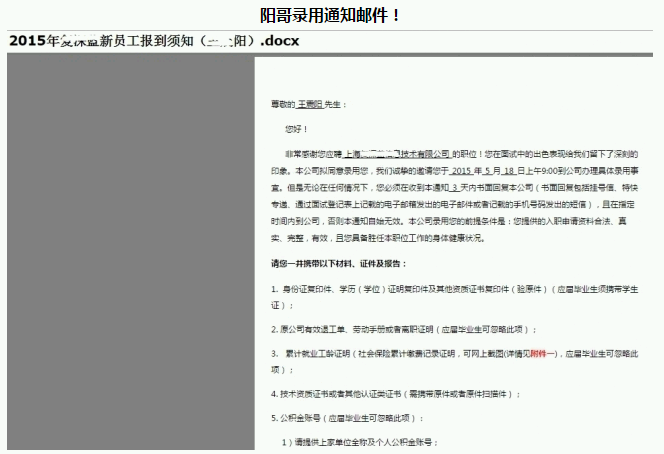
**CTO问：**那你说说弱引用是怎么使用的？

**我（阳哥）答：**WeakRefrence是一个类，在ArrayList中我们把这个类作为对象传递进去，把我们的图片放在WeakRefrence里面，这样当davlik虚拟机内存不够用的时候，就会把WeakRefrence对象回收掉，这样我们在WeakRefrence里面保存的数据也被回收了。

**面试吐槽：**

       阳哥在上海的第二面终于遇到懂技术的人，把笔试、技术一面、人事、技术二面一气呵成的通了个关，不拖泥带水，最终人事给了基本薪资17k+1k多补助的offer，这种感觉也许只有你经历过了才能体会吧！相信黑马学子经过四个月的刻苦磨练也能远远超过我的水平。加油吧，骚年们！





## **挑战公司No.3：**中阜投融资产管理股份有限公司---中投融

**公司地址：**上海市静安区威海路228号招商局广场南楼21楼



**面试时间：**5月12日 16:30 PM.

**面试结果：**顺利拿下Android工程师，15K offer+1k住房补贴=16k:victory:

**面试过程：**

16:30 到公司，因为这一天安排了3家面试，这家本来是安排的16:00的，但是迟到了半个小时，不过企业招人都比较急，这个可以理解。

16:30~16:40 填写面试登记表，这家没有笔试题，填完后坐在公司前台附件的沙发上等了几分钟，桌子上放了一盘水果糖，诱人，也不敢吃，哈哈。

16:40~17:15 跟一位Android程序员进行第一轮PK。

17:20~17:50 跟项目经理进行第二轮PK。

17:50~18:00 跟人事谈薪资。

**疯狂的笔试+面试记录（前方内容“高能”:funk:，请准备高度精神集中前行）：**

**1. 笔试**

   这一家没有笔试题，其实阳哥发现一个规律，一般Android开发团队刚刚组建的，或者Android开发人员不多的企业基本都没有笔试这一环节，这可能是他们公司的Android开发团队各方面还没有形成一套标准的流程原因吧。

**2. 第一轮技术面试过程问答精选：**

**旁白：**面试我的哥们儿比较腼腆，也许是他也刚进公司没多久的缘故，好像才4个月的样子。他主要问了我都做过哪些Android项目，怎么学习的Android，都做过哪些Android控件，然后他又问了一个他们公司目前正在做的项目遇到的一个难题，问我怎么解决。

**PS：**最后这个问题，阳哥感觉他并不是在考察我的技术，而是以请教我问题的态度在向我咨询了。面试能面到这种程度，基本已经有戏了。

**面试官问：**介绍下你都做过哪些Android项目？

**我（阳哥）答：**这自己主要做过3个项目，新闻类的，应用平台类的，手机管理类的，其中自己参与多也做的成熟的是新闻类的一款App。

**旁白：**上面的三种类型的项目，都是黑马Android课程体系中的教案，因此介绍项目基本没压力。

**面试官问：**那你在项目的开发中担任一个什么角色？

**旁白：**这样的问题其实很多面试官都问我了，感觉被问到的概率有80%，此种问题显然是想考察我们的担当能力，技术担当和责任担当。在上海黑马66期的开班典礼上，我是这样给大家说的，我们66期有70多位同学，我们班分成10个小组，每个小组选出一名组长，组长负责全组的学习，组长不是固定不变的，每周考试一次，每次考试成绩高者为组长。每次考试如果这个组的平均分在班排名低的，组长得无条件接受惩罚。在黑马，没有个人排名，只有团队排名，我们更注重团队的协同能力，而不注重个人的表现。

**我（阳哥）答：**我在这个项目中属于主要参与者之一吧，或者说是主要负责人，我们Android项目不像是javaweb项目那样需要大量的程序员，像我们的项目，总共2到3个程序员3个月功夫就能搞定，因此我们几个人也没有绝对的说谁领导谁。不过这样的项目，让我们任何一个人去开发都是很OK的，只是时间问题。

**面试官问：**你们用什么代码管理软件？

**我（阳哥）答：**SVN。

**旁白：**在黑马有专门的一节课是讲SVN和Git的，并且后面的项目也是用SVN跟大家进行代码共享的。

**面试官问：**你学习Android的途径都有哪些？

**我（阳哥）答：**现在是互联网时代了，不像我们大学时代主要通过书本学习。在大学的时候选修的Java课程，那时候还是诺基亚时代，图书馆的移动开发书架基本是被诺基亚的塞班占据的。自己大学毕业的时候Android已经开始风靡全国了，那时候自己从网络上看到的Android开发的各种视频，然后开始学习的Android。自己在Android的开发中遇到的各种问题一般都是靠百度解决的，说度娘是好老师真不为过，其实也用过Google，不过被屏蔽了，也懒得翻墙，百度现在做的也不差，百度出来的Android技术主要来自如下专业网站，比如：CSDN、51CTO、ITEye、AndroidBus、EOEAndroid等，对了还有一个国外的没有被屏蔽的网站是Github，我们项目中的很多控件其实都是从Github上学习过来的。

**面试官问：**那你从Github上都用到过什么控件？

**我（阳哥）答：**Github上的开源项目非常的多，基本上你想用的东西都有，比如xUtils、HelloCharts、Clander、SlideMenu、SeatTable、LDrawer、SmoothProgress、Touch Gallery、ViewPagerIndicator。

**面试官问：**你自己有做过自定义控件吗？

**我（阳哥）答：**自定义控件做过，比如我们项目中的SlideMenu，LazyViewPager，Pull2RefreshListView，VerticalSeekbar,RandomLayout等都是自定义控件。哦，对了，最近我刚给我女朋友还做了一个HideSlideBar控件，我可以让你看一看。里面主要由VerticalSeekBar+ProgressBar+NineOldAnimation完成的。

**旁白：**我把我自己做的自定义控件给他演示了一遍，这个是我女朋友项目中需要用的一个需求，她不会做，我帮她写了一个Demo，没想到正好用上了。

**面试官问：**哦，不错，你这个动画用的是属性动画吧？

**我（阳哥）答：**对的，这个属性动画，用了JakeWharton大神的开源框架。不过这个属性动画其实自己写也可以，内部原理比较简单，就是给控件设置一个开始位置，一个结束位置，设置一个延时时间，然后不停的更改控件的layout位置即可。

**旁白：**这个东西在黑马的项目中都有讲解，回答不难，在Android应用中我们会用到很多第三发的框架，我们不仅仅要会用别人的东西还必须知道别人写这个东西的内部原理。

**面试官问：**那好，现在我们有个项目遇到一个问题，你看如果让你做你应该怎么去解决？

**旁白：**这时候，阳哥已经感觉到自己已经PK成功了。不过面试官性格也不错，遇到问题善于寻求外界的任何帮助。

**我（阳哥）答：**可以的，什么需求您讲吧，我听听。

**面试官问：**我们的项目中有多个Fragment，Fragment A跳转到Fragment B，Fragment B跳转到FragmentC，那么这时候我多次按返回键，如何能让Fragment跟Activity的任务栈一样，依次从FragmentC跳转到FragmentB，再跳转到FragmentA？

**旁白：**这个确实是需要比较棘手的问题。其实面试官也知道你没有做过类似需求的开发，事实也是这样。这就比较考验临场发挥能力了。这种问题可以从模仿Activity任务栈考虑解决。任务栈是一种数据结构，我们可以自定义这种数据结构，然后管理这个数据结构，但是每个Fragment都是独立的，如何把这些独立的Fragment关联起来，这都是需要考虑的问题。

**我（阳哥）答：**这个需求应该很好实现。我们可以这样，首先定义一个BaseFragment，让FragmentA、FragmentB、FragmentC都继承BaseFragment，第二在BaseFragment中定义一个ArrayList，每打开一个Fragment，把这个Fragment对象添加到ArrayList中，这样这个ArrayList就可以当做一个栈结构，第三我们需要设置返回键监听，当监听到返回键的时候，查看当前ArrayList中倒数第二个Fragment有没有Fragment，如果有则取出该Fragment并把ArrayList中末尾Fragment删除，然后用FragmentManager的Replace方法，将当前Fragment替换成最新Fragment即可，如果ArrayList中只有一个Fragment，且监听到了返回键，那就不对Fragment做处理，同时也不拦截该事件，这样也不会影响其他Activity之间的切换。

**旁白：**回答了上面的问题后，面试官说他回去试试，然后就去叫他们的领导了。

**3. 第二轮技术复试过程问答精选：**

**旁白：**第二轮面试我的是个技术大拿，主要负责公司整个软件架构的设计，这家公司之前只有web端，全都是他干的，现在公司向移动端发展，因此最近一直在招Android和iOS开发人员，他不太懂Android，不然估计他一个人就能把一个Android项目搞定了。他问了我一些关于http、数据安全的知识，然后就开始跟我谈人生了，一直到他们快下班，他才把人事叫来跟我谈了谈薪资。

**经理问：**你做的Android的项目跟服务器交互都有哪些接口？

**旁白：**大家不要被接口这个词给迷惑了。这里的接口不是java中的接口类，也不是很高大上的东西，其实换成大白话就是你们的客户端跟服务端是如何实现数据的交互的。

**我（阳哥）答：**我们的接口有多种形式，第一种是http的形式，客户端跟服务器通过http协议传输数据，比如我们的新闻列表的请求都是给服务器发送的get请求，然后服务器把数据发给我们，我们上传给服务的图片是通过http的post请求方式完成的。第二种是socket完成的，服务端开启ServerSocket，客户端开启socket，然后客户端跟服务端建立长连接，这样实现了客户端跟服务端数据的即时通信，通信的协议是我们公司按照xmpp开放协议的基础上修改的，其实xmpp协议就是一个xml格式的数据。第三种是集成了第三方接口，比如分享功能用的是ShareSDK，消息推送用的是JPush，内置广告用的是万普世纪。

**经理问：**你们http传输数据的时候安全是怎么保证的？

**我（阳哥）答：**我们的数据有些是需要安全设置的有些不需要，我们的新闻类数据不需要特殊的添加安全设置，而用户注册，用户登录以及用户隐私数据保存是考虑安全性的。用户的密码等信息肯定不能进行明文传输的，我们将用户的密码在本地进行了MD5算法的加密，然后再传输。同时保存在本地的时候也是加密后的数据。还有需要安全性更高的数据需要通过我们自定义协议通过Socket传输。

**经理问：**那你知道MD5的原理吗？

**旁白：**老程序员就是喜欢刨根问底，如果技术功底不雄厚的话确实这个问题很难应付，因为对于大多数人只需要用一个技术就行了，不会去关注他的内部原理。

**我（阳哥）答：**MD5算以512位分组来处理输入的信息，且每一分组又被划分为16个32位子分组，经过了复杂处理后，输出由四个32位分组组成，将这四个32位分组级联后将生成一个128位散列值。这个过程是不可逆的，我们也把他叫做数据指纹，但是我们依然对用户的输入进行安全校验，如果是纯数字类型密码，是不允许注册的，因此就算你MD5加密了，黑客也可以通过彩虹碰撞的形式进行暴力破解。

**旁白：**接下来，经理问了我几个问题可能能难住我，关键是他也不懂Android，只能问我javaSE和思想上的一些东西，之后就开始跟我聊人生了，说他们公司是国企背景，自己在公司发展多么好，公司未来要做一个什么什么样的产品，公司弹性工作制，每年至少14薪，又问我女朋友在哪，什么时候考虑结婚，在哪买房，老家是哪的，爸妈是干什么工作的，你是如何规划未来的等等。。。。。

**4. 人事面试：**

这家的人事面试与其说是人事面试还不如说成人事咨询，人事跟我介绍了公司背景，跟我谈了薪资，跟我谈了入职安排，又谈公司发展，公司的福利待遇等等。人事的最后一个问题都是出奇的相似，

**人事：**我该问你的问题都问完了，你还有什么要问我的吗？

**PS：**这些问题千万不要一个都不问，这样人事会认为你这个应聘者对我们公司不感兴趣。如果有其他人也去面试，那么你就可以能被PASS了。其实我们真的想去这家公司的话，肯定会有一大堆问题的，但是也不要问的太多，问的太多显得你很啰嗦，人事的耐心也是有限的，同时也不要问太低级的问题。对于我们程序员来说，应该最关注如下两大类问题，一是自己将要加入的开发团队的情况，二是公司的薪资待遇以及员工培训发展情况。第一类问题显得我们技术专业，技术屌丝的特性被发挥的淋淋尽致，第二类问题很现实，我们不仅要眼前的工资还要考虑未来的发展。

**我（阳哥）答：**咱们公司的技术团队目前有多少人，都做哪些项目？

**人事：**公司目前技术团队还在不停的扩建，目前总共有十几名，不过服务端人比较多。现在主要做p2p理财类产品，这也是很火的一个趋势。

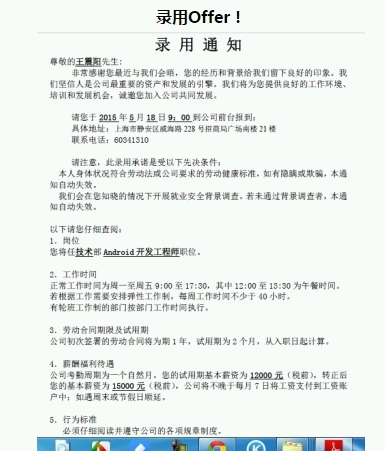
**我（阳哥）答：**咱们公司给开发人员有定期的培训吗？

**人事：**有的，公司每年都有拓展培训，拓展培训是针对全体员工的，技术团队的话每周或者每个月可能有技术分享活动，每周一名技术人员进行分享，同时可以获100元/次的奖励。

**面试吐槽：**

       阳哥在上海第一天竟然面试了3家，上海这么大，每个公司都在不同的区，回到家已经晚上7点多了，我的感受只有一个字儿累，躺在床上就起不来了，不过还好今天本来是抱着被面试官“虐”的心态去的，结果没有被“虐”，而且还能旗开得胜。这给了我很多自信，不仅仅是对自己技术的自信，更是对黑马Android课程的自信。

最后送大家一句话：没有企业给不了的薪资，只有自己掌握不了的技术。没有自己掌握不了的技术，只有不够努力的自己。



## **挑战公司No.4：**上海游竞网络科技有限公司---PLU

**公司地址：**上海市闸北区广中西路777弄99号江裕大厦10层



**面试时间：**5月13日 14:00 PM.

**面试结果：**Android工程师，16K offer+500元住房补贴+490元交通和饭补+200元全勤奖=17.1k:victory:

**面试过程：**

16:00到公司，这家公司在闸北区，比较远，不过办公楼很高大上。

16:00~16:30 填写面试登记表和笔试题。

16:30~17:15 跟面试官进行技术PK（这家面试流程比较简洁，技术只考察一轮就让人事谈薪资了）。

17:20~17:40 跟人事谈人生谈薪资。

**疯狂的笔试+面试记录（前方内容“高能”:funk:，请准备高度精神集中前行）：**

**1. 笔试**

**PS：**笔试是在公司前台旁白的桌子上做的。阳哥本来胆子就小，这一下搞的偷拍都没自信了，导致对焦不成功，拍出来的照片很模糊，大家凑合着看吧，我把图片上的试题以文本的形式列出来，如下。

**Q：**Activity的生命周期？

**PS：**我晕，跟复深蓝的笔试题如此的雷同，难道他们两家公司技术人员互相抄袭的吗？忍住，不笑！

**A：**onCreate、onStart、onResume、onPause、onStop、onDestroy、onRestart。

**Q：**Activity销毁前，如何保存Activity的状态？

**A：**可以使用onSaveInstanceState（Bundle）方法将Activity中需要的数据保存起来，当下次重新启动Activity的时候在onCreate（Bundle）中获取Bundle数据。

**Q：**请介绍下Android中常用的5种布局？

**A：**LinearLayout、RelativeLayout、FrameLayout、RelativeLayout、TableLagout。

**Q：**请介绍下Android的数据存储方式？

**A：**SharedPreference、XML、SQLite、文件系统、内存（如果算的话）。

**Q：**AIDL的全称是什么？如何工作？能处理哪些类型的数据？

**A：**

① Android Interface Definition Language

② AIDL一般用于远程服务，也就是进程间通信。我们可以分服务端和客户端，服务端声明AIDL文件，该文件命名为xxx.aidl,ADT会自动将xxx.aidl生成代码文件，代码文件提供了aidl中接口的实现。客户端如果要使用服务端提供的服务需要将xxx.aidl文件放到客户端源代码目录下，然后生成xxx.java类，客户端通过bindService的形参ServiceConnection的onServiceConnected获取到Service对象，这个对象通过Stub.asInterface（service）返回aidl的实现类。之后我们就可用调用这个aidl的实现类。

③ 基本数据类型都可以，复杂对象也可以，只不过需要实现Parcelable接口。

**Q：**请介绍一下handler机制？

**A：**Android中handler多用于主线程和子线程之间的通信，比如在Android中子线程是不允许修改UI的，如果修改只能让子线程给主线程通过handler发送message，然后主线程进行修改。Handler整个机制的实现，还依赖Looper、Message两个核心内容。在主线程中Android默认给我们创建了Looper

**Q：**java如何调用c、c++语言？

**A：**java通过JNI调用C/C++代码，在使用的时候首先通过System.loadLibrary("xxx")将xxx.so文件加载到jvm中，同时在类中必须对so文件中的方法进行生命，格式:public native void test();

**Q：**Android分几层，分别是什么？

**A：**四层。Linux Core、Libraries（Android Runtime）、Application Framework、Applications。

**Q：**final、finally、finalize的区别？

**A：**第一个是关键字最终，用final修饰的类为最终类，不能被继承，修饰的方法不能被覆写，修饰的变量不能被改变。finally是异常体系中的关键字，当系统遇到异常是，在进行trycatch的时候，finally代码块里的代码是必须被执行的。finalize是Object类中的方法，当GC回收对象时回调的方法。

**Q：**heap和stack的区别？

**A：**堆和栈。栈存放对象的引用，堆存放对象实体。堆中的对象是有jvm的垃圾回收器负责回收。

**2. 第一轮技术面试过程问答精选：**

**旁白：**面试我的是85年出生的Android组组长（这是后来他们公司人事给我打电话让我入职的时候，跟我说的）。他们公司是做游戏视频直播平台的，因此对app的性能要求比较高，他用手机让我看了一段代码。代码（记得不是很清楚了）大概如下：

1. public class MainActivity extends Activity {
2. @Override
3. protected void onCreate(Bundle savedInstanceState) {
4. super.onCreate(savedInstanceState);
5. setContentView(R.layout.activity\_main);
6. new Thread(new Runnable() {
7. @Override
8. public void run() {
9. while (true) {
10. SystemClock.sleep(1000);
11. }
12. }
13. }).start();
14. }

**面试官问：**上面的代码有问题吗？

**我（阳哥）答：**当然有问题呀，这个Activity在启动的时候开启了一个子线程，但是当Activity退出的时候该子线程还在运行，并没有停止。子线程运行在进程中的，Activity退出的时候进程并没有退出，而由前台进程变为后台进程。

**面试官问：**那应该怎么解决？

**我（阳哥）答：**我们可以这样，把while（true）改成while(flag)，flag是一个boolean类型的变量，通过改变flag的true或者false来终止子线程的运行，当Activity退出时会调用onDestroy方法，因此在该方法中我们可以把flag设置为false。

**旁白：**面试官听了我的答案，从他的表情上来看好像他不是很满意。但是他也没说我说的对不对，而是给我说了他心中的答案，不过他给的答案我当时是虚心接受了，不过到现在我还没明白他说的道理在哪，我写了测试代码也没测试出来啥。

**面试官问：**这个代码有问题就出在这个Thread是一个匿名的，而且没有声明为静态成员变量？

**我（阳哥）答：**哦，这样子呀，这个我真不知道。

**旁白：**其实我内心是很想问他为什么呢？但是如果他答不上来就会让他很难堪，如果回答上来了倒没啥问题。从他也不自信的言语中我感觉还是不问他比较好？

**面试官问：**你对RTSP流媒体协议有了解吗？有没有做过手机播放器的应用？

**我（阳哥）答：**这个知道，自己做过流媒体播放器。我使用过开源的Vitamio开源库。

旁白：黑马课程中有一个项目是手机影音。因此这个东西自己是知道的。不懂的人可能以为RSTP、ffmpeg这些专业名词都多么的高大上，其实在黑马手机影音中，我们会做一个视频播放器，视频播放器分两种，一种是Android自带解码，第二种是万能播放器，所谓万能播放器就是绝大多数流媒体格式都支持。

**面试官问：**那你知道代码测试怎么测？

旁白：关于测试，阳哥真心不太懂，怎么办？？尽量回答一些自己知道的，同时把测试方面的话题给关闭掉，省得面试官追问我。

**我（阳哥）答：**您说的是Monkey Test吗？。

**面试官问：**恩。。。也算吧？还有其他吗？

**我（阳哥）答：**我们程序员在写代码的时候都是有规范的，优良的代码规范是规避bug重要的一步，同时我们公司每周都有一个代码走读会议，所谓的代码走读，就是开发人员互相看对方的代码，检查代码是否规范以及是否存在bug。当然也有测试，不过我们的测试都是黑盒测试。

**面试官问：**哦，代码测试，就是可以通过测试一段代码来分析这段代码有没有问题，性能是否可靠。

**我（阳哥）答：**对的，Android自带了AndroidTest功能，可以对一个类一个方法进行测试。

**面试官问：**那好，图片你有处理过吧？

**我（阳哥）答：**这个肯定有呀，哪个Android应用没有图片。比如图片缓存呀，图片缩放呀，图片缓存呀。

**旁白：**其实自己对图片处理这块儿，还是比较了解的，可惜他也没怎么深问我，没有给我很大的发挥机会。

**面试官问：**我们现在要做一个视频在线播放的app，播放的都是游戏直播或者录像视频，iOS已经做好了的，Android还没有做好，你要是进来的话你就是做视频直播这一块儿。

**我（阳哥）答：**这个已经做好的能让我看一下吗？

**旁白：**面试官拿来水果机让我看了一下，主界面就是一个ListView，每个ListView的一行都有大概4个视频预览图，点击之后可以播放。

**我（阳哥）问：**这个切图，设计啥的都做好了吧，剩下的就是编码了其实？

**面试官答：**是的，后期我们Android团队还会不停的夸大，我们的发展重心已经偏向移动端。

**旁白：**接下来就是跟面试官侃大山了，从天文到地理，从BAT到初创型公司，阳哥也不能甘拜下风。

**我（阳哥）答：**恩，现在是移动互联网时代了，移动端肯定得做好，不管什么样的公司都特别重视移动端。咱们公司现在才开始研发移动端，其实已经稍微有点儿晚了，就得赶紧拼命追赶了。

**3. 人事面试：**

跟面试官聊完后，技术官对我的评价是，我挺喜欢你，你跟人事好好聊聊吧，哈哈，我肯定会跟人事MM好好聊天的。人事问的问题太老套路了，感觉全上海人事问的问题都一样，难道他们都是从一个地方培训出来的？人事也有培训吗？其实应付人事的问题不难，难的是如何回答人事的问题。答题的方式直接决定了我们在人事那里的印象。我把人事问的几个问题大概列一下。

**人事：**你为什么离职来上海？

**PS：**人事问这个问题的主要目的就是看你的动机纯不纯，有没有不良倾向。

**我（阳哥）答：**主要是两个方面吧，第一个是我的女朋友在这边工作，我年龄也不小了，也到了谈婚论嫁的时候了，我不着急，家里父母亲着急呀，来上海工作的话离女朋友比较近，结婚就比较好办，第二个主要还是。。怎么说呢。。。高大上的说就是为了自己事业更高的发展吧，说的直接点就是上海毕竟国际化都市，发展机会很多，自己也想来上海工作个3~5年挣钱买个房子首付啥的，如果在上海的这几年发展的好可以考虑再公司附近买房，如果自己发展的不好，在上海周边买个房也行。

**人事：**你女朋友一直在上海，还是刚回来上海？。

**我（阳哥）答：**她上一年来的这边，之前是我大学同学，主要是她家有亲戚帮她找了一份设计院的工作，而我只能靠自己找工作。

**人事：**你以后打算长期留在上海了吗？

**旁白：**上面的问题其实是人事在考察你的稳定性，公司可不想招到一个人刚培养出来就走掉了。但是如果你直接肯定的回答：我会一直留在上海，又显得很假。“装”的高境界就是让对方感觉你很不会“装”。

**我（阳哥）答：**这个有想过，自己其实蛮喜欢上海的，可是计划赶不上变化，来上海的前一年我也从来没想到我会来，因此不能说一辈子都在上海，不过目前这几年是打算在上海好好发展一下。上海的房价、户口政策都是问题，现在不是我选不选择上海，而是上海选择不选择我的问题，呵呵，其实谁都不傻，都想来好地方，关键就看自己有没有本事扎根于此了。  
**人事：**你能承受加班吗？

**旁白**：其实互联网企业加班是很常见的现象，尤其是像上海这种生活节奏比较快的城市，加班肯定是避免不了的，但是我们也有我们的底线，不能无条件无休止的为公司加班，但是也不能一点儿班都不能加，毕竟有时候公司忙的话，比如新产品上线，新bug的解决都是需要加班的，这个时候我们还真的义不容辞的为公司付出，毕竟我们的薪水是公司给的。在公司工作一方面是我们为公司付出，另一方面我也要知道感恩公司。

**我（阳哥）答：**加班很正常，自己之前的公司也经常加班。不过公司加班一般都不会让员工平白无辜的加班的，我们加班的时候公司会发晚餐补助，打车补助等，如果加班超过半天会计入到我们的存休中，等活不忙的时候我们可以申请休息。因此只要不是高频度的天天加班到凌晨应该没啥问题的。

**人事：**你的五年规划是啥？

**我（阳哥）答：**自己还是想往技术方向发展，自己也比较喜欢代码，愿意去研究技术。人往高处走水往低处流，经过自己几年的积累后自己想当一名技术总监或者项目经理类的技术管理类岗位。不过这些都得靠自己的努力以及对机遇的把握情况了。

**面试吐槽：**

       这家企业各个方面的条件真心不错，想拒绝都很难找到合适的理由！我要了15k的薪资，人家给了16k，还有各种诱人的福利。感觉还是游戏类的公司土豪。你以为你要15k，人家给你16k已经很高了吗？请让我把话说完，告诉你啥叫游戏公司。该入职的时候，当然我没有去入职（入职是上午10:00），上午10:20的公司人事给我打电话，问我迟到了还是怎么会儿事儿，我说对不起，感谢贵公司对我的认可，自己已经有了新的选择。把他们给拒绝了。下午1点多的时候，我正在吃饭，结果这家公司的技术官又给我打电话来了，在电话里跟我聊了很久，他说如果再给你多几k你会考虑过来上班呢？我哑巴了，自己从来就没有享受过这么好的待遇，受宠若惊的阳哥无言以对！自己的内心像打翻了五味瓶一样，这种感觉估计也没有几个人理解。后来还是被我给拒绝了，内心在流泪！

   这篇文章阳哥已经写到尾声了，但是想给大家说的话却依然很多很多。

   亲爱的黑马同学们：

           如果你还在撸着代码，看着视频，再苦再难，请你一定要坚持，今天你付出的一点一滴都将会在明天变成千千万万倍的回报。

           加油&坚持！

## **挑战公司No.5：**一号店旗下壹药网

**公司地址：**上海市浦东新区碧波路572弄115号10幢



**面试时间：**5月14日 10:00 AM.

**面试结果：**Android工程师，15K offer&14个月薪资:victory:

**面试过程：**

10:10到公司，这就是传说中的一号店！，公司总共三栋独栋楼，两个在装修，一个在用，因此工位紧张，导致我面试都是在前台所在的大厅里进行的。

10:20~10:50 填写面试登记表和技术官面试。

10:50~11:10 跟研发部Leader面试。

11:10~11:30 跟人事谈薪资以及入职事宜。

**疯狂的笔试+面试记录（前方内容“高能”:funk:，请准备高度精神集中前行）：**

**PS：**笔试跟面试是一起的，一个技术官拿着一张正反面都写满题的A4纸，从第一题到最后一题，他指一题我回答一题，我回答一题，他指下一题。配合的天意无缝，哈哈~~~

**1. 笔试**

**1）java基础部分**

**Q：**Java面向对象有哪些特征？

**A：**封装、继承、多态。

**Q：**short s1=1;s1=s1+1有什么错？

      short s1=1;s1+=1;有什么错？

**A：**第一个是有错的，short在内存中占2个字节，而整数1默认为int型占4个字节，s1+1其实这个时候就向上转型为int类型了，因此第一行代码必须强转才行。第二个之所以可以是以为这句话翻译过来就是s1++，也就是short类型的数据自身加增1，因此不会有问题。

**Q：**静态成员类、非静态成员类有什么区别？什么是匿名内部类？

**A：**静态成员类相当于外部类的静态成员，是外部类在加载的时候进行初始化，非静态成员类相当于外部类的普通成员，当外部类创建对象的时候才会初始化。匿名内部一般都是在方法里面直接通过new ClassName(){};形式的类。比如我们new Thread（new Runnable(){}）.start()；就用到了匿名内部类。

**Q：**abstract class 和 interface有什么区别？

**A：**前者是抽象类，可以有抽象方法，也可以没有。后者是接口，只能有抽象方法。他们都不能创建对象，需要被继承。

**Q：**ArrayList是不是线程安全的？如果不是，如何是ArrayList成为线程安全的？

**A：**不安全的。可以使用Collections.synchronizedList(list)将list变为线程安全的。

**Q：**是否可以继承String类？

**A：**不可以，因为String类是final类。为啥不解释了吧。

**Q：**以下两条语句返回值为true的有：

     A："yiyaowang"=="yiyaowang";

     B:   "yiyaowang".equals(new String("yiyaowang"));

**A：**第一个返回true，都是字符串常量，存储在字符串常量池中，且只有一份。第二个返回true，用equals比较的是字符串内容。

**Q：**当一个对象被当做参数传递到一个方法后，此方法可以改变这个对象的属性，并可返回变化后的结果，那么这里到底是值传递还是引用传递？

**A：**java中只有值传递，没有引用传递。这里的引用本身就是值，传递的是引用这个值。

**Q：**定义类A和类B如下：

1. class A{
2. int a =1;
3. double d = 2.0;
4. void show(){
5. System.out.println("Class A:a="+a+"\td="+d);
6. }
7. }
8. class B extends A{
9. float a = 3.0f;
10. String d = "Hello World!";
11. void show(){
12. super.show();
13. System.out.println("Class B:a="+a+"\td="+d);
14. }
15. }

(1)若在应用程序的main方法中有以下语句：

    A a = new A();

    a.show();

    则输出结果是？

(2)若在应用程序的main方法中定义类B的对象b:

    A b = new B();

   b.show();

   则输出结果是？

**A：**第一个是ClassA ：a=1     d=2.0

     第二个是ClassA ：a =1    d=2.0

                    ClassB : a=3.0   d=Hello World!

**Q：**heap和stack有什么区别？

**A：**堆和栈。栈存放对象的引用，堆存放对象实体。堆中的对象是有jvm的垃圾回收器负责回收。

**Q：**请描述下JVM加载class文件的原理机制。

**A：**JVM加载class是动态性的，也就是当“需要”的时候才会加载，这是也是为节约JVM内存来考虑的。同时JVM的类加载是父类委托机制，这个机制简单来讲，就是“类装载器有载入类的需求时，会先请示其Parent使用其搜索路径帮忙载入，如果Parent 找不到,那么才由自己依照自己的搜索路径搜索类”。

**2）Android基础部分**

**Q：**如何适配不同屏幕分辨率的机型？

**A：**屏幕适配方式就多了去了。Android系统本身提供了很多适配方法，比如存放图片资源的drawable目录根据不同分辨率的手机提供了drawable-hdpi、drawable-ldpi、drawable-mdpi、drawable-xhdpi等多中目录。我们只需把适应不同分辨率的多套图片分别放到对应的目录中即可。Android的layout、values目录也提供了类似drawable的适配功能。

     但是在开发中，不可能针对不同的手机分辨率提供多种图片资源，这太耗费资源了。我们一般在写控件宽高的时候都会用dp单位取代pix单位。因为dp是一个相对单位，pix是绝对单位，使用dp替代pix也可以解决很多适配问题。dp跟pix之间可以通过公式进行转换。

**Q：**View和ViewGroup的关系是什么？View的绘制过程（主要方法）有哪些？

**A：**ViewGroup继承了View。onMesure、onLayout、onDraw。

**Q：**Activity和Task的区别及启动模式有哪些？

**A：**Activity运行Task中。Activity有四种启动模式。standard、singleTop、singleTask、singleInstance。standard:默认的启动模式，多个Activity位于同一Task中。singleTop，顾名知义就是Task栈顶只能有一个相同的Activity，singleTask就是一个Task中只有一个Activity，singleInstance就是一个Activity独享一个Task。

PS：关于Activity的四种启动模式其实还有更详细的说法，我在上面就简单介绍一下了，如果面试官需要问的更详细再往深处介绍就行了。

**Q：**如何注册BroadCastReceiver和Service？Service有什么特征，哪些情况会用到Service？

**A：**都可以通过AndroidManifest.xml进行静态注册。不过BroadCastReceiver可以在代码中通过registReceiver方法来注册。Service运行在后天进程中，一般需要在后台一直运行的任务会让Service来完成。比如我们的telephoneService、locationService等等。

**Q：**Android有哪些安全机制？

**A：**权限机制。我们的应用只要涉及到了用户的隐私、网络都需要在AndroidManifest.xml中进行声明，这样用户在安装的时候可以根据你申请的权限进行判断是否允许应用的某些行为。

**Q：**Handler机制的原理，内部是如何实现的？

**A：**Android中handler多用于主线程和子线程之间的通信，比如在Android中子线程是不允许修改UI的，如果修改只能让子线程给主线程通过handler发送message，然后主线程进行修改。Handler整个机制的实现，还依赖Looper、Message两个核心内容。在主线程中Android默认给我们创建了Looper,当我们通过handler.sendMessage()后，该消息被添加到MessageQueue中，Looper.looper中有个while(true)的循环不停的从消息队列中取消息。取消息的过程是线程阻塞的，这样不至于在没有消息的时候过多的耗费CPU资源。

**Q：**Thread和AsyncTask的区别是什么？

**A：**AsyncTask是封装好的线程池，比起Thread+Handler的方式，AsyncTask在操作UI线程上更方便，因为onPreExecute()、onPostExecute()及更新UI方法onProgressUpdate()均运行在主线程中，这样就不用Handler发消息处理了；

**Q：**说说MVC模式的原理，在Android中的运用。

**A：**MVC是Model、View、Controller三部分组成的。其中View主要由xml布局文件，或者用代码编写动态布局来体现。Model是数据模型，其实类似javabean，不过这些JavaBean封装了对数据库、网络等的操作。Controller一般由Activity负责，它根据用户的输入，控制用户界面数据的显示及更新model对象的状态，它通过控制View和Model跟用户进行交互。

**Q：**如何加载ndk库？如何在jni中注册native函数，有几种注册方式？

**A：**通过System.loadLibrary("xxx")进行加载。其实native有几种注册方式，自己当时并不知道，自己只知道一种注册方法，就是首先根据native 方法名，生成Java\_com\_xxx\_MethodName(xxx,xxx);当然这个c/c++源码文件中需要引入jni.h，然后把这个c/c++源码编译成so文件。

自己后来百度了一下，网上有人数还有一种注册方式是动态注册，我就把关于动态注册的东西直接拷贝过来：

JNI 允许你提供一个函数映射表，注册给Jave虚拟机，这样Jvm就可以用函数映射表来调用相应的函数，就可以不必通过函数名来查找需要调用的函数了。Java与JNI通过JNINativeMethod的结构来建立联系，它在jni.h中被定义，其结构内容如下：

1. typedef struct {
2. const char\* name; //Java中函数的名字
3. const char\* signature; //用字符串描述的函数的参数和返回值
4. void\* fnPtr; //指向C函数的函数指针
5. } JNINativeMethod;

第一个变量name是Java中函数的名字。

第二个变量signature，用字符串是描述了函数的参数和返回值

第三个变量fnPtr是函数指针，指向C函数。

当java通过System.loadLibrary加载完JNI动态库后，紧接着会查找一个JNI\_OnLoad的函数，如果有，就调用它，

而动态注册的工作就是在这里完成的。

1)JNI\_OnLoad()函数

JNI\_OnLoad()函数在VM执行System.loadLibrary(xxx)函数时被调用，它有两个重要的作用：

指定JNI版本：告诉VM该组件使用那一个JNI版本(若未提供JNI\_OnLoad()函数，VM会默认该使用最老的JNI 1.1版)，如果要使用新版本的JNI，

例如JNI 1.4版，则必须由JNI\_OnLoad()函数返回常量JNI\_VERSION\_1\_4(该常量定义在jni.h中) 来告知VM。

初始化设定，当VM执行到System.loadLibrary()函数时，会立即先呼叫JNI\_OnLoad()方法，因此在该方法中进行各种资源的初始化操作很恰当，

2)RegisterNatives

RegisterNatives在AndroidRunTime里定义

syntax:

jint RegisterNatives(jclass clazz, const JNINativeMethod\* methods,jint nMethods)

**Q：**App在什么情况下会出现内存泄露？如何避免这些情况？

**A：**造成内存泄露的可能性有很多，我说几种吧，1）资源未及时释放，比如引用的io流资源、网络资源、数据库游标Cursor等没有释放2）注册的监听器、广播等未及时取消3）集合对象没有及时清理4）不良代码

    避免上述问题，主要还看程序员知识掌握的程度和编码经验的多少，但是从技术角度考虑我们需要注意一些细节，比如，重复使用的资源可以考虑使用缓存技术、池技术。使用的任何资源都记得关闭或者异常处理，保证在恶劣的情况下也能使资源得到释放。对于图片的操作要注意缓存的使用，同时要记住对图片对象进行及时的回收。使用ListView的时候，尽量让ConvertView得到复用。

**3）逻辑思考**

**Q：**你让工人为你工作7天，给你工人的回报是一根金条，金条评分成7段，你必须在每天结束时给他们一段金条，如果只许你两次把金条弄断，如何给你的工人付费？

**PS：**因为上面的问题自己回答的都比较溜，所有这个逻辑思考题，面试官直接说我想着你也会就不做吧。哈哈~面试官已经放弃继续考察我了，嗨皮~~~然后他去找老大去了，趁机我赶紧把笔试题给偷拍了下来（是不是阳哥胆子越来越大呀？！），这样才能大家看到真实的笔试题是什么样子滴。

哦，对了，上面的逻辑思考题我面试的时候是免试的，亲爱的黑马童鞋们你们知道答案吗？

**2. 第二轮面试过程问答精选：**

**旁白：**面试我的应该是开发组部门的老大。胖胖的，黑黑的，矮矮的，从开始面试到结束，一直面带微笑。因为之前那个技术官已经面试过我的技术了，因此他也没问我技术，就是跟我瞎聊了一些关于Android方面的东西。因此详细内容这里就省略了，请大家直接进入下一关~duang~

**3. 人事面试：**

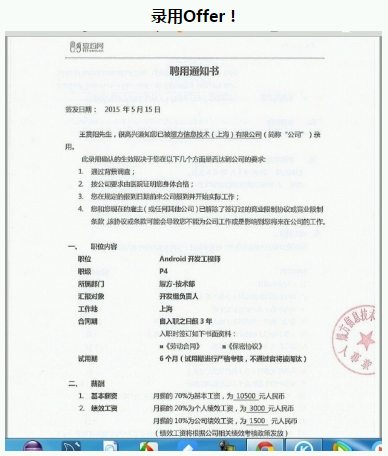
    人事面试的问题，其实我在V4版本中也说过，基本所有的人事问的问题都大同小异，跟这个人事聊的唯一有趣的就是，由于公司正在装修，找不到面试我的办公地点，然后她竟然把我带到公司旁边的凉亭子下面，我们两个并排坐着，就聊起来了，知道的知道我们是在面试，不知道的估计还以为我们在谈恋爱呢。:#

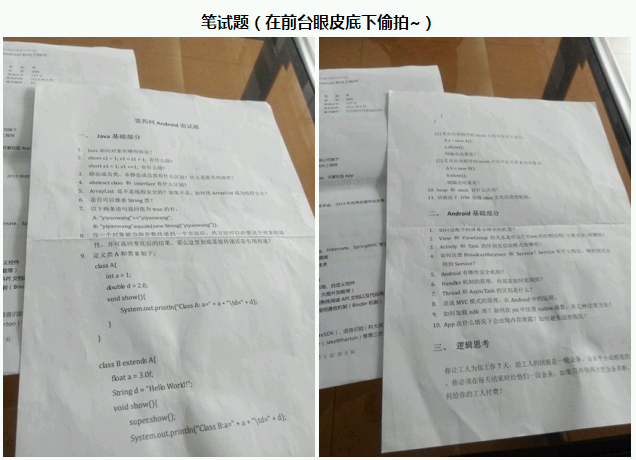
**面试吐槽：**

       在前台我填写面试登记表的时候写的期望薪资是15k，他们也没有压低我工资，我知道我要少了。不过，没办法，谁让阳哥不懂行情，在投递简历的时候都是写着期望15k呢！想写个16、17、18都不好意思了，因此大家以后找工作可别学阳哥这么傻，能多要一定多要。

   跟大家分享一个励志语句以结束本篇文章吧：

***相信梦想是价值的源泉，相信眼光决定未来的一切，相信成功的信念比成功本身更重要，相信人生有挫折没有失败，相信生命的质量来自决不妥协的信念。***





## **挑战公司No.6：**聚信租赁

**公司地址：**上海市徐汇区漕溪北路398号汇智大厦28楼



**面试时间：**5月14日 14:00 PM.

**面试结果：**Android工程师，16K +入职送iPhone 6+每年出国旅游一次+补充住房公积金+至少14薪:victory:

**面试过程：**

14:00到公司前台，领了一大堆资料，然后让我在一间办公室做题。

14:00~14:30 填写面试登记表和性格测试。

14:30~15:20 两个技术官坐在我对面，同时面试我。

15:20~15:30 人事妹子跟我单聊公司薪资福利以及入职事宜。

**疯狂的笔试+面试记录（前方内容“高能”:funk:，请准备高度精神集中前行）：**

**1. 笔试**

    这家比较奇怪，是给了6页的笔试题，打开一看全是性格测试题。性格测试题只拍了一张图片，在该篇文章的结果。估计大多数童鞋都想遇到阳哥这样的狗屎运吧;P~阳哥的性格绝对没问题的，这个我可以保证哈~:P

**2. 技术面试**

    竟然同时来了两个技术官面我，一大一小，一高一矮，并排坐在一起，一对二，好吧，阳哥还是首次遇到1:2的阵容，这下可有好戏看了。**PS：**阳哥是个没有见过世面的人，面试的时候上个31层高的楼都兴奋的不得了，上去以后，发现上错楼了~糗事说多了都是泪:#，要不是阳哥有着过五关斩六将身经百战的成功经验，估计要吓尿:loveliness:。好吧开始吧，阳哥命中注定必有次劫，看来是躲不过了。

**Q：**你做一个自我介绍吧？

**旁白：**其实遇到好几家面试官都让我做自我介绍了，该如何自我介绍阳哥估计都会背了，好玩（恶心）的是在万达信息面试，面试了3个技术官，每个人都分别让我做了自我介绍，尼玛，他们3个就不会沟通一下要问我啥吗，一个问题至于问我3遍吗~:funk:阳哥是敢怒不敢言，毕竟在人家的地盘。

**PS：**自我介绍的内容就不说了，每个人都是独特的，我就跟大家说一下应该如何自我介绍吧。

     一个优良的自我介绍会给面试官留下深刻的印象，大部分情况下，所谓的面试好坏其实看的就是你给面试官留下的印象怎么样了，我们用俗语叫感觉。

     自我介绍应该分以下几个部分，按照一定的逻辑连贯起来。如果连贯不起来，或者不够熟练一定在台下多背几遍，多讲几遍，但是面试的时候不要说的跟背过似的，高境界就是让面试官感觉你是临场发挥的，却又比背的都好。

     1）个人基本信息（姓名、年龄、老家、居住地等）

     2）自己来自哪里（工作地点），是干什么的（给自己一个清晰的定位，比如：我是一名Android开发工程师），担任过什么职务、做过什么样的项目

     3）自己为何来贵公司面试

     4）最后祝愿（希望能得到贵公司的认可等等，不用太多，一两句话就ok）

**Q：**介绍一下你做过的项目吧？

**PS：**黑马那么多项目，随便准备3个就ok了。

    介绍项目大概的思路如下：

     1）这个项目是干什么的（比如是一个类似网易新闻的地方新闻客户端，或者类似美团的o2o，或者类似豌豆荚的一个应用市场，或者类似淘宝的购物平台）？解释就是拿一个市场上耳熟能详的应用跟自己的应用做类比，省的面试官听的云里雾里的。

    2）自己负责了哪些模块（功能）的职责（比如负责系统的架构，核心代码的编写，xx功能模块的开发等等）

    3）自己在这个项目中担当的责任（比如，这个项目是自己独立开发的，这个项目是和另外一个同事一起架构一起开发的，这个项目是自己负责了几个核心模块）

    4）项目中都用到了哪些技术

    5）从项目中学到了哪些东西（可以从技术方向和业务两个方向入手）

**旁白：**面试官问的很多技术性问题跟之前问的都大同小异，因此这里只给出有特色且技术含量高的。阳哥正在写面试宝典，该宝典核心内容针对的还是技术问题，阳哥会从javase基础到javase高级，从Android基础到Android高级以及到Android项目依次展开分析，其次也会写一些常见的非技术性问题，敬请期待~

**Q：**①在Listview的优化中，我们为何使用ConvertView？②为何使用ViewHolder？③你认为哪个更能解决问题？④你认为view.inflate和view.findviewById哪个更耗时，为什么？⑤如果这两个AP让你重新写，你怎么写？

**PS：**上面的问题，阳哥认为是面试以来遇到很难的一个，也是很有技术含量的一道题。前一半问题还好回答，最后一个问题真的需要发挥想象了。

**A：**①使用ConvertView可以实现对view的复用，这样大大节约了每次创建对象的时间，提升了ListView的显示效率。②使用ViewHolder作为内部类，可以将view的子控件封装在ViewHolder类中，然后通过View.setTag(ViewHolder)将view和ViewHolder进行绑定，这样我们就不用每次都调用view的findViewById(id)方法来查找控件。③使用ConvertView解决了一大部分问题，使用ViewHolder实现了控件换时间的问题，因为给View对象设置一个Tag本身就是占用内存的，因此ViewHolder的使用还是需要区分不同的应用场景的， 没有绝对的好与不好。如果内存足够需要高效则ViewHolder建议使用，否则不建议使用。④当然是view.inflate耗时，这个函数完成的功能是把xml布局文件通过pullParser的形式给解析到内存中，需要io，需要递归子节点。⑤我其实还不太相信我写出来的代码比Google官方写的好，如果让我写的话我可能会这样考虑，当用户在使用view.inflate的时候将多个id作为数组添加到形参中，这样在初始化view的使用我就可以给这个view直接调用setTag方法绑定需要的子控件。不过这个原生方法其实也应该保留共不同的需求使用。

**PS：**技术面试时间并不长，我回答了几个之后，他们两个大眼瞪小眼，A看看B问：你还有什么问的吗？B说我没有，你还有吗？A说我也没了。那行，接下来，他们就让我等人事了。

**3. 人事面试：**

    人事问的问题都差不多，我在上一篇也说过，这里就不说人事的问题了。唯一要给大家爆料的就是人事给我讲的他们公司的福利待遇，可以用土豪不差钱形容:)~

**人事说：**入职后转正即送iPhone 6一部。

**我（阳哥）说：**我是做Android开发的，给我iPhone 6干嘛

**旁白：**你认为阳哥真不想要iPhone 6吗？为了显示咱的高度敬业精神，毕竟咱是做Android开发的，更需要的是Android手机，决不能像iPhone低腰（ni dao shi gei wo ya）！

**人事说：**我们Android开发人员也会额外配Android机的，iPhone6可以生活用。

**旁白：**我去~这就是没见过世面的阳哥，当时的表情。

IMG_256

**人事说：**转正满一年，每年都有一次出国旅游。

**我（阳哥）说：**哦，咱们公司还挺不错的嘛！

**旁白：**不知说啥好了，只能用表情来形容了，阳哥至今没出过国，国外啥个样子嘛，

IMG_257

**人事说：**我们一般一年更少发14薪，都是介于14~16薪之间，具体发多少要看个人平时的一个KPI考核了。

**我（阳哥）说：**哦，这样呀，还行吧。

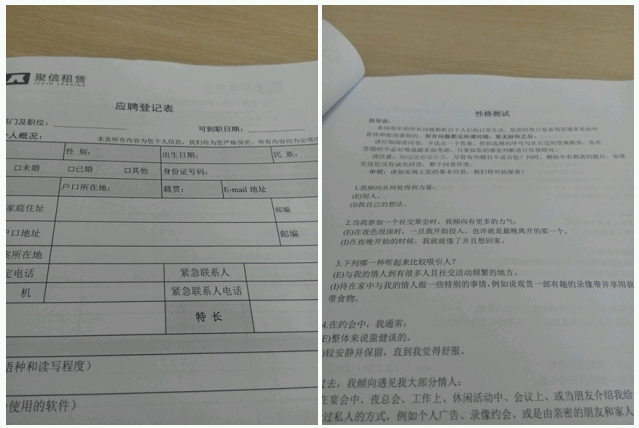
IMG_258

**面试吐槽：**

       在家公司的人事跟我算了基本工资16k+各种补助1k=17k+。每年出国旅游，入职送iPhone，补充住房公积金，好吧，阳哥受宠若惊了。自己回来会百度了一下，终于搞明白这家公司为何这么奢侈！不多说，看图吧。



跟人事面试的时候，人事说他们在上海总部目前有大概100多名员工，阳哥不懂经济，不知道100多人的公司能融资13个亿是个什么样的概念？这再次验证了阳哥的那句千古流传的话语，只要技术好，公司不差钱！



# 附录：更新记录

|  |  |  |  |
| --- | --- | --- | --- |
| 时间 | 类型 | 操作人 | 备注 |
| 2016年1月28号 | 创建文档 | 谌桃/孙健钦/周浩 | 新建 |
|  | OC语言/UI高级 | 谌桃 | 负责 |
|  | Swift语言/UI基础 | 孙健钦 | 负责 |
|  | C语言/网络多线程 | 周浩 | 负责 |
|  | 数据存储 | 周浩 | 负责 |
|  | 屏幕适配 | 谌桃 | 负责 |
|  | iOS性能优化 | 孙健钦 | 负责 |
|  | 第三方框架 | 谌桃/孙健钦/周浩 | 负责 |
|  | 设计模式 | 周浩 | 负责 |
|  | 项目常见问题 | 孙健钦/周浩/谌桃 | 负责 |
|  | 项目实战 | 周浩/孙健钦/谌桃 | 负责 |
|  | 面试实战记录 | 阳哥 |  |

***至此，本文档完！***

2015年11月26日 星期四 小雨

上海市闵行区联航路1588号坤大科技楼5层501