最全的iOS面试题及答案

1. Object-c的类可以多重继承么?可以实现多个接口么?Category是什么?重写一个类的方式用继承好还是分类好?为什么?

答： Object-c的类不可以多重继承;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。

2. #import 跟#include 又什么区别，@class呢, #import<> 跟#import""又什么区别?

答：#import是Objective-C导入头文件的关键字，#include是C/C++导入头文件的关键字,使用#import头文件会自动只导入一次，不会重复导入，相当于#include和#pragma once;@class告诉编译器某个类的声明，当执行时，才去查看类的实现文件，可以解决头文件的相互包含;#import<>用来包含系统的头文件，#import""用来包含用户头文件。

3. 属性readwrite，readonly，assign，retain，copy，nonatomic 各是什么作用，在那种情况下用?

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

2. 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

3. 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;

4. 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;

5. 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。

6. 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

4.写一个setter方法用于完成@property (nonatomic,retain)NSString *name,写一个setter方法用于完成@property(nonatomic，copy)NSString *name

```
- (void) setName:(NSString*) str
{
[str retain];
[name release];
name = str;
}
- (void)setName:(NSString *)str
{
id t = [str copy];
[name release];
name = t;
}
```

5.对于语句NSString*obj = [[NSData alloc] init]; obj在编译时和运行时分别时什么类型的对象?

编译时是NSString的类型;运行时是NSData类型的对象

6.常见的object-c的数据类型有那些， 和C的基本数据类型有什么区别?如：NSInteger和int

object-c的数据类型有NSString，NSNumber，NSArray，NSMutableArray，NSData等等，这些都是class，创建后便是对象，而C语言的基本数据类型int，只是一定字节的内存空间，用于存放数值;NSInteger是基本数据类型，并不是NSNumber的子类，当然也不是NSObject的子类。NSInteger是基本数据类型Int或者Long的别名(NSInteger的定义typedef long NSInteger)，它的区别在于，NSInteger会根据系统是32位还是64位来决定是本身是int还是Long。

7.id 声明的对象有什么特性?

Id 声明的对象具有运行时的特性，即可以指向任意类型的objcetive-c的对象;

8.Objective-C如何对内存管理的,说说你的看法和解决方法?

Objective-C的内存管理主要有三种方式ARC(自动内存计数)、手动内存计数、内存池。

1. (Garbage Collection)自动内存计数：这种方式和java类似，在你的程序的执行过程中。始终有一个高人在背后准确地帮你收拾垃圾，你不用考虑它什么时候开始工作，怎样工作。你只需要明白，我申请了一段内存空间，当我不再使用从而这段内存成为垃圾的时候，我就彻底的把它忘记掉，反正那个高人会帮我收拾垃圾。遗憾的是，那个高人需要消耗一定的资源，在携带设备里面，资源是紧俏商品所以iPhone不支持这个功能。所以“Garbage Collection”不是本入门指南的范围，对“Garbage Collection”内部机制感兴趣的同学可以参考一些其他的资料，不过说老实话“Garbage Collection”不大适合适初学者研究。

解决: 通过alloc – initial方式创建的, 创建后引用计数+1, 此后每retain一次引用计数+1, 那么在程序中做相应次数的release就好了.

2. (Reference Counted)手动内存计数：就是说，从一段内存被申请之后，就存在一个变量用于保存这段内存被使用的次数，我们暂时把它称为计数器，当计数器变为0的时候，那么就是释放这段内存的时候。比如说，当在程序A里面一段内存被成功申请完成之后，那么这个计数器就从0变成1(我们把这个过程叫做alloc)，然后程序B也需要使用这个内存，那么计数器就从1变成了2(我们把这个过程叫做retain)。紧接着程序A不再需要这段内存了，那么程序A就把这个计数器减1(我们把这个过程叫做release);程序B也不再需要这段内存的时候，那么也把计数器减1(这个过程还是release)。当系统(也就是Foundation)发现这个计数器变成了0，那么就会调用内存回收程序把这段内存回收(我们把这个过程叫做dealloc)。顺便提一句，如果没有Foundation，那么维护计数器，释放内存等等工作需要你手工来完成。

解决:一般是由类的静态方法创建的, 函数名中不会出现alloc或init字样, 如[NSString string]和[NSArray arrayWithObject:], 创建后引用计数+0, 在函数出栈后释放, 即相当于一个栈上的局部变量. 当然也可以通过retain延长对象的生存期.

3. (NSAutoRealeasePool)内存池：可以通过创建和释放内存池控制内存申请和回收的时机.

解决:是由autorelease加入系统内存池, 内存池是可以嵌套的, 每个内存池都需要有一个创建释放对, 就像main函数中写的一样. 使用也很简单, 比如[[[NSString alloc]initialWithFormat:@”Hey you!”] autorelease], 即将一个NSString对象加入到最内层的系统内存池, 当我们释放这个内存池时, 其中的对象都会被释放.

9. 原子(atomic)跟非原子(non-atomic)属性有什么区别?

1. atomic提供多线程安全。是防止在写未完成的时候被另外一个线程读取，造成数据错误

2. non-atomic:在自己管理内存的环境中，解析的访问器保留并自动释放返回的值，如果指定了 nonatomic ，那么访问器只是简单地返回这个值。

10. 看下面的程序,第一个NSLog会输出什么?这时str的retainCount是多少?第二个和第三个呢? 为什么?

```
==========================================================
NSMutableArray* ary = [[NSMutableArray array] retain];
NSString *str = [NSString stringWithFormat:@"test"];
[strretain];
[aryaddObject:str];
NSLog(@”%@%d”,str,[str retainCount]);
[strretain];
[strrelease];
[strrelease];
NSLog(@”%@%d”,str,[str retainCount]);
[aryremoveAllObjects];
NSLog(@”%@%d”,str,[str retainCount]);
==========================================================
```

str的retainCount创建+1，retain+1，加入数组自动+1 3

retain+1，release-1，release-1 2

数组删除所有对象，所有数组内的对象自动-1 1

11. 内存管理的几条原则时什么?按照默认法则.那些关键字生成的对象需要手动释放?在和property结合的时候怎样有效的避免内存泄露?

谁申请，谁释放

遵循Cocoa Touch的使用原则;

内存管理主要要避免“过早释放”和“内存泄漏”，对于“过早释放”需要注意@property设置特性时，一定要用对特性关键字，对于“内存泄漏”，一定要申请了要负责释放，要细心。

关键字alloc 或new 生成的对象需要手动释放;

设置正确的property属性，对于retain需要在合适的地方释放，

12.如何对iOS设备进行性能测试?

Profile-> Instruments ->Time Profiler

13. Object C中创建线程的方法是什么?如果在主线程中执行代码，方法是什么?如果想延时执行代码、方法又是什么?

线程创建有三种方法：使用NSThread创建、使用GCD的dispatch、使用子类化的NSOperation,然后将其加入NSOperationQueue;在主线程执行代码，方法是performSelectorOnMainThread，如果想延时执行代码可以用performSelector:onThread:withObject:waitUntilDone:

14.描述一下iOS SDK中如何实现MVC的开发模式

MVC是模型、试图、控制开发模式，对于iOS SDK，所有的View都是视图层的，它应该独立于模型层，由视图控制层来控制。所有的用户数据都是模型层，它应该独立于视图。所有的ViewController都是控制层，由它负责控制视图，访问模型数据。

15 浅复制和深复制的区别?

答案：浅层复制：只复制指向对象的指针，而不复制引用对象本身。

深层复制：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。深复制就好理解了,内存中存在了

两份独立对象本身。

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。

16. 类别的作用?继承和类别在实现中有何区别?

答案：category 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。

类别主要有3个作用：

(1)将类的实现分散到多个不同文件或多个不同框架中。

(2)创建对私有方法的前向引用。

(3)向对象添加非正式协议。

继承可以增加，修改或者删除方法，并且可以增加属性。

17. 类别和类扩展的区别。

答案：category和extensions的不同在于 后者可以添加属性。另外后者添加的方法是必须要实现的。

extensions可以认为是一个私有的Category。

18. oc中的协议和java中的接口概念有何不同?

答案：OC中的代理有2层含义，官方定义为 formal和informal protocol。前者和Java接口一样。

informal protocol中的方法属于设计模式考虑范畴，不是必须实现的，但是如果有实现，就会改变类的属性。

其实关于正式协议，类别和非正式协议我很早前学习的时候大致看过，也写在了学习教程里

“非正式协议概念其实就是类别的另一种表达方式“这里有一些你可能希望实现的方法，你可以使用他们更好的完成工作”。

这个意思是，这些是可选的。比如我门要一个更好的方法，我们就会申明一个这样的类别去实现。然后你在后期可以直接使用这些更好的方法。

这么看，总觉得类别这玩意儿有点像协议的可选协议。”

现在来看，其实protocal已经开始对两者都统一和规范起来操作，因为资料中说“非正式协议使用interface修饰“，

现在我们看到协议中两个修饰词：“必须实现(@requied)”和“可选实现(@optional)”。

19. 什么是KVO和KVC?

答案：kvc:键 – 值编码是一种间接访问对象的属性使用字符串来标识属性，而不是通过调用存取方法，直接或通过实例变量访问的机制。

很多情况下可以简化程序代码。apple文档其实给了一个很好的例子。

kvo:键值观察机制，他提供了观察某一属性变化的方法，极大的简化了代码。

具体用看到嗯哼用到过的一个地方是对于按钮点击变化状态的的监控。

比如我自定义的一个button

```
[self addObserver:self forKeyPath:@"highlighted" options:0 context:nil];
#pragma mark KVO
- (void)observeValueForKeyPath:(NSString *)keyPath ofObject:(id)object change:(NSDictionary *)change context:(void *)context
{
if ([keyPath isEqualToString:@"highlighted"] ) {
[self setNeedsDisplay];
}
}
```

对于系统是根据keypath去取的到相应的值发生改变，理论上来说是和kvc机制的道理是一样的。

对于kvc机制如何通过key寻找到value：

“当通过KVC调用对象时，比如：[self valueForKey:@”someKey”]时，程序会自动试图通过几种不同的方式解析这个调用。首先查找对象是否带有someKey 这个方法，如果没找到，会继续查找对象是否带有someKey这个实例变量(iVar)，如果还没有找到，程序会继续试图调用 -(id) valueForUndefinedKey:这个方法。如果这个方法还是没有被实现的话，程序会抛出一个NSUndefinedKeyException异常错误。

(cocoachina.com注：Key-Value Coding查找方法的时候，不仅仅会查找someKey这个方法，还会查找getsomeKey这个方法，前面加一个get，或者_someKey以及_getsomeKey这几种形式。同时，查找实例变量的时候也会不仅仅查找someKey这个变量，也会查找_someKey这个变量是否存在。)

设计valueForUndefinedKey:方法的主要目的是当你使用-(id)valueForKey方法从对象中请求值时，对象能够在错误发生前，有最后的机会响应这个请求。这样做有很多好处，下面的两个例子说明了这样做的好处。“

来至cocoa，这个说法应该挺有道理。

因为我们知道button却是存在一个highlighted实例变量.因此为何上面我们只是add一个相关的keypath就行了，

可以按照kvc查找的逻辑理解，就说的过去了。

20. 代理的作用?

答案：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。
另外一点，代理可以理解为java中的回调监听机制的一种类似。

21. oc中可修改和不可以修改类型。
答案：可修改不可修改的集合类。这个我个人简单理解就是可动态添加修改和不可动态添加修改一样。
比如NSArray和NSMutableArray。前者在初始化后的内存控件就是固定不可变的，后者可以添加等，可以动态申请新的内存空间。

22. 我们说的oc是动态运行时语言是什么意思?
答案：多态。 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?~~~

23. 通知和协议的不同之处?
答案：协议有控制链(has-a)的关系，通知没有。
首先我一开始也不太明白，什么叫控制链(专业术语了~)。但是简单分析下通知和代理的行为模式，我们大致可以有自己的理解
简单来说，通知的话，它可以一对多，一条消息可以发送给多个消息接受者。
代理按我们的理解，到不是直接说不能一对多，比如我们知道的明星经济代理人，很多时候一个经济人负责好几个明星的事务。
只是对于不同明星间，代理的事物对象都是不一样的，一一对应，不可能说明天要处理A明星要一个发布会，代理人发出处理发布会的消息后，别称B的
发布会了。但是通知就不一样，他只关心发出通知，而不关心多少接收到感兴趣要处理。
因此控制链(has-a从英语单词大致可以看出，单一拥有和可控制的对应关系。

24. 是推送消息?
答案：太简单，不作答~~~~~~~~~~
这是cocoa上的答案。
其实到不是说太简单，只是太泛泛的一个概念的东西。就好比说，什么是人。
推送通知更是一种技术。
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。 测试push的实现可以查看该博文。

25. 关于多态性
答案：多态，子类指针可以赋值给父类。

这个题目其实可以出到一切面向对象语言中，
因此关于多态，继承和封装基本最好都有个自我意识的理解，也并非一定要把书上资料上写的能背出来。
最重要的是转化成自我理解。

26. 对于单例的理解

答案：11，12题目其实出的有点泛泛的感觉了，可能说是编程语言需要或是必备的基础。
基本能用熟悉的语言写出一个单例，以及可以运用到的场景或是你编程中碰到过运用的此种模式的框架类等。
进一步点，考虑下如何在多线程访问单例时的安全性。

27. 说说响应链

答案： 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。因为问的太抽象化了
严重怀疑题目出到越后面就越笼统。
可以从责任链模式，来讲通过事件响应链处理，其拥有的扩展性

28. frame和bounds有什么不同?

答案: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)

29. 方法和选择器有何不同?

答案：selector是一个方法的名字，method是一个组合体，包含了名字和实现.
详情可以看apple文档。

30. OC的垃圾回收机制?

答案： OC2.0有Garbage collection，但是iOS平台不提供。
一般我们了解的objective-c对于内存管理都是手动操作的，但是也有自动释放池。
但是差了大部分资料，貌似不要和arc机制搞混就好了。

31. NSOperation queue?

答案：存放NSOperation的集合类。
操作和操作队列，基本可以看成java中的线程和线程池的概念。用于处理ios多线程开发的问题。
网上部分资料提到一点是，虽然是queue，但是却并不是带有队列的概念，放入的操作并非是按照严格的先进现出。
这边又有个疑点是，对于队列来说，先进先出的概念是Afunc添加进队列，Bfunc紧跟着也进入队列，Afunc先执行这个是必然的，
但是Bfunc是等Afunc完全操作完以后，B才开始启动并且执行，因此队列的概念离乱上有点违背了多线程处理这个概念。

但是转念一想其实可以参考银行的取票和叫号系统。
因此对于A比B先排队取票但是B率先执行完操作，我们亦然可以感性认为这还是一个队列。
但是后来看到一票关于这操作队列话题的文章，其中有一句提到
“因为两个操作提交的时间间隔很近，线程池中的线程，谁先启动是不定的。”
瞬间觉得这个queue名字有点忽悠人了，还不如pool~
综合一点，我们知道他可以比较大的用处在于可以帮组多线程编程就好了。

32. 什么是延迟加载?
答案：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。

33. 是否在一个视图控制器中嵌入两个tableview控制器?
答案：一个视图控制只提供了一个View视图，理论上一个tableViewController也不能放吧，
只能说可以嵌入一个tableview视图。当然，题目本身也有歧义，如果不是我们定性思维认为的UIViewController，
而是宏观的表示视图控制者，那我们倒是可以把其看成一个视图控制者，它可以控制多个视图控制器，比如TabbarController
那样的感觉。

34. 一个tableView是否可以关联两个不同的数据源?你会怎么处理?
答案：首先我们从代码来看，数据源如何关联上的，其实是在数据源关联的代理方法里实现的。
因此我们并不关心如何去关联他，他怎么关联上，方法只是让我返回根据自己的需要去设置如相关的数据源。
因此，我觉得可以设置多个数据源啊，但是有个问题是，你这是想干嘛呢?想让列表如何显示，不同的数据源分区块显示?

35. 什么时候使用NSMutableArray，什么时候使用NSArray?
答案：当数组在程序运行时，需要不断变化的，使用NSMutableArray，当数组在初始化后，便不再改变的，使用NSArray。需要指出的是，使用NSArray只表明的是该数组在运行时不发生改变，即不能往NSAarry的数组里新增和删除元素，但不表明其数组内的元素的内容不能发生改变。NSArray是线程安全的，NSMutableArray不是线程安全的，多线程使用到NSMutableArray需要注意。

36. 给出委托方法的实例，并且说出UITableVIew的Data Source方法
答案：CocoaTouch框架中用到了大量委托，其中UITableViewDelegate就是委托机制的典型应用，是一个典型的使用委托来实现适配器模式，其中UITableViewDelegate协议是目标，tableview是适配器，实现UITableViewDelegate协议，并将自身设置为talbeview的delegate的对象，是被适配器，一般情况下该对象是UITableViewController。
UITableVIew的Data Source方法有- (NSInteger)tableView:(UITableView

*)tableView numberOfRowsInSection:(NSInteger)section;
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath;

37. 在应用中可以创建多少autorelease对象，是否有限制?
答案：无

38. 如果我们不创建内存池，是否有内存池提供给我们?
答案:界面线程维护着自己的内存池，用户自己创建的数据线程，则需要创建该线程的内存池

39. 什么时候需要在程序中创建内存池?
答案：用户自己创建的数据线程，则需要创建该线程的内存池

40. 类NSObject的那些方法经常被使用?
答案：NSObject是Objetive-C的基类，其由NSObject类及一系列协议构成。
其中类方法alloc、class、 description 对象方法init、dealloc、–performSelector:withObject:afterDelay:等经常被使用

41. 什么是简便构造方法?
答案：简便构造方法一般由CocoaTouch框架提供，如NSNumber的 + numberWithBool: + numberWithChar: + numberWithDouble: + numberWithFloat: + numberWithInt:
Foundation下大部分类均有简便构造方法，我们可以通过简便构造方法，获得系统给我们创建好的对象，并且不需要手动释放。

42. 如何使用Xcode设计通用应用?
答案：使用MVC模式设计应用，其中Model层完成脱离界面，即在Model层，其是可运行在任何设备上，在controller层，根据iPhone与iPad(独有UISplitViewController)的不同特点选择不同的viewController对象。在View层，可根据现实要求，来设计，其中以xib文件设计时，其设置其为universal。

43. UIView的动画效果有那些?
答案：有很多，如 UIViewAnimationOptionCurveEaseInOut UIViewAnimationOptionCurveEaseIn UIViewAnimationOptionCurveEaseOut UIViewAnimationOptionTransitionFlipFromLeft UIViewAnimationOptionTransitionFlipFromRight UIViewAnimationOptionTransitionCurlUpUIViewAnimationOptionTransitionCurlDown

44. 在iPhone应用中如何保存数据?
答案：有以下几种保存机制：
1.通过web服务，保存在服务器上

2.通过NSCoder固化机制，将对象保存在文件中
3.通过SQlite或CoreData保存在文件数据库中

45. 什么是coredata?
答案：coredata是苹果提供一套数据保存框架，其基于SQlite

46. 什么是NSManagedObject模型?
答案：NSManagedObject是NSObject的子类 ，也是coredata的重要组成部分，它是一个通用的类,实现了core data 模型层所需的基本功能，用户可通过子类化NSManagedObject，建立自己的数据模型。

47. 什么是NSManagedobjectContext?
答案：NSManagedobjectContext对象负责应用和数据库之间的交互。

48. 什么是谓词?
答案：谓词是通过NSPredicate，是通过给定的逻辑条件作为约束条件，完成对数据的筛选。
predicate = [NSPredicate predicateWithFormat:@"customerID == %d",n];
a = [customers filteredArrayUsingPredicate:predicate];

49. 和coredata一起有哪几种持久化存储机制?
答案：存入到文件、 存入到NSUserDefaults(系统plist文件中)、存入到Sqlite文件数据库

50. 谈谈对Block 的理解?并写出一个使用Block执行UIVew动画?
答案：Block是可以获取其他函数局部变量的匿名函数，其不但方便开发，并且可以大幅提高应用的执行效率(多核心CPU可直接处理Block指令)
[UIView transitionWithView:self.view
duration:0.2
options:UIViewAnimationOptionTransitionFlipFromLeft
animations:^{ [[blueViewController view] removeFromSuperview]; [[self view] insertSubview:yellowViewController.view atIndex:0]; }
completion:NULL];

51. 写出上面代码的Block的定义。
答案：
typedef void(^animations) (void);
typedef void(^completion) (BOOL finished);

52. 试着使用+ beginAnimations:context:以及上述Block的定义，写出一个可以完成
+ (void)transitionWithView:(UIView *)view duration:(NSTimeInterval)duration options:(UIViewAnimationOptions)options animations:(void (^)(void))animations completion:(void (^)(BOOL

finished))completion NS_AVAILABLE_IOS(4_0);操作的函数执行部分

答案：无

网络部分

53. 做过的项目是否涉及网络访问功能，使用什么对象完成网络功能?

答案：ASIHTTPRequest与NSURLConnection

54. 简单介绍下NSURLConnection类及+ sendSynchronousRequest:returningResponse:error:与– initWithRequest:delegate:两个方法的区别?

答案: NSURLConnection主要用于网络访问，其中+ sendSynchronousRequest:returningResponse:error:是同步访问数据，即当前线程会阻塞，并等待request的返回的response，而– initWithRequest:delegate:使用的是异步加载，当其完成网络访问后，会通过delegate回到主线程，并其委托的对象。

55. 多线程是什么

多线程是个复杂的概念，按字面意思是同步完成多项任务，提高了资源的使用效率，从硬件、操作系统、应用软件不同的角度去看，多线程被赋予不同的内涵，对于硬件，现在市面上多数的CPU都是多核的，多核的CPU运算多线程更为出色;从操作系统角度，是多任务，现在用的主流操作系统都是多任务的，可以一边听歌、一边写博客;对于应用来说，多线程可以让应用有更快的回应，可以在网络下载时，同时响应用户的触摸操作。在iOS应用中，对多线程最初的理解，就是并发，它的含义是原来先做烧水，再摘菜，再炒菜的工作，会变成烧水的同时去摘菜，最后去炒菜。

56. iOS 中的多线程

iOS中的多线程，是Cocoa框架下的多线程，通过Cocoa的封装，可以让我们更为方便的使用线程，做过C++的同学可能会对线程有更多的理解，比如线程的创立，信号量、共享变量有认识，Cocoa框架下会方便很多，它对线程做了封装，有些封装，可以让我们创建的对象，本身便拥有线程，也就是线程的对象化抽象，从而减少我们的工程，提供程序的健壮性。

GCD是(Grand Central Dispatch)的缩写 ，从系统级别提供的一个易用地多线程类库，具有运行时的特点，能充分利用多核心硬件。GCD的API接口为C语言的函数，函数参数中多数有Block，关于Block的使用参看这里，为我们提供强大的“接口”，对于GCD的使用参见本文

NSOperation与Queue

NSOperation是一个抽象类，它封装了线程的细节实现，我们可以通过子类化该对象，加上NSQueue来同面向对象的思维，管理多线程程序。具体可参看这里：一个基于NSOperation的多线程网络访问的项目。

NSThread

NSThread是一个控制线程执行的对象，它不如NSOperation抽象，通过它我们可以方便的得到一个线程，并控制它。但NSThread的线程之间的并发控制，是需要我们自己来控制的，可以通过NSCondition实现。

参看 iOS多线程编程之NSThread的使用

其他多线程

在Cocoa的框架下，通知、Timer和异步函数等都有使用多线程，(待补充).

57. 在项目什么时候选择使用GCD，什么时候选择NSOperation?

项目中使用NSOperation的优点是NSOperation是对线程的高度抽象，在项目中使用它，会使项目的程序结构更好，子类化NSOperation的设计思路，是具有面向对象的优点(复用、封装)，使得实现是多线程支持，而接口简单，建议在复杂项目中使用。

项目中使用GCD的优点是GCD本身非常简单、易用，对于不复杂的多线程操作，会节省代码量，而Block参数的使用，会是代码更为易读，建议在简单项目中使用。

58. 什么是block

对于闭包(block),有很多定义，其中闭包就是能够读取其它函数内部变量的函数，这个定义即接近本质又较好理解。对于刚接触Block的同学，会觉得有些绕，因为我们习惯写这样的程序main(){ funA();} funA(){funB();} funB(){…..}; 就是函数main调用函数A，函数A调用函数B… 函数们依次顺序执行，但现实中不全是这样的，例如项目经理M，手下有3个程序员A、B、C，当他给程序员A安排实现功能F1时，他并不等着A完成之后，再去安排B去实现F2，而是安排给A功能F1，B功能F2，C功能F3，然后可能去写技术文档，而当A遇到问题时，他会来找项目经理M，当B做完时，会通知M，这就是一个异步执行的例子。在这种情形下，Block便可大显身手，因为在项目经理M，给A安排工作时，同时会告诉A若果遇到困难，如何能找到他报告问题(例如打他手机号)，这就是项目经理M给A的一个回调接口，要回掉的操作，比如接到电话，百度查询后，返回网页内容给A，这就是一个Block，在M交待工作时，已经定义好，并且取得了F1的任务号(局部变量)，却是在当A遇到问题时，才调用执行，跨函数在项目经理M查询百度，获得结果后回调该block。

59. block 实现原理

Objective-C是对C语言的扩展，block的实现是基于指针和函数指针。

从计算语言的发展，最早的goto，高级语言的指针，到面向对象语言的block，从机器的思维，一步步接近人的思维，以方便开发人员更为高效、直接的描述出现实的逻辑(需求)。

下面是两篇很好的介绍block实现的博文

iOS中block实现的探究

谈Objective-C Block的实现

3 block的使用

使用实例

cocoaTouch框架下动画效果的Block的调用

使用typed声明block

typedef void(^didFinishBlock) (NSObject *ob);

这就声明了一个didFinishBlock类型的block，

然后便可用

@property (nonatomic,copy) didFinishBlock finishBlock;

声明一个blokc对象，注意对象属性设置为copy，接到block 参数时，便会自动复制一份。

__block是一种特殊类型，

使用该关键字声明的局部变量，可以被block所改变，并且其在原函数中的值会被改变。

4 常见系列面试题

面试时，面试官会先问一些，是否了解block，是否使用过block，这些问题相当于开场白，往往是下面一系列问题的开始，所以一定要如实根据自己的

情况回答。

1 使用block和使用delegate完成委托模式有什么优点?

首先要了解什么是委托模式，委托模式在iOS中大量应用，其在设计模式中是适配器模式中的对象适配器，Objective-C中使用id类型指向一切对象，使委托模式更为简洁。了解委托模式的细节：

iOS设计模式一-委托模式

使用block实现委托模式，其优点是回调的block代码块定义在委托对象函数内部，使代码更为紧凑;

适配对象不再需要实现具体某个protocol，代码更为简洁。

2 多线程与block

GCD与Block

使用 dispatch_async 系列方法，可以以指定的方式执行block

GCD编程实例

dispatch_async的完整定义

void dispatch_async(

dispatch_queue_t queue,

dispatch_block_t block);

功能：在指定的队列里提交一个异步执行的block，不阻塞当前线程

通过queue来控制block执行的线程。主线程执行前文定义的 finishBlock对象

dispatch_async(dispatch_get_main_queue(),^(void){finishBlock();});

盘点2016年iOS开发面试题及答案整理

2016-03-03 14:53:59 3145浏览

在现如今，随着移动互联网科技不断的发展和创新，如今无论是公司还是开发者或设计师个人而言，面试都是一项耗时耗钱的项目，而面对iOS开发者及设计师在面试时可能会遇到的问题进行了筛选与汇总。下面我们一起来一下吧。

1、简述OC中内存管理机制。与retain配对使用的方法是dealloc还是release，为什么?需要与alloc配对使用的方法是dealloc还是release，为什么?readwrite，readonly，assign，retain，copy，nonatomic 、atomic、strong、weak属性的作用?

管理机制：使用了一种叫做引用计数的机制来管理内存中的对象。OC中每个对象都对应着他们自己的引用计数，引用计数可以理解为一个整数计数器，当使用alloc方法创建对象的时候，持有计数会自动设置为1。当你向一个对象发送retain消息 时，持有计数数值会增加1。相反，当你像一个对象发送release消息时，持有计数数值会减小1。当对象的持有计数变为0的时候，对象会释放自己所占用的内存。

retain(引用计数加1)->release(引用计数减1)

alloc(申请内存空间)->dealloc(释放内存空间)

readwrite: 表示既有getter，也有setter (默认)

readonly: 表示只有getter，没有setter

nonatomic:不考虑线程安全

atomic:线程操作安全 (默认)

线程安全情况下的setter和getter：

- (NSString*) value {

@synchronized(self) {

return [[_value retain] autorelease];

}}

(void) setValue:(NSString*)aValue {

@synchronized(self) {

[aValue retain];

[_value release];

_value = aValue;

} }

retain: release旧的对象，将旧对象的值赋予输入对象，再提高输入对象的索引计数为1

assign: 简单赋值，不更改索引计数 (默认)

copy: 其实是建立了一个相同的对象,地址不同(retain：指针拷贝 copy：内容拷贝)

strong:(ARC下的)和(MRC)retain一样 (默认)

weak:(ARC下的)和(MRC)assign一样， weak当指向的内存释放掉后自动nil化，防止野指针

unsafe_unretained 声明一个弱应用，但是不会自动nil化，也就是说，如果所指向的内存区域被释放了，这个指针就是一个野指针了。
autoreleasing 用来修饰一个函数的参数，这个参数会在函数返回的时候被自动释放。

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

@private：作用范围只能在自身类

@protected：作用范围在自身类和继承自己的子类 (默认)

@public：作用范围最大，可以在任何地方被访问。

@package：这个类型最常用于框架类的实例变量,同一包内能用，跨包就不能访问

3、线程是什么?进程是什么?二者有什么区别和联系?

一个程序至少有一个进程,一个进程至少有一个线程:

进程：一个程序的一次运行，在执行过程中拥有独立的内存单元，而多个线程共享一块内存

线程：线程是指进程内的一个执行单元。

联系：线程是进程的基本组成单位

区别：(1)调度：线程作为调度和分配的基本单位，进程作为拥有资源的基本单位
(2)并发性：不仅进程之间可以并发执行，同一个进程的多个线程之间也可并发执行
(3)拥有资源：进程是拥有资源的一个独立单位，线程不拥有系统资源，但可以访问隶属于进程的资源.
(4)系统开销：在创建或撤消进程时，由于系统都要为之分配和回收资源，导致系统的开销明显大于创建或撤消线程时的开销。

举例说明：操作系统有多个软件在运行(QQ、office、音乐等)，这些都是一个个进程，而每个进程里又有好多线程(比如QQ，你可以同时聊天，发送文件等)

4、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中的任务放到后台去处理

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.程序的运行速度可能加快

4·在一些等待的任务实现上如用户输入、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,因为操作系统需要在它们之间切换。

2.更多的线程需要更多的内存空间。

3.线程的中止需要考虑其对程序运行的影响。

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的发生。

实现多线程的方法:

NSObject类方法

NSThread

NSOperation

GCD

5、线程同步和异步的区别?IOS中如何实现多线程的同步?

异步：举个简单的例子 就是游戏，游戏会有图像和背景音乐

同步:是指一个线程要等待上一个线程执行完之后才开始执行当前的线程,上厕所

NSOperationQueue：maxcurrentcount

NSConditionLock

6、假设有一个字符串aabcad，请写一段程序，去掉字符串中不相邻的重复字符串，即上述字符串处理之后的输出结果为：aabcd

```
NSMutableString * str = [[NSMutableString alloc]initWithFormat;@“aabcad”];

for (int i = 0 ,i < str.length - 1 ;i++){

unsigned char a = [str characterAtIndex:i];

for (int j = i + 1 ,j < str.length ,j++){

unsigned char b = [str characterAtIndex:j];

if (a == b ){

if (j == i + 1){

}else{

[str deleteCharactersInRange:NSMakeRange(j, 1)];

}

}

}

}

NSLog(@“%@”,str);
```

7、获取一台设备唯一标识的方法有哪些?

(1)UDID

(2)UUID

(3)MAC Address

(4)OPEN UDID

(5)广告标识符

(6)Vindor标示符

8、iOS类是否可以多继承?如果没有，那可以用其他方法实现吗?简述实现过程。

不可以多继承 用protocol实现

9、堆和栈的区别?

堆需要用户手动释放内存，而栈则是编译器自动释放内存

问题扩展：要知道OC中NSString的内存存储方式

10、iOS本地数据存储都有哪几种方式?

NSKeyedArchiver

NSUserDefaults

Write写入方式

SQLite3

(问题扩展：什么情况下使用什么样的数据存储)

1.NSKeyedArchiver：采用归档的形式来保存数据，数据对象需要遵守NSCoding协议，对象对应的类必须提供encodeWithCoder:和initWithCoder:方法。缺点：只能一次性归档保存以及一次性解压。所以只能针对小量数据，对数据操作比较笨拙，如果想改动数据的某一小部分，需要解压或归档整个数据。

2.NSUserDefaults：用来保存应用程序设置和属性、用户保存的数据。用户再次打开程序或开机后这些数据仍然存在。NSUserDefaults可以存储的数据类型包括：NSData、NSString、NSNumber、NSDate、NSArray、NSDictionary。缺点：如果要存储其他类型，需要转换为前面的类型，才能用NSUserDefaults存储。

3.Write写入方式：永久保存在磁盘中。第一步：获得文件即将保存的路径：第二步：生成在该路径下的文件：第三步：往文件中写入数据：最后：从文件中读出数据：

4. SQLite：采用SQLite数据库来存储数据。SQLite作为一中小型数据库，应用ios中，跟前三种保存方式相比，相对比较复杂一些。

11、写出方法获取iOS内存使用情况。

```
// 获取当前设备可用内存及所占内存的头文件
#import
#import
// 获取当前设备可用内存(单位：MB)
- (double)availableMemory
{
vm_statistics_data_t vmStats;
mach_msg_type_number_t infoCount = HOST_VM_INFO_COUNT;
kern_return_t kernReturn = host_statistics(mach_host_self(),
HOST_VM_INFO,
(host_info_t)&vmStats,
&infoCount);
if (kernReturn != KERN_SUCCESS) {
return NSNotFound;
}
return ((vm_page_size *vmStats.free_count) / 1024.0) / 1024.0;
}
// 获取当前任务所占用的内存(单位：MB)
- (double)usedMemory
{
task_basic_info_data_t taskInfo;
mach_msg_type_number_t infoCount = TASK_BASIC_INFO_COUNT;
kern_return_t kernReturn = task_info(mach_task_self(),
TASK_BASIC_INFO,
```

(task_info_t)&taskInfo,

&infoCount);

if (kernReturn != KERN_SUCCESS

) {

return NSNotFound;

}

return taskInfo.resident_size / 1024.0 / 1024.0;

}

12、什么是安全释放?

置nil 再释放

13、写一个标准宏MIN，这个宏输入两个参数并返回较小的一个?

#define MIN(X,Y) ((X)>(Y)?(Y):(X))

扩展：在定义宏的时候需要注意哪些问题?

宏全部大写 写在#import 下 @interface上 结尾无分号

14、iphone os有没有垃圾回收机制?简单阐述一下OC内存管理。

iphone os没有垃圾回收机制 oc的内存管理是谁创建谁释放 程序中遇到retain 该对象引用计数+1 遇release该对象引用计数-1 retainCount为0时 内存释放

15、简述应用程序按Home键进入后台时的生命周期，以及从后台回到前台时的生命周期?

自己可以写个demo来测试一下

进入后台时

-(void)applicationWillResignActive:(UIApplication *)application;

-(void)applicationDidEnterBackground:(UIApplication *)application;

进入前台时

-(void)applicationDidEnterForeground:(UIApplication *)application;

-(void)applicationWillResignActive:(UIApplication *)application;

16、ViewController 的 alloc，loadView, viewDidLoad,viewWillAppear,viewDidUnload,dealloc、init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?

自己写代码测试加深理解

alloc申请内存时调用

loadView加载视图时调用

ViewDidLoad视图已经加载后调用

ViewWillAppear视图将要出现时调用

ViewDidUnload视图已经加载但没有加载出来调用

dealloc销毁该视图时调用

init视图初始化时调用

17、描述应用程序的启动顺序。

1. 程序入口main函数创建UIApplication实例和UIApplication代理实例。

2. 在UIApplication代理实例中重写启动方法，设置第一ViewController。

3. 在第一ViewController中添加控件，实现应用程序界面。

18、为什么很多内置类如UITableViewControl的delegate属性都是assign而不是retain?请举例说明。

防止循环引用

19、使用UITableView时候必须要实现的几种方法?

-(NSInteger)tableView:(UITableView*)tableViewNumberOfRowsInSection:(NSInteger)section;
这个方法返回每个分段的行数，不同分段返回不同的行数可以用switch来做，如果是单个列表就直接返回单个你想要的函数即可。
-(UITableViewCell*)tableView:(UITableView*)tableViewCellForRowAtIndexPath:(NSIndexPath)indexPath;
这个方法是返回我们调用的每一个单元格。通过我们索引的路径的section和row来确定

20、写一个便利构造器。

//id代表任意类型指针，这里代表Student *,类方法

+(id)studentWithName:(NSString *)newName andAge:(int)newAge
{

```
Student *stu=[[Student alloc]initName:newName andAge:newAge];
return [stu autorelease];//自动释放
}
```

21、UIImage初始化一张图片有几种方法?简述各自的优缺点。

3种

imageNamed:系统会先检查系统缓存中是否有该名字的Image，如果有的话，则直接返回，如果没有，则先加载图像到缓存，然后再返回。

initWithContentsOfFile：系统不会检查系统缓存，而直接从文件系统中加载并返回。

imageWithCGImage：scale：orientation当scale=1

常见错误列表:

第一个

this class is not key value coding-compliant for the key sdfd.'

出错原因:关联控件出错

解决方案:进到**sb**里面找到可能会出问题的页面**(ViewController)** 在黄色按钮上面右键 把带感叹号的删除

第二个

Collection <__NSArrayM: 0x7fea2ad0b290> was mutated while being enumerated.

出错原因:forin遍历数组时 数组长度发生了改变 导致

解决方案: 如果需要在遍历时改变数组 改变完之后 不能再继续遍历 应break 或 return

第三个

-[MyTableViewController addAction]: unrecognized selector sent to instance 0x7f892bc382b0'

原因：找不到方法

解决方法：查看是否类型错误 或者 方法未实现

第四个

'Invalid update: invalid number of rows in section 0. The number of rows contained in an existing section after the update (3) must be equal to the number of rows contained in that section before the update (3), plus or minus the number of rows inserted or deleted from that section (0 inserted, 1 deleted) and plus or minus the number of rows moved into or out of that section (0 moved in, 0 moved out).'

原因：在做tableview数据插入或者删除的时候 只从界面中删除了数据 没有从数据源数组中删除

解决方法： 从数据源数组中删除

第五个

unable to dequeue a cell with identifier cell – must register a nib or a class for the identifier or connect a prototype cell in a storyboard

原因：没有注册cell 或cell的重用标示没有加
解决方法：加上即可
第六个
Terminating app due to uncaught exception 'NSInternalInconsistencyException', reason: 'Invalid update: invalid number of rows in section 0. The number of rows contained in an existing section after the update (1) must be equal to the number of rows contained in that section before the update (1), plus or minus the number of rows inserted or deleted from that section (0 inserted, 1 deleted) and plus or minus the number of rows moved into or out of that section (0 moved in, 0 moved out).'
原因：在tableView删除数据的时候 只从界面中删除了 没有让数据源数组删除数据
解决方法：需要删除界面的同时 删除数据源数组里面的内容