收集百度、阿里等100家企业面试题合集上

本文编辑：修心

关注作者简书：iOS开发湿

本文分上下两篇，此篇是上篇（1-87），下篇（88-188）；从基础到进阶、底层，此文档由一个人整理，如有出现语法，排版错误，请联系作者，谢谢！

关注作者微信公众号，学习交流iOS技术，获取更多知识

目录：

6、NSProxy & NSObject

7、Object & Swift

8、传值通知 & 推送通知（本地&远程）

9、第三方库 & 第三方平台

10、NSCache & NSDcitionary

11、 UIView的setNeedsDisplay和setNeedsLayout方法

12、UILayer & UIView

13、layoutSubViews & drawRects

14、UDID & UUID

15、CPU & GPU

16、点（pt）& 像素（px）

17、属性与成员变量

18、int和NSInteger的区别

（1）import和include

（2）@class

（3）全局 & 静态变量

19、类和对象

(1）分类拓展协议中哪些可以声明属性?
(2）继承和类别的区别
(3）分类的作用
(4）分类的局限性

20、category & extension

21、Foundation

(1）字符串
(2）字符串截取
(3）格式

22、NSArray和NSDictionary

(1）iOS遍历数组/字典的方法
(2）NSValue NSNumber
(3）其它
(4）如何避免循环引用

23、CFSocket使用有哪几个步骤

24、Core Foundation中提供了哪几种操作Socket的方法?

25、解析XML文件有哪几种方式?

26、什么是沙盒模型？哪些操作是属于私有api范畴?

27、在一个对象的方法里面：self.name= “object”；和 name =”object” 有什么不同吗?

28、请简要说明viewDidLoad和viewDidUnload何时调用

29、创建控制器、视图的方式

30、简述内存分区情况

31、队列和栈有什么区别

32、iOS的系统架构

33、控件主要响应3种事件

34、xib文件的构成分为哪3个图标？都具有什么功能

35、简述视图控件器的生命周期

36、app 项目的生命周期

（1）应用的生命周期

（2）简要说明一下APP的启动过程，main文件说起，main函数中有什么函数？作用是什么？

（3）UIApplicationMain函数作用

（4）main函数作用

37、 动画有基本类型有哪几种；表视图有哪几种基本样式。

38、实现简单的表格显示需要设置UITableView的什么属性、实现什么协议?

39、Cocoa Touch提供了哪几种Core Animation过渡类型?

40、UIView与CLayer有什么区别?

41、Quatrz 2D的绘图功能的三个核心概念是什么并简述其作用

42、iPhone OS主要提供了几种播放音频的方法?

43、使用AVAudioPlayer类调用哪个框架、使用步骤?

44、有哪几种手势通知方法、写清楚方法名?

45、ViewController的didReceiveMemoryWarning怎么被调用

46、什么时候用delegate,什么时候用Notification?

47、用预处理指令#define声明一个常数，用以表明1年中有多少秒（忽略闰年问题）

48、写一个”标准"宏MIN ，这个宏输入两个参数并返回较小的一个。

49、关键字const有什么含意？修饰类呢?static的作用,用于类呢?还有extern c的作用

50、关键字volatile有什么含意?并给出三个不同的例子

51、一个参数既可以是const还可以是volatile吗？ 一个指针可以是volatile 吗？解释为什么。

52、static 关键字的作用

53、列举几种进程的同步机制，并比较其优缺点。

54、进程之间通信的途径

55、进程死锁的原因

56、死锁的4个必要条件

57、死锁的处理

58、cocoa touch框架

59、自动释放池是什么,如何工作

60、sprintf,strcpy,memcpy使用上有什么要注意的地方

61、你了解svn,cvs等版本控制工具么?

62、什么是push

63、静态链接库

64、OC三大特性

(1）封装_点语法
(2）继承
(3）多态

65、OC中如何实现多态

66、Objective-C的优缺点

67、对于OC,你认为最大的优点和最大的不足是什么？对于不足之处，现在有没有可用的方法绕过这些不足来实现需求。如果可以话，有没有考虑或者实现过重新实现OC的功能，如果有，具体怎么做？

68、oc中可修改和不可以修改类型

69、我们说的oc是动态运行时语言是什么意思？

70、通知和协议的不同之处？

71、什么是推送消息？

72、关于多态性

73、什么是谓词？

74、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？

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

76、谈谈Object-C的内存管理方式及过程？

77、Object-C有私有方法吗？私有变量呢？

78、说说响应链

79、时间传递 & 响应者链

80、frame和bounds有什么不同?

81、方法和选择器有何不同?

82、OC的垃圾回收机制?

83、什么是延迟加载?

84、是否在一个视图控制器中嵌入两个tableview控制器?

85、一个tableView是否可以关联两个不同的数据源?你会怎么处理?

86、什么时候使用NSMutableArray，什么时候使用NSArray?

87、给出委托方法的实例，并且说出UITableVIew的Data Source方法

88、在应用中可以创建多少autorelease对象，是否有限制?

89、如果我们不创建内存池，是否有内存池提供给我们?

90、什么时候需要在程序中创建内存池?

91、类NSObject的那些方法经常被使用?

92、什么是简便构造方法?

93、如何使用Xcode设计通用应用?

94、 UIView的动画效果有那些?

95、Object-C有多继承吗? 没有的话用什么代替?
cocoa 中所有的类都是NSObject 的子类

96、内存管理 Autorelease、retain、copy、assign的set方法和含义?

97、C和obj-c 如何混用

98、类别的作用?继承和类别在实现中有何区别?

99、类别和类扩展的区别。

100、oc中的协议和java中的接口概念有何不同?

101、深拷贝与前拷贝区别

（1）什么是深拷贝浅拷贝

（2）字符串什么时候使用copy,strong

（3）字符串所在内存区域

（4）mutablecopy和copy @property(copy) NSMutableArray *arr;这样写有什么问题*

（5）如何让自定义类可以使用copy修饰符

102、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?

103、#import 跟#include 又什么区别，@class呢，#import<> 跟 #import”"又什么区别?

104、Objective-C的类可以多重继承么?可以实现多个接口么?Category是什么?重写一个类的方法用继承好还是分类好?为什么?

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

106、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

107、常见的Objective-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

108、id 声明的对象有什么特性?

109、Objective-C如何对内存管理的,说说你的看法和解决方法?

110、原子(atomic)跟非原子(non-atomic)属性有什么区别?

111、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?

112、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?

113、如何对iOS设备进行性能测试?

114、设计模式

（1）mvc模式
（2）单例模式
（3）mvvm模式
（4）观察者模式
（5）工厂模式
（6）代理模式
（7）策略模式
（8）适配器模式
（9）模版模式
（10）外观模式
（11）创建模式
（12）MVP模式

115、MVVM模式原理分析

116、说说常用的几种传值方式

117、什么时候用delegate，什么时候用Notification

118、对于单例的理解

119、从设计模式角度分析代理，通知和KVO区别？ios SDK 提供 的framework使用了哪些设计模式，为什么使用？有哪些好处和坏处?

120、KVO，NSNotification，delegate及block区别

121、运行时（runTime）

122、runtime/消息转发机制

（1）runtime

1.1、什么是runtime
1.2、runtime干什么用，使用场景
(2）消息机制
2.1、消息转发的原理
2.2、SEL isa super cmd 是什么
(3）动态绑定

123、使用bugly进行崩溃分析

124、jenkens 持续打包

125、KVO & KVC

(1）底层实现
(2）KVO概述
(3）KVC概述

126、什么是KVO和KVC?

KVO和KVC

(1）如何调用私有变量，如何修改系统的只读属性，KVC的查找顺序
(2）什么是键-值,键路径是什么
(3）kvo的实现机制
(4）KVO计算属性，设置依赖键
(5）KVO集合属性
(6）kvo使用场景

127、SDWebImage(SDWebImage的实现机制）

(1）主要功能
(2）缓存
(3）内存缓存与磁盘缓存

128、框架 SDWebimage的缓存机制

129、网络安全

密码的安全原则

130、多线程

（1）多线程概念
（2）多线程的作用
（3）使用场景

131、NSOperationQueue和GCD的区别是什么

132、GCD与NSThread的区别

133、进程和线程的区别与联系是什么?

134、别异步执行两个耗时操作，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

135、在项目什么时候选择使用GCD，什么时候选择NSOperation?

136、对比iOS中的多线程技术

137、多线程优缺点

138、iOS中的延迟操作

139、串行队列同步执行和异步主队列

140、资源抢夺解决方案

141、`dispatch_barrier_async`的作用是什么?

142、在多线程Core Data中,NSC,MOC,NSObjectModel哪些需要在线程中创建或者传递?你是用什么策越来实现的?

143、+(void)load与 +(void)initialize区别

load 和 initialize方法的区别

144、http的post与区别与联系，实践中如何选择它们?

145、说说关于UDP/TCP的区别?

146、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?

147、HTTP请求常用的几种方式

148、block

(1）使用block时什么情况会发生引用循环，如何解决?
(2）在block内如何修改block外部变量?
(3）Block & MRC-Block
(4）什么是block
(5）block 实现原理
(6）关于block
(7）使用block和使用delegate完成委托模式有什么优点
(8）多线程与block
(9）谈谈对Block 的理解?并写出一个使用Block执行UIVew动画?
(10）写出上面代码的Block的定义（接上题）

149、Weak、strong、copy、assign 使用

（1）什么情况使用 weak 关键字，相比 assign 有什么不同?
（2）怎么用 copy 关键字?
（3）weak & strong
（4）这个写法会出什么问题： @property (copy) NSMutableArray *array
（5） 如何让自己的类用 copy 修饰符？如何重写带 copy 关键字的 setter?
（6） @property 的本质是什么？ivar、getter、setter 是如何生成并添加到这个类中的
（7）ivar、getter、setter 是如何生成并添加到这个类中的?
（8）用@property声明的NSString（或NSArray，NSDictionary）经常使用copy关键字，为什么？如果改用strong关键字，可能造成什么问题?
（9）@protocol 和 category 中如何使用 @property
（10）runtime如何通过selector找到对应的IMP地址?
（11）retain和copy区别
（12）copy和strong的使用?
（13）NSString和NSMutableString，前者线程安全，后者线程不安全。
（14）readwrite，readonly，assign，retain，copy，weak ,strong,nonatomic 属性的作用

150、OC与JS的交互（iOS与H5混编）

TableView性能优化
UITableView核心思想
UITableView的优化主要从三个方面入手：

151、TableView为什么会卡?

152、UITableView

（1）UITableView最核心的思想
（2）定义高度
（3）自定义高度原理
（4）老生常谈之UITableView的性能优化

(5）cell高度的计算

(5.1）定高的cell和动态高度的cell

(6）TableView渲染

(7）减少视图的数目

(8）减少多余的绘制操作

(9）不要给cell动态添加subView

(10）异步化UI，不要阻塞主线程

(11）滑动时按需加载对应的内容

(12）离屏渲染的问题

(13）离屏渲染优化方案

153、环信SDK使用

154、蓝牙

155、在iPhone应用中如何保存数据?

156、什么是coredata?

157、 什么是NSManagedObject模型?

158、什么是NSManagedobjectContext?

159、 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?

160、CoreData & SQLite3

161、数据存储

(1）数据存储技术

(1.1）数据存储的几种方式

(1.2）各自特点（面试考点）

（1.3）偏好设置（面试考点）
（1.4）归档（面试考点）
（2）数据库技术（SQLite&CoreData）

162、Objective-C堆和栈的区别?

163、内存泄露 & 内存溢出

164、堆 & 栈

（1）堆栈空间分配区别
（2）堆栈缓存方式区别
（3）堆栈数据结构区别

165、内存管理

（1）内存区域
（1.1）堆和栈的区别
（1.2）iOS内存区域
（2）字符串的内存管理
（3）你是如何优化内存管理
（4）循环引用
（5）autorelease的使用
（5.1）工厂方法为什么不释放对象
（5.2）ARC下autorelease的使用场景
（5.3）自动释放池如何工作
（5.4）避免内存峰值
（5.5）ARC和MRC的混用
（5.6）NSTimer的内存管理
（5.7）ARC的实现原理

166、Runloop

167、fmmpeg框架

168、fmdb框架

169、320框架

170、UIKit和CoreAnimation和CoreGraphics的关系是什么？在开发中是否使用过CoreAnimation和CoreGraphics?

171、trasform

172、点讲动画和layer ,view的区别

173、图层与视图

174、平行的层级关系

175、图层的能力

176、使用图层

177、核心绘图

（1）View和layer的区别
（2）new和alloc init的区别

178、动画

179、UICollectionView

（1）何实现瀑布流,流水布局
（2）和UITableView的使用区别

180、UIImage

181、webview

182、描述九宫格算法

183、实现图片轮播图

184、iOS网络框架

185、网络

（1）网络基础
（2）网络传输
（3）AFN

186、AFNetworking & ASIHttpRequest & MKNetWorking

（1）底层实现
（2）对服务器返回的数据处理
（3）监听请求过程
（4）在文件下载和文件上传的使用难易度
（5）网络监控
（6）ASI提供的其他实用功能
（7）MKNetworkKit

187、性能优化

188、算法

————————————————————————————————
————————————————————————————————
————————————————————————————————

1、swift和oc的区别

（1）Swift没有地址/指针的概念

（2）泛型
（3）类型严谨 对比oc的动态绑定

2、编译连接

id和instancetype的区别
instancetype只能做返回值,编译时判断真实类型,不符合发警告
特殊情况: 关联类型返回方法,如类方法alloc或new开头,实例方法中,以autorelease,init,retain,或self开头

3、synthesize & denamic

1：通过@synthesize 指令告诉编译器在编译期间产生getter/setter方法。
2: 通过@dynamic指令，自己实现方法。
有些存取是在运行时动态创建的，如在CoreData的NSManagedObject类使用的某些。如果你想这些情况下，声明和使用属性，但要避免缺少方法在编译时的警告，你可以使用@dynamic动态指令，而不是@synthesize合成指令。

4、在项目开发中常用 的开发工具有哪些?

instrument
beyondCompare
git

5、UITableView & UICollection

UICollectionView是iOS6新引进的API，用于展示集合视图，布局更加灵活，其用法类似于UITableView。而 UICollectionView、UICollectionViewCell与UITableView、UITableViewCell在用法上有相似 的也有不同的，下面是一些基本的使用方法：
对于UITableView，仅需要UITableViewDataSource,UITableViewDelegate这两个协议，使用 UICollectionView需要实现UICollectionViewDataSource,UICollectionViewDelegate，UICollectionViewDelegateFlowLayout这三个协议，这是因为 UICollectionViewDelegateFlowLayout实际上是UICollectionViewDelegate的一个子协议，它继承 了UICollectionViewDelegate，它的作用是提供一些定义UICollectionView布局

模式的函数

6、NSProxy & NSObject

NSObjetct：
NSObject协议组对所有的Object－C下的objects都生效。 如果objects遵从该协议，就会被看作是first－class objects（一级类）。 另外，遵从该协议的objects的retain，release，autorelease等方法也服从objects的管理和在Foundation中 定义的释放方法。一些容器中的对象也可以管理这些objects，比如 说NSArray 和NSDictionary定义的对象。 Cocoa的根类也遵循该协议，所以所有继承NSObjects的objects都有遵循该协议的特性。
NSProXY：
NSProxy 是一个虚基类，它为一些表现的像是其它对象替身或者并不存在的对象定义一套API。一般的，发送给代理的消息被转发给一个真实的对象或者代理本身 load(或者将本身转换成)一个真实的对象。NSProxy的基类可以被用来透明的转发消息或者耗费巨大的对象的lazy 初始化。

7、Object & Swift

Obejective-C复杂的语法，更加简单易用、有未来，让许多开发者心动不已，Swift明显的特点有：
苹果宣称 Swift 的特点是：快速、现代、安全、互动，而且明显优于 Objective-C 语言
可以使用现有的 Cocoa 和 Cocoa Touch 框架
Swift 取消了 Objective C 的指针及其他不安全访问的使用
舍弃 Objective C 早期应用 Smalltalk 的语法，全面改为句点表示法提供了类似 Java 的名字空间(namespace)、泛型(generic)、运算对象重载（operator overloading）Swift 被简单的形容为 “没有 C 的 Objective-C”（Objective-C without the C）为苹果开发工具带来了Xcode Playgrounds功能，该功能提供强大的互动效果，能让Swift源代码在撰写过程中实时显示出其运行结果；
基于C和Objective-C，而却没有C的一些兼容约束；
采用了安全的编程模式；
界面基于Cocoa和Cocoa Touch框架；
保留了Smalltalk的动态特性

8、传值通知 & 推送通知（本地 & 远程）

传值通知：类似通知，代理，Block实现值得传递
推送通知：推送到用户手机对应的App上（主要是不再前台的情况）
本地通知。
local notification，用于基于时间行为的通知，比如有关日历或者todo列表的小应用。另外，应用如果在后台执行，iOS允许它在受限的时间内运行，它也会发现本地通知有用。比如，一个应用，在后台运行，向应用的服务器端获取消息，当消息到达时，比如下载更新版本的提示消息，通过本地通知机制通知 用户。
本地通知是UILocalNotification的实例，主要有三类属性：
scheduled time，时间周期，用来指定iOS系统发送通知的日期和时间；
notification type，通知类型，包括警告信息、动作按钮的标题、应用图标上的badge（数字标记）和播放的声音；
自定义数据，本地通知可以包含一个dictionary类型的本地数据。
对本地通知的数量限制，iOS最多允许最近本地通知数量是64个，超过限制的本地通知将被iOS忽略。
远程通知（需要服务器）。
流程大概是这样的
1.生成CertificateSigningRequest.certSigningRequest文件
2.将CertificateSigningRequest.certSigningRequest上传进developer，导出.cer文件
3.利用CSR导出P12文件
4.需要准备下设备token值（无空格）
5.使用OpenSSL合成服务器所使用的推送证书
一般使用极光推送，步骤是一样的，只是我们使用的服务器是极光的，不需要自己大服务器！

9、第三方库 & 第三方平台

第三方库:一般是指大牛封装好的一个框架（库），或者第三方给我们提供的一个库，这里比较笼统 *第三方平台：指第三方提供的一些服务，其实很多方面跟第三方库是一样的，但是还是存在一些区别。
区别：
库：AFN，ASI，Alomofire，MJRefresh，MJExtension，MBProgressHUD
平台：极光，百度，友盟，Mob，环信

imageName和ImageWithContextOfFile的区别？哪个性能高

（1）用imageNamed的方式加载时，图片使用完毕后缓存到内存中，内存消耗多，加载速度快。即使生成的对象被 autoReleasePool释放了，这份缓存也不释放，如果图像比较大，或者图像比较多，用这种方式会消耗很大的内存。

imageNamed采用了缓存机制，如果缓存中已加载了图片，直接从缓存读就行了，每次就不用再去读文件了，效率会更高 。

（2）ImageWithContextOfFile加载，图片是不会缓存的，加载速度慢。

（3）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.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.

10、NSCache & NSDcitionary

NSCache与可变集合有几点不同：

NSCache类结合了各种自动删除策略，以确保不会占用过多的系统内存。如果其它应用需要内存时，系统自动执行这些策略。当调用这些策略时，会从缓存中删除一些对象，以最大限度减少内存的占用。

NSCache是线程安全的，我们可以在不同的线程中添加、删除和查询缓存中的对象，而不需要锁定缓存区域。

不像NSMutableDictionary对象，一个缓存对象不会拷贝key对象。

NSCache和NSDictionary类似，不同的是系统回收内存的时候它会自动删掉它的内容。

(1)可以存储(当然是使用内存)

(2)保持强应用, 无视垃圾回收. =>这一点同 NSMutableDictionary

(3)有固定客户.

11、 UIView的setNeedsDisplay和setNeedsLayout方法

1、在Mac OS中NSWindow的父类是NSResponder，而在i OS 中UIWindow 的父类是UIVIew。程序一般只有一个窗口但是会又很多视图。

2、UIView的作用：描画和动画，视图负责对其所属的矩形区域描画、布局和子视图管理、事件处理、可以接收触摸事件、事件信息的载体、等

等。
3、UIViewController 负责创建其管理的视图及在低内存的时候将他们从内存中移除。还为标准的系统行为进行响应。
4、layOutSubViews 可以在自己定制的视图中重载这个方法，用来调整子视图的尺寸和位置。
5、UIView的setNeedsDisplay(需要重新显示,绘制)和setNeedsLayout(需要重新布局)方法。首先两个方法都是异步执行的。而 setNeedsDisplay会调用自动调用drawRect方法，这样可以拿到UIGraphicsGetCurrentContext，就可以画画 了。而setNeedsLayout会默认调用layoutSubViews，就可以处理子视图中的一些数据。
综上所述：setNeedsDisplay方便绘图，而layoutSubViews方便出来数据
setNeedDisplay告知视图它发生了改变，需要重新绘制自身，就相当于刷新界面.

12、UILayer & UIView

UIView是iOS系统中界面元素的基础，所有的界面元素都继承自它。它本身完全是由CoreAnimation来实现的（Mac下似乎不是这 样）。它真正的绘图部分，是由一个叫CALayer（Core Animation Layer）的类来管理。
UIView本身，更像是一个CALayer的管理器，访问它的跟绘图和跟坐标有关的属性，例如frame，bounds等等， 实际上内部都是在访问它所包含的CALayer的相关属性。
UIView有个重要属性layer，可以返回它的主CALayer实例。
UIView的CALayer类似UIView的子View树形结构，也可以向它的layer上添加子layer，来完成某些特殊的表示。即CALayer层是可以嵌套的。
UIView的layer树形在系统内部，被维护着三份copy。分别是逻辑树，这里是代码可以操纵的；动画树，是一个中间层，系统就在这一层上更改属性，进行各种渲染操作；显示树，其内容就是当前正被显示在屏幕上得内容。
动画的运作：对UIView的subLayer（非主Layer）属性进行更改，系统将自动进行动画生成，动画持续时间的缺省值似乎是0.5秒。
坐标系统：CALayer的坐标系统比UIView多了一个anchorPoint属性，使用CGPoint结构表示，值域是0~1，是个比例值。
渲染：当更新层，改变不能立即显示在屏幕上。当所有的层都准备好时，可以调用setNeedsDisplay方法来重绘显示。

变换：要在一个层中添加一个3D或仿射变换，可以分别设置层的transform或affineTransform属性。
变形：Quartz Core的渲染能力，使二维图像可以被自由操纵，就好像是三维的。图像可以在一个三维坐标系中以任意角度被旋转，缩放和倾斜。CATransform3D的一套方法提供了一些魔术般的变换效果。

13、layoutSubViews & drawRects

layoutSubviews在以下情况下会被调用(视图位置变化是触发)：
1、init初始化不会触发layoutSubviews。
2、addSubview会触发layoutSubviews。
3、设置view的Frame会触发layoutSubviews，当然前提是frame的值设置前后发生了变化。
4、滚动一个UIScrollView会触发layoutSubviews。
5、旋转Screen会触发父UIView上的layoutSubviews事件。
6、改变一个UIView大小的时候也会触发父UIView上的layoutSubviews事件。
7、直接调用setLayoutSubviews。
drawRect在以下情况下会被调用：
1、如果在UIView初始化时没有设置rect大小，将直接导致drawRect不被自动调用。drawRect 掉用是在Controller->loadView, Controller->viewDidLoad 两方法之后掉用的.所以不用担心在 控制器中,这些View的drawRect就开始画了.这样可以在控制器中设置一些值给View(如果这些View draw的时候需要用到某些变量值).
2、该方法在调用sizeToFit后被调用，所以可以先调用sizeToFit计算出size。然后系统自动调用drawRect:方法。
3、通过设置contentMode属性值为UIViewContentModeRedraw。那么将在每次设置或更改frame的时候自动调用drawRect:。
4、直接调用setNeedsDisplay，或者setNeedsDisplayInRect:触发drawRect:，但是有个前提条件是rect不能为0。

drawRect方法使用注意点：
1、 若使用UIView绘图，只能在drawRect：方法中获取相应的contextRef并绘图。如果在其他方法中获取将获取到一个invalidate 的ref并且不能用于画图。drawRect：方法不能手动显示调用，必须通过调用setNeedsDisplay 或 者 setNeedsDisplayInRect，让系统自动调该方法。

2、若使用calayer绘图，只能在drawInContext: 中（类似鱼drawRect）绘制，或者在delegate中的相应方法绘制。同样也是调用setNeedDisplay等间接调用以上方法 3、若要实时画图，不能使用gestureRecognizer，只能使用touchbegan等方法来掉用setNeedsDisplay实时刷新屏幕

14、UDID & UUID

UDID是Unique Device Identifier的缩写,中文意思是设备唯一标识.
在很多需要限制一台设备一个账号的应用中经常会用到,在Symbian时代,我们是使用IMEI作为设备的唯一标识的,可惜的是Apple官方不允许开发者获得设备的IMEI.
[UIDevice currentDevice] uniqueIdentifier]
但是我们需要注意的一点是,对于已越狱了的设备,UDID并不是唯一的.使用Cydia插件UDIDFaker,可以为每一个应用分配不同的UDID. 所以UDID作为标识唯一设备的用途已经不大了.
UUID是Universally Unique Identifier的缩写,中文意思是通用唯一识别码.
由网上资料显示,UUID是一个软件建构的标准,也是被开源软件基金会(Open Software Foundation,OSF)的组织在分布式计算环境(Distributed Computing Environment,DCE)领域的一部份.UUID的目的,是让分布式系统中的所有元素,都能有唯一的辨识资讯,而不需要透过中央控制端来做辨识资 讯的指定.

15、CPU & GPU

CPU:中央处理器（英文Central Processing Unit）是一台计算机的运算核心和控制核心。CPU、内部存储器和输入/输出设备是电子计算机三大核心部件。其功能主要是解释计算机指令以及处理计算机软件中的数据。
GPU:英文全称Graphic Processing Unit，中文翻译为“图形处理器”。一个专门的图形核心处理器。GPU是显示卡的“大脑”，决定了该显卡的档次和大部分性能，同时也是2D显示卡和3D 显示卡的区别依据。2D显示芯片在处理3D图像和特效时主要依赖CPU的处理能力，称为“软加速”。3D显示芯片是将三维图像和特效处理功能集中在显示芯 片内，也即所谓的“硬件加速”功能

16、点（pt）&像素（px）

像素（pixels）是数码显示上最小的计算单位。在同一个屏幕尺寸，更高的PPI（每英寸的像素数目），就能显示更多的像素，同时渲染的内容也会更清晰。
点（points）是一个与分辨率无关的计算单位。根据屏幕的像素密度，一个点可以包含多个像素（例如，在标准Retina显示屏上1 pt里有2 x 2个像素）。
当你为多种显示设备设计时，你应该以“点”为单位作参考，但设计还是以像素为单位设计的。这意味着仍然需要以3种不同的分辨率导出你的素材，不管你以哪种分辨率设计你的应用。

17、属性与成员变量

成员变量是不与外界接触的变量，应用于类的内部，如果你说那用@Public外部不就是可以访问了么。简单的说public只能适当使用，不要泛滥，否则就像你把钥匙插在你自己家门上了。谁来都可以开门。毫无安全性。
由于成员变量的私有性，为了解决外部访问的问题就有了属性变量。属性变量个人认为最大的好处就是让其他对象访问这个变量。而且你可以设置只读、可写等等属性，同时设置的方法我们也可以自己定义。记住一点，属性变量主要是用于与其他对象相互交互的变量
如果对于上面所说还是含糊不清那就记住这几点吧！
1.只有类内使用，属性为private，那么就定义成员变量。
2.如果你发现你需要的这个属性需要是public的，那么毫不犹豫就用属性在.h中定义。
3.当你自己内部需要setter实现一些功能的时候，用属性在.m中定义。
4.当你自己内部需要getter实现一些功能的时候，用属性在.m中定义

18、int和NSInteger的区别

SInteger表示当前cpu下整型所占最大字节,不同CPU的long型所占字节不同,32位int4 long4,64位int4,long8

（1）import和include

import可以避免重复包含

（2）@class

避免循环引用头文件

（3）全局 & 静态变量

全局变量和静态变量的区别
1> 修饰符
全局变量在声明源文件之外使用,需要extern引用一下;
静态变量使用static来修饰
2> 存储地址
两者都是存储在静态存储区,非堆栈上,它们与局部变量的存储分开
3> 生命周期
两者都是在程序编译或加载时由系统自动分配的,程序结束时消亡
4> 外部可访问性
全局变量在整个程序的任何地方均可访问,而静态变量相当于面向对象中的私有变量,他的可访问性只限定于声明它的那个源文件,即作用于仅局限于本文件中

19、类和对象

1.category

（1）分类 拓展 协议中哪些可以声明属性?

都可以,但分类和协议创建的属性只相当于方法,但是内部没有对成员变量的操作(无法创建成员变量),拓展可以(私有成员变量)
代理中声明属性,没有实际创建成员变量,相当于声明了属性名对应的访问方法,遵守协议的类需要实现对应的访问器方法,否则运行报错
分类中声明属性,警告提示需要手动实现访问器方法(Swift中叫计算型属性),而分类中不能创建成员变量,可以在手写访问器方法中使用runtime的objc_setAssociatedObject方法关联对象间接创建属性(静态库添加属性)
拓展里可以声明属性,直接可以使用

（2）继承和类别的区别

1> 使用继承:
1.1> 添加新方法和父类方法一致,但父类方法仍需要使用
1.2> 添加新属性
2> 类别:
2.1> 针对系统提供的一些类,系统本身不提倡继承,因为这些类的内部实现对继承有所限制(NSString initWithFormat继承崩溃)
2.2> 类别可以将自己构建的类中的方法进行分组,对于大型的类,提高可维护性

（3）分类的作用

将类的实现分散到多个不同文件或多个不同框架中。
创建对私有方法的前向引用。
向对象添加非正式协议。
(非正式协议:即NSObject的分类,声明方法可以不实现,OC2.0以前protocal没有@optional,主要使用分类添加可选协议方法
oc中声明方法不实现,不调用则只警告不报错
正式协议的优点:可继承,泛型约束
如kvo的observeValueForKeyPath属于nsobject的分类,且不需要调父类,说明可选实现该方法,没警告可能是编译器规则过滤)

（4）分类的局限性

无法向类中添加新的实例变量，类别没有位置容纳实例变量。
名称冲突，即当类别中的方法与原始类方法名称冲突时，类别具有更高的优先级。类别方法将完全取代初始方法从而无法再使用初始方法。
无法添加实例变量的局限可以使用字典对象解决。

2.extension

3.protocol

20、category & extension

类别主要有三个作用
(1)可以将类的实现分散到多个不同文件或多个不同框架中，方便代码管理。也可以对框架提供类的扩展（没有源码，不能修改）。
(2)创建对私有方法的前向引用：如果其他类中的方法未实现，在你访问其他类的私有方法时编译器报错这时使用类别，在类别中声明这些方法（不必提供方法实现），编译器就不会再产生警告
(3)向对象添加非正式协议：创建一个NSObject的类别称为“创建一个非正式协议”，因为可以作为任何类的委托对象使用。
他们的主要区别是:
1、形式上来看，extension是匿名的category。
2、extension里声明的方法需要在mainimplementation中实现，category不强制要求。
3、extension可以添加属性（变量），category不可以。
Category和Extension都是用来给已定义的类增加新的内容的。
Category和原有类的耦合更低一些，声明和实现都可以写在单独的文件里。但是只能为已定义类增加Method，而不能加入instance variable。

Extension耦合比较高，声明可以单独写，但是实现必须写在原有类的@implementation中。可以增加Method和instance variable。
Extension给人感觉更像是在编写类时为了封装之类的特性而设计，和类是同时编写的。而category则是在用到某一个framework中的类时临时增加的特性。
Extension的一个特性就是可以redeclare一个instance variable，将之从readonly改为对内readwrite.
使用Extension可以更好的封装类，在h文件中能看到的都是对外的接口，其余的instance variable和对内的@property等都可以写在Extension，这样类的结构更加清晰。

21、Foundation

（1）字符串

1> 字符串比较
NSString *a = @“hello”;
NSString *b = [NSString stringWithFormat:@hello”];
if (a == b){
nslog(@“a==b”); }
if ([a isEqualToString: b]){
nslog(@“a isEqualToString b”); }
== 比较变量中保存的数值(地址) 速度快 内容同,可能地址不同(常量区,堆区)
isEqualTo 比较字符串 非常耗时

（2）字符串截取

截取字符串”20 | http://www.baidu.com”中,”|”字符前面和后面的数据,分别输出它们。

```
NSString * str = @"20 | http://www.baidu.com";
NSArray *array = [str componentsSeparatedByString:@"|"]; //这是分别输出的截取后的字符串
for (int i = 0; i<[array count]; ++i) {
NSLog(@"%d=%@",i,[array objectAtIndex:i]);
```

（3）格式

```
<p>NSString *str1 = [NSString stringWithFormat:@<q>a</q>b”];
```

```
  //报错，a”后加b非法<br/>
NSString *str2 = [NSString stringWithFormat:@“a&quot;<q>b”];
 //显示  ab<br/>
NSString *str3 = [NSString stringWithFormat:@“a\</q>b”];   //显
示  a”b  反斜杠转义</p>
```

22、NSArray和NSDictionary

（1）iOS遍历数组/字典的方法

数组: for循环 for in enumerateObjectsUsingBlock(正序) enumerateObjectsWithOptions:usingBlock:(多一个遍历选项,不保证顺序)

字典:

1. for(NSString *object in [testDic allValues])*

2. for(id akey in [testDic allKeys]){

[sum appendString:[testDic objectForKey:akey]]; }

*3. [testDic enumerateKeysAndObjectsUsingBlock:dkey,idobj,BOOL*stop) {

[sum appendString:obj]; }];

速度: 对于数组, 增强for最快,普通for和block速度差不多,增强最快是因为增强for语法会对容器里的元素的内存地址建立缓冲,遍历的时候直接从缓冲中取元素地址而不是通过调用方法来获取,所以效率高.这也是使用增强for时不能在循环体中修改容器元素的原因之一(可以在循环体中添加标记,在循环体外修改元素)

对于字典,allValues最快,allKey和block差不多,原因是allKey需要做objcetForKey的方法

（2）NSValue NSNumber

1> 归档视图尺寸，坐标

（3）其它

nil Nil null NSNull 的区别

（4）如何避免循环引用

两个对象相互强引用，都无法release，解决办法为一个使用strong，一个使用assign（weak）

23、CFSocket使用有哪几个步骤

答：创建 Socket 的上下文；创建 Socket ；配置要访问的服务器信息；封

装服务器信息；连接服务器；

24、Core Foundation中提供了哪几种操作Socket的方法?

答： CFNetwork 、 CFSocket 和 BSD Socket 。

25、解析XML文件有哪几种方式?

答：以 DOM 方式解析 XML 文件；以 SAX 方式解析 XML 文件；

26、什么是沙盒模型？哪些操作是属于私有api范畴?

答：某个iphone工程进行文件操作有此工程对应的指定的位置，不能逾越。
iphone沙箱模型的有四个文件夹documents，tmp，app，Library，永久数据存储一般放documents文件夹，得到模拟器的路径的可使用NSHomeDirectory()方法。Nsuserdefaults保存的文件在tmp文件夹里。

27、在一个对象的方法里面：self.name= “object”；和 name =”object” 有什么不同吗?

答：self.name =”object”：会调用对象的setName()方法；
name = “object”：会直接把object赋值给当前对象的name属性。

28、请简要说明viewDidLoad和viewDidUnload何时调用

答：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 。

29、创建控制器、视图的方式

1> 创建控制器的方式

（1）通过代码的方式加载viewController

```
UIViewController *controller = [[UIViewController alloc] init];
```

（2）通过stroyboard来加载viewController

（2.1) 加载storyboard中箭头指向的viewController

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil]; //加载箭头指向的viewController
CZViewController *controller = [storyboard instantiateInitialViewController];
```

（2.2) 加载storyboard中特定标示的viewController(storyboard可以有多个controller)

```
UIStoryboard *storyboard = [UIStoryboard storyboardWithName:@"Main" bundle:nil];
CZViewController *controller = [storyboard instantiateViewControllerWithIdentifier:@"two"];
```

（3）通过xib加载viewController

（3.1) 传统方法

（3.1.1）创建Xib,并指定xib的files owner为自定义控制器类(为了能连线关联管理IB的内容)

（3.1.2）xib中要有内容，且xib中描述的控制器类的view属性要与xib的view控件完成关联（关联方法两种,一种是control+files owner拖线到xib中搭建的指定view控件,另一种是指定xib中的view拖线到@interface）

（3.1.3）从xib加载viewController

```
CZViewController *controller = [[CZViewController alloc] initWithNibName:@"CZOneView" bundle:nil];
```

（3.2）bundle中取出xib内容

```
CZViewController *vc = [[NSBundle mainBundle] loadNibNamed:@"Two" owner:nil options:nil].lastObject;
```

2> 创建视图的方式

1.用系统的loadView方法创建控制器的视图

2.如果指定加载某个storyboard文件做控制器的视图，就会加载storyboard里面的描述去创建view

3.如果指定读取某个xib文件做控制器的视图，就根据指定的xib文件去加载创建

4.如果有xib文件名和控制器的类名前缀（也就是去掉controller）的名字一样的 xib文件 就会用这个xib文件来创建控件器的视图 例：控件器的名为 MJViewController xib文件名为 MJView.xib 如果xib文件名后有一个字不一

样就不会去根据它去创建如：MJView8.xib
5.找和控制器同名的xib文件去创建
6.如果以上都没有就创建一个空的控制器的视图;

UIWindow

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

30、简述内存分区情况

答:
1).代码区：存放函数二进制代码
2).数据区：系统运行时申请内存并初始化，系统退出时由系统释放。存放全局变量、静态变量、常量
3).堆区：通过malloc等函数或new等操作符动态申请得到，需程序员手动申请和释放
4).栈区：函数模块内申请，函数结束时由系统自动释放。存放局部变量、函数参数

31、队列和栈有什么区别

答：队列和栈是两种不同的数据容器。从”数据结构”的角度看，它们都是线性结构，即数据元素之间的关系相同。
队列是一种先进先出的数据结构，它在两端进行操作，一端进行入队列操作，一端进行出列队操作。
栈是一种先进后出的数据结构，它只能在栈顶进行操作，入栈和出栈都在栈顶操作。

32、iOS的系统架构

答： iOS的系统架构分为（核心操作系统层 theCore OS layer）、（核心服务层theCore Services layer）、（媒体层 theMedia layer）和（Cocoa界面服务层 the Cocoa Touch layer）四个层次。

33、控件主要响应3种事件

答：1). 基于触摸的事件；2). 基于值的事件；3).基于编辑的事件。

34、xib文件的构成分为哪3个图标？都具有什么功能

答： File's Owner 是所有 nib 文件中的每个图标，它表示从磁盘加载 nib 文件的对象；
First Responder 就是用户当前正在与之交互的对象；
View 显示用户界面；完成用户交互；是 UIView 类或其子类。

35、简述视图控件器的生命周期

答： loadView 尽管不直接调用该方法，如多手动创建自己的视图，那么应该覆盖这个方法并将它们赋值给试图控制器的 view 属性。
viewDidLoad 只有在视图控制器将其视图载入到内存之后才调用该方法，这是执行任何其他初始化操作的入口。
viewDidUnload 当试图控制器从内存释放自己的方法的时候调用，用于清楚那些可能已经在试图控制器中创建的对象。
viewVillAppear 当试图将要添加到窗口中并且还不可见的时候或者上层视图移出图层后本视图变成顶级视图时调用该方法，用于执行诸如改变视图方向等的操作。实现该方法时确保调用 [super viewWillAppear:
viewDidAppear 当视图添加到窗口中以后或者上层视图移出图层后本视图变成顶级视图时调用，用于放置那些需要在视图显示后执行的代码。确保调用 [super viewDidAppear：] 。
viewWillDisappear－UIViewController对象的视图即将消失、被覆盖或是隐藏时调用；
viewDidDisappear－UIViewController对象的视图已经消失、被覆盖或是隐藏时调用；
viewVillUnload－当内存过低时，需要释放一些不需要使用的视图时，即将释放时调用；
viewDidUnload－当内存过低，释放一些不需要的视图时调用。

36、app 项目的生命周期

（1）应用的生命周期

各个程序运行状态时代理的回调：

-(BOOL)application:(UIApplication*)application willFinishLaunchingWithOptions:(NSDictionary *)*launchOptions 告诉代理进程启动但还没进入状态保存*

-(BOOL)application:(UIApplication)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions 告诉代理启动基本完成程序准备开始运行

- (void)applicationWillResignActive:(UIApplication *)application 当应用程序将要入非活动状态执行，在此期间，应用程序不接收消息或事件，比如来电话了

- (void)applicationDidBecomeActive:(UIApplication *)application 当应用程序入活动状态执行，这个刚好跟上面那个方法相反

- (void)applicationDidEnterBackground:(UIApplication *)application 当程序被推送到后台的时候调用。所以要设置后台继续运行，则在这个函数里面设置即可

- (void)applicationWillEnterForeground:(UIApplication *)application 当程序从后台将要重新回到前台时候调用，这个刚好跟上面的那个方法相反。

- (void)applicationWillTerminate:(UIApplication *)application 当程序将要退出是被调用，通常是用来保存数据和一些退出前的清理工作。

（2）简要说明一下APP的启动过程，main文件说起，main函数中有什么函数？作用是什么？

http://www.jianshu.com/p/3f262ae413b4

打开程序——->执行main函数———>UIAPPlicationMain函数——->初始化UIAPPlicationMain函数(设置代理，开启事件循环)———>监听系统事件——->程序结束

先执行main函数，main内部会调用UIApplicationMain函数

（3）UIApplicationMain函数作用

(1)、根据传入的第三个参数创建UIApplication对象或它的子类对象。如果该参数为nil,直接使用该UIApplication来创建。(该参数只能传人UIApplication或者是它的子类)

(2)、根据传入的第四个参数创建AppDelegate对象,并将该对象赋值给第1步创建的UIApplication对象的delegate属性。

(3)、开启一个事件循环,循环监控应用程序发生的事件。每监听到对应的系统事件时，就会通知AppDelegate。

（4）main函数作用

（1）创建UIApplication对象

（2）创建应用程序代理
（3）开启时间循环，包括应用程序的循环运行，并开始处理用户事件。

37、 动画有基本类型有哪几种；表视图有哪几种基本样式。

答：动画有两种基本类型：隐式动画和显式动画。

38、实现简单的表格显示需要设置UITableView的什么属性、实现什么协议?

答：实现简单的表格显示需要设置 UITableView 的 dataSource 和 delegate 属性，实现UITableViewDataSource 和 UITableViewDelegate 协议。

39、Cocoa Touch提供了哪几种Core Animation过渡类型?

答： Cocoa Touch 提供了 4 种 Core Animation 过渡类型，分别为：交叉淡化、推挤、显示和覆盖。

40、UIView与CLayer有什么区别?

答：
1).UIView 是 iOS 系统中界面元素的基础，所有的界面元素都是继承自它。它本身完全是由 CoreAnimation 来实现的。它真正的绘图部分，是由一个 CALayer 类来管理。 UIView 本身更像是一个 CALayer 的管理器，访问它的跟绘图和跟坐标有关的属性。
2).UIView 有个重要属性 layer ， 可以返回它的主 CALayer 实例。
3).UIView 的 CALayer 类似 UIView 的子 View 树形结构，也可以向它的 layer 上添加子layer ， 来完成某些特殊的表示。即 CALayer 层是可以嵌套的。
4).UIView 的 layer 树形在系统内部，被维护着三份 copy 。分别是逻辑树，这里是代码可以操纵的；动画树，是一个中间层，系统就在这一层上更改属性，进行各种渲染操作；显示树，其内容就是当前正被显示在屏幕上得内容。
5).动画的运作：对 UIView 的 subLayer （非主 Layer ）属性进行更改，系

统将自动进行动画生成，动画持续时间的缺省值似乎是 0.5 秒。
6).坐标系统： CALayer 的坐标系统比 UIView 多了一个 anchorPoint 属性，使用CGPoint 结构表示，值域是 0~1 ，是个比例值。这个点是各种图形变换的坐标原点，同时会更改 layer 的 position 的位置，它的缺省值是 {0.5,0.5} ，即在 layer 的中央。
7).渲染：当更新层，改变不能立即显示在屏幕上。当所有的层都准备好时，可以调用setNeedsDisplay 方法来重绘显示。
8).变换：要在一个层中添加一个 3D 或仿射变换，可以分别设置层的 transform 或affineTransform 属性。
9).变形： Quartz Core 的渲染能力，使二维图像可以被自由操纵，就好像是三维的。图像可以在一个三维坐标系中以任意角度被旋转，缩放和倾斜。 CATransform3D 的一套方法提供了一些魔术般的变换效果。

41、Quatrz 2D的绘图功能的三个核心概念是什么并简述其作用

答：上下文：主要用于描述图形写入哪里；
路径：是在图层上绘制的内容；
状态：用于保存配置变换的值、填充和轮廓， alpha 值等。

42、iPhone OS主要提供了几种播放音频的方法?

答： SystemSound Services
AVAudioPlayer 类
Audio Queue Services
OpenAL

43、使用AVAudioPlayer类调用哪个框架、使用步骤?

答： AVFoundation.framework
步骤：配置 AVAudioPlayer 对象；
实现 AVAudioPlayer 类的委托方法；
控制 AVAudioPlayer 类的对象；
监控音量水平；
回放进度和拖拽播放。

44、有哪几种手势通知方法、写清楚方法名?

答:

-(void)touchesBegan:(NSSet*)touchedwithEvent:(UIEvent*)event;

-(void)touchesMoved:(NSSet*)touched withEvent:(UIEvent*)event;

-(void)touchesEnded:(NSSet*)touchedwithEvent:(UIEvent*)event;

-(void)touchesCanceled:(NSSet*)touchedwithEvent:(UIEvent*)event;

-(void)touchesBegan:(NSSet*)touchedwithEvent:(UIEvent*)event;

-(void)touchesMoved:(NSSet*)touched withEvent:(UIEvent*)event;

-(void)touchesEnded:(NSSet*)touchedwithEvent:(UIEvent*)event;

-(void)touchesCanceled:(NSSet*)touchedwithEvent:(UIEvent*)event;

45、ViewController的didReceiveMemoryWarning怎么被调用

答:[super didReceiveMemoryWarning];

46、什么时候用delegate,什么时候用Notification?

答: delegate针对one-to-one关系，用于sender接受到reciever的某个功能反馈值。

notification针对one-to-one/many/none,reciver,用于通知多个object某个事件。

47、用预处理指令#define声明一个常数，用以表明1年中有多少秒（忽略闰年问题）

答:

```
#define SECONDS_PER_YEAR (60 * 60 * 24 * 365)UL
```

```
#define SECONDS_PER_YEAR (60 * 60 * 24 * 365)UL
```

我在这想看到几件事情：
`#define` 语法的基本知识（例如：不能以分号结束，括号的使用，等等）
懂得预处理器将为你计算常数表达式的值，因此，直接写出你是如何计算一年中有多少秒而不是计算出实际的值，是更清晰而没有代价的。
意识到这个表达式将使一个16位机的整型数溢出-因此要用到长整型符号L，告诉编译器这个常数是的长整型数。
如果你在你的表达式中用到UL（表示无符号长整型），那么你有了一个好的起点。记住，第一印象很重要。

48、写一个”标准"宏MIN ，这个宏输入两个参数并返回较小的一个。

答：

```
#define MIN(A,B)  ( (A)  <= (B) ? (A) : (B))
```


```
#define MIN(A,B)  ( (A)  <= (B) ? (A) : (B))
```


这个测试是为下面的目的而设的：
标识#define在宏中应用的基本知识。这是很重要的，因为直到嵌入(inline)操作符变为标准C的一部分，宏是方便产生嵌入代码的唯一方
法，
对于嵌入式系统来说，为了能达到要求的性能，嵌入代码经常是必须的方法。
三重条件操作符的知识。这个操作符存在C语言中的原因是它使得编译器能产生比 if-then-else 更优化的代码，了解这个用法是很重要的。
懂得在宏中小心地把参数用括号括起来
我也用这个问题开始讨论宏的副作用，例如：当你写下面的代码时会发生什么事?
least = MIN(*p++, b);*
*least = MIN(*p++, b);
结果是：
((*p++) <= (b) ? (p++) : (p++))*
((p++) <= (b) ? (p++) : (p++))
这个表达式会产生副作用，指针p会作三次++自增操作。

49、关键字const有什么含意？修饰类呢?static的作用，

用于类呢?还有extern c的作用

答:
const 意味着"只读"，下面的声明都是什么意思?
const int a;
int const a;
const int *a;
int * const a;
int const * a const
const int a;
int const a;
const int *a;
int * const a;
int const * a const;
前两个的作用是一样，a是一个常整型数。
第三个意味着a是一个指向常整型数的指针（也就是，整型数是不可修改的，但指针可以）。
第四个意思a是一个指向整型数的常指针（也就是说，指针指向的整型数是可以修改的，但指针是不可修改的）。
最后一个意味着a是一个指向常整型数的常指针（也就是说，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）。
结论:
关键字const的作用是为给读你代码的人传达非常有用的信息，实际上，声明一个参数为常量是为了告诉了用户这个参数的应用目的。
如果你曾花很多时间清理其它人留下的垃圾，你就会很快学会感谢这点多余的信息。（当然，懂得用const的程序员很少会留下的垃圾让别人来清理的） 通过给优化器一些附加的信息，使用关键字const也许能产生更紧凑的代码。合理地使用关键字const可以使编译器很自然地保护那些不希望被改变的参数，防止其被无意的代码修改。简而言之，这样可以减少bug的出现。
1).欲阻止一个变量被改变，可以使用 const 关键字。在定义该 const 变量时，通常需要对它进行初
始化，因为以后就没有机会再去改变它了;
2).对指针来说，可以指定指针本身为 const，也可以指定指针所指的数据为 const，或二者同时指

定为 const；
3).在一个函数声明中，const 可以修饰形参，表明它是一个输入参数，在函数内部不能改变其值；
4).对于类的成员函数，若指定其为 const 类型，则表明其是一个常函数，不能修改类的成员变量；
5).对于类的成员函数，有时候必须指定其返回值为 const 类型，以使得其返回值不为“左值”。

50、关键字volatile有什么含意?并给出三个不同的例子

答：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

51、一个参数既可以是const还可以是volatile吗？ 一个指针可以是volatile 吗？解释为什么。

答：1).是的。一个例子是只读的状态寄存器。它是volatile因为它可能被意想不到地改变。它是const因为程序不应该试图去修改它。
2).是的。尽管这并不很常见。一个例子是当一个中服务子程序修该一个指向一个buffer的指针时。

52、static 关键字的作用

答：
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。

53、列举几种进程的同步机制，并比较其优缺点。

答： 原子操作 信号量机制 自旋锁 管程，会合，分布式系统

54、进程之间通信的途径

答：共享存储系统消息传递系统管道：以文件系统为基础

55、进程死锁的原因

答：资源竞争及进程推进顺序非法

56、死锁的4个必要条件

答：互斥、请求保持、不可剥夺、环路

57、死锁的处理

答：鸵鸟策略、预防策略、避免策略、检测与解除死锁

58、cocoa touch框架

答：iPhone OS 应用程序的基础 Cocoa Touch 框架重用了许多 Mac 系统的成熟模式，但是它更多地专注于触摸的接口和优化。
UIKit 为您提供了在 iPhone OS 上实现图形，事件驱动程序的基本工具，其建立在和 Mac OS X 中一样的 Foundation 框架上，包括文件处理，网络，字符串操作等。
Cocoa Touch 具有和 iPhone 用户接口一致的特殊设计。有了 UIKit，您可以使用 iPhone OS 上的独特的图形接口控件，按钮，以及全屏视图的功

能，您还可以使用加速仪和多点触摸手势来控制您的应用。
各色俱全的框架 除了UIKit 外，Cocoa Touch 包含了创建世界一流 iPhone 应用程序需要的所有框架，从三维图形，到专业音效，甚至提供设备访问 API 以控制摄像头，或通过 GPS 获知当前位置。
Cocoa Touch 既包含只需要几行代码就可以完成全部任务的强大的 Objective-C 框架，也在需要时提供基础的 C 语言 API 来直接访问系统。
这些框架包括：
Core Animation：通过 Core Animation，您就可以通过一个基于组合独立图层的简单的编程模型来创建丰富的用户体验。
Core Audio：Core Audio 是播放，处理和录制音频的专业技术，能够轻松为您的应用程序添加强大的音频功能。
Core Data：提供了一个面向对象的数据管理解决方案，它易于使用和理解，甚至可处理任何应用或大或小的数据模型。
功能列表：框架分类
下面是 Cocoa Touch 中一小部分可用的框架：
音频和视频：Core Audio ， OpenAL ， Media Library ， AV Foundation
数据管理 ： Core Data ， SQLite
图形和动画 ： Core Animation ， OpenGL ES ， Quartz 2D
网络：Bonjour ， WebKit ， BSD Sockets
用户应用：Address Book ， Core Location ， Map Kit ， Store Kit

59、自动释放池是什么,如何工作

答：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.它仍然是个正当的对象，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。

60、sprintf,strcpy,memcpy使用上有什么要注意的地方

答：
1). sprintf是格式化函数。将一段数据通过特定的格式，格式化到一个字符串缓冲区中去。sprintf格式化的函数的长度不可控，有可能格式化后的字符串会超出缓冲区的大小，造成溢出。
2).strcpy是一个字符串拷贝的函数，它的函数原型为strcpy(char *dst,

const char *src
将src开始的一段字符串拷贝到dst开始的内存中去，结束的标志符号为‘\0'，由于拷贝的长度不是由我们自己控制的，所以这个字符串拷贝很容易出错。
3). memcpy是具备字符串拷贝功能的函数，这是一个内存拷贝函数，它的函数原型为memcpy(char *dst, const char* src, unsigned int len);将长度为len的一段内存，从src拷贝到dst中去，这个函数的长度可控。但是会有内存叠加的问题。

61、你了解svn,cvs等版本控制工具么?

答： 版本控制 svn,cvs 是两种版控制的器,需要配套相关的svn，cvs服务器。
scm是xcode里配置版本控制的地方。版本控制的原理就是a和b同时开发一个项目，a写完当天的代码之后把代码提交给服务器，b要做的时候先从服务器得到最新版本，就可以接着做。 如果a和b都要提交给服务器，并且同时修改了同一个方法，就会产生代码冲突，如果a先提交，那么b提交时，服务器可以提示冲突的代码，b可以清晰的看到，并做出相应的修改或融合后再提交到服务器。

62、什么是push

答： 客户端程序留下后门端口，客户端总是监听针对这个后门的请求，于是 服务器可以主动像这个端口推送消息。

63、静态链接库

答：此为.a文件，相当于java里的jar包，把一些类编译到一个包中，在不同的工程中如果导入此文件就可以使用里面的类，具体使用依然是#import “ xx.h”。

64、OC三大特性

（1）封装_点语法
1> 本质
//以下代码有什么问题
- (void)setName:(NSString *)name {

```
self.name = name;

}
- (NSString *)name {
return self.name;
}
```

点语法的本质是调用类的getter方法和setter方法，如果类中没有getter方法和setter方法就不能使用点语法。

（2）继承

1> 如何实现多重继承

消息转发

forwardingTargetForSelector methodSignatureForSelector forwardInvocation

delegate和protocol

类别

http://www.cocoachina.com/ios/20130528/6295.html

（3）多态

1> 什么是多态

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。

由于每个类都属于该类的名字空间，这使得多态称为可能。类定义中的名字和类定义外的名字并不会冲突。类的实例变量和类方法有如下特点：

• 和C语言中结构体中的数据成员一样，类的实例变量也位于该类独有的名字空间。

• 类方法也同样位于该类独有的名字空间。与C语言中的方法名不同，类的方法名并不是一个全局符号。一个类中的方法名不会和其他类中同样的方法名冲突。两个完全不同的类可以实现同一个方法。

方法名是对象接口的一部分。对象收到的消息的名字就是调用的方法的名字。因为不同的对象可以有同名的方法，所以对象必须能理解消息的含义。同样的消息发给不同的对象，导致的操作并不相同。

多态的主要好处就是简化了编程接口。它容许在类和类之间重用一些习惯性的命名，而不用为每一个新加的函数命名一个新名字。这样，编程接口

就是一些抽象的行为的集合，从而和实现接口的类区分开来。
Objective-C支持方法名的多态，但不支持参数和操作符的多态。

65、OC中如何实现多态

在Objective-C中是通过一个叫做selector的选取器实现的。在Objective-C中，selector有两个意思， 当用在给对象的源码消息时，用来指方法的名字。它也指那个在源码编译后代替方法名的唯一的标识符。 编译后的选择器的类型是SEL有同样名字的方法、也有同样的选择器。你可以使用选择器来调用一个对象的方法。
选取器有以下特点：
* 所有同名的方法拥有同样的选取器
* 所有的选取器都是不一样的
(1) SEL和@selector
选择器的类型是 SEL。@selector指示符用来引用选择器，返回类型是SEL。
例如：
SEL responseSEL; responseSEL = @selector(loadDataForTableView:);

可以通过字符串来得到选取器，例如：
responseSEL = NSSelectorFromString(@"loadDataForTableView:");
也可以通过反向转换，得到方法名，例如：
NSString *methodName = NSStringFromSelector(responseSEL);
(2) 方法和选取器
选取器确定的是方法名，而不是方法实现。这是多态性和动态绑定的基础，它使得向不同类对象发送相同的消息成为现实；否则，发送 消息和标准C中调用方法就没有区别，也就不可能支持多态性和动态绑定。
另外，同一个类的同名类方法和实例方法拥有相同的选取器。
(3) 方法返回值和参数类型
消息机制通过选取器找到方法的返回值类型和参数类型，因此，动态绑定（例：向id定义的对象发送消息）需要同名方法的实现拥有相 同返回值类型和相同的参数类型；否则，运行时可能出现找不到对应方法的错误。
有一个例外，虽然同名类方法和实例方法拥有相同的选取器，但是它们可以有不同的参数类型和返回值类型。
<3.动态绑定

66、Objective-C的优缺点

答：objc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
objc缺点:
1).不支援命名空间
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。

67、对于OC,你认为最大的优点和最大的不足是什么？对于不足之处，现在有没有可用的方法绕过这些不足来实现需求。如果可以话，有没有考虑或者实现过重新实现OC的功能，如果有，具体怎么做?

最大的优点是它的运行时特性，不足是没有命名空间，对于命名冲突，可以使用长命名法或特殊前缀解决，如果是引入的第三方库之间的命名冲突，可以使用link命令及flag解决冲突。

68、oc中可修改和不可以修改类型

答：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。

69、我们说的oc是动态运行时语言是什么意思?

答：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?~~~

70、通知和协议的不同之处?

答：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。

71、什么是推送消息?

答：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。

72、关于多态性

答：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。

73、什么是谓词?

答：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];
predicate = [NSPredicate predicateWithFormat:@"customerID == %d",n];
a = [customers filteredArrayUsingPredicate:predicate];
```

74、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：ASIHTTPRequest与NSURLConnection

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

答: 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。

76、谈谈Object-C的内存管理方式及过程?

答: 1).当你使用new,alloc和copy方法创建一个对象时,该对象的保留计数器值为1.当你不再使用该对象时,你要负责向该对象发送一条release或autorelease消息.这样,该对象将在使用寿命结束时被销毁.

2).当你通过任何其他方法获得一个对象时,则假设该对象的保留计数器值为1,而且已经被设置为自动释放,你不需要执行任何操作来确保该对象被清理.如果你打算在一段时间内拥有该对象,则需要保留它并确保在操作完成时释放它.

3).如果你保留了某个对象,你需要(最终)释放或自动释放该对象.必须保持retain方法和release方法的使用次数相等.

77、Object-C有私有方法吗？私有变量呢?

答: objective-c – 类里面的方法只有两种, 静态方法和实例方法. 这似乎就不是完整的面向对象了,按照OO的原则就是一个对象只暴露有用的东西. 如果没有了私有方法的话, 对于一些小范围的代码重用就不那么顺手了. 在类里面声名一个私有方法

```
@interface Controller : NSObject {
NSString *something;
}
+ (void)thisIsAStaticMethod;
– (void)thisIsAnInstanceMethod;
@end
@interface Controller (private)
- (void)thisIsAPrivateMethod;
@end
```

```
@interface Controller : NSObject {
NSString *something;
}
+ (void)thisIsAStaticMethod;
– (void)thisIsAnInstanceMethod;
@end
@interface Controller (private)
- (void)thisIsAPrivateMethod;
@end
```

@private可以用来修饰私有变量

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

78、说说响应链

答：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。

描述响应者链条

当触摸事件发生时,压力转为电信号,iOS系统将产生UIEvent对象,记录事件产生的时间和类型,然后系统将事件加入到一个由UIApplication管理的事件队列中。
UIApplication会从事件队列中取出最前面的事件，并将事件分发下去以便处理，通常，先发送事件给应用程序的主窗口（keyWindow）
主窗口会在视图层次结构中找到一个最合适的视图来处理触摸事件(从父到子,从后到前)，这也是整个事件处理过程的第一步
找到合适的视图控件后，就会调用视图控件的touches方法来作具体的事件处理

79、时间传递 & 响应者链

事件的产生和传递过程：
1.发生触摸事件后，系统会将该事件加入到一个由UIApplication管理的队列事件中
2.UIApplication会从事件队列中取出最前面的事件，并将事件分发下去以便处理，通常会先发送事件给应用程序的主窗口(keyWindow)
3.主窗口会在视图层次结构中找到一个最合适的视图来处理触摸事件
4.找到合适的视图控件后，就会调用视图控件的touches方法来作事件的具体处理：touchesBegin… touchesMoved…touchesEnded等
5.这些touches方法默认的做法是将事件顺着响应者链条向上传递，将事件叫个上一个相应者进行处理
一般事件的传递是从父控件传递到子控件的
如果父控件接受不到触摸事件，那么子控件就不可能接收到触摸事件
UIView不能接收触摸事件的三种情况：
1.不接受用户交互：userInteractionEnabled = NO;
2.隐藏：hidden = YES;
3.透明：alpha = 0.0~0.01
用户的触摸事件首先会由系统截获，进行包装处理等。
然后递归遍历所有的view，进行碰触测试(hitTest)，直到找到可以处理事件的view。
- (UIView *)hitTest:(CGPoint)point withEvent:(UIEvent *)event; // recursively calls -pointInside:withEvent:. point is in the receiver's coordinate system
- (BOOL)pointInside:(CGPoint)point withEvent:(UIEvent *)event; // default returns YES if point is in bounds

大致的过程application –> window –> root view –>……–>lowest view

响应者链
响应者链条其实就是很多响应者对象(继承自UIResponder的对象)一起组合起来的链条称之为响应者链条
一般默认做法是控件将事件顺着响应者链条向上传递，将事件交给上一个响应者进行处理。那么如何判断当前响应者的上一个响应者是谁呢？有以下两个规则：
1.判断当前是否是控制器的View，如果是控制器的View，上一个响应者就是控制器
2.如果不是控制器的View，上一个响应者就是父控件
当有view能够处理触摸事件后，开始响应事件。 系统会调用view的以下方法：
- (void)touchesBegan:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesMoved:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesEnded:(NSSet *)touches withEvent:(UIEvent *)event;
- (void)touchesCancelled:(NSSet *)touches withEvent:(UIEvent *)event;
可以多对象共同响应事件。只需要在以上方法重载中调用super的方法。
大致的过程initial view –> super view –> …..–> view controller –> window –> Application
需要特别注意的一点是，传递链中时没有controller的，因为controller本身不具有大小的概念。但是响应链中是有controller的，因为controller继承自UIResponder。
UIApplication–>UIWindow–>递归找到最合适处理的控件–>控件调用touches方法–>判断是 否实现touches方法–>没有实现默认会将事件传递给上一个响应者–>找到上一个响应者–>找不到方法作废
PS：利用响应者链条我们可以通过调用touches的super 方法，让多个响应者同时响应该事件。

80、frame和bounds有什么不同?

答: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)

81、方法和选择器有何不同?

答：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。

82、OC的垃圾回收机制?

答： 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。

83、什么是延迟加载?

答：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。

84、是否在一个视图控制器中嵌入两个tableview控制器?

答：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。

85、一个tableView是否可以关联两个不同的数据源?你会怎么处理?

答：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?

86、什么时候使用NSMutableArray，什么时候使用NSArray?

答：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。

87、给出委托方法的实例，并且说出UITableVIew的Data Source方法

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