1. iPhone中得线程并不是无节制的，官方给出来得资料显示，iPhone OS下得主线程的堆栈的大小事1M，第二个线程开始就是512K，并且该数值不能通过编译器开关或者线程API来更改，只有主线程有直接修改UI的能力
2. 多线程中NSTHread提供了检测当前线程是否是主线程的方法，但是这种方法需要程序员手动的管理线程，而且在多线程的情况下线程什么时间执行完毕都是未知的，如果管理不好就会造成内存的泄露，所以这种方法不是很提倡的
3. 其实在iOS中统一时间点只有一个进程是在使用CPU 的，只是系统把这个时间片段分割的非常短，造成一种多个进程同时在执行的假象。
4. NSOperation这种方法更加的面向对象，而且可以控制最大的并发数，而且通常我们遇到操作之间的依赖关系的时候都会采用这种方法来管理多线程，因为这种情况下多线程的操作是相比于其他的两种多线程之间的操作来说是非常简单的。
5. GCD是一种纯C语言的多线程操作语言，这种语言最大的好处就是程序员不需要多线程进行操作，只需要知道当前的操作需要在当前线程操作还是需要到后台进行操作对当前的操作分配相应 的线程就行了，但是这种方式遇到依赖的关系的时候虽然说也是能够执行的但是相比于NSOperation来说显得非常的麻烦，所以再遇到依赖关系的线程或者操作的时候不推荐这种方式
6. 串行-同步：显然这是一个线程在执行（这个就是真正意义上的单线程了）

串行-异步：可能会产生多个线程，但是同一时间只会有一个线程在执行

并行-同步：同一时间点有多个相同的线程在执行

并行-异步：同一

时间有多个不同的线程在执行（这个才是真正意义上得多线程执行）

1. Socket描述了一个IP、端口对，包含了通信的双方，即客户端和服务端，DNS的端口是53 HTTP的端口是80
2. 网络七层：物理层、数据链路层、网络层、传输层、会话层、表示层、应用层 其中前三层被称作是媒体层，是网络工程师研究的对象，后面的四层是用户所面向和关心的内容 HTTP 协议对应于应用层 TCP协议对应于传输层 iP协议对应于网络层 HTTP协议是基于TCP链接的 tcp/ip是传输层协议，主要解决数据如何在网络中传输，而HTTP是应用层协议，主要解决如何包装数据 socket本身并不是协议，而是一个调用的接口（API），通过socket我们才能使用tcp/ip协议
3. 建立一个TCP链接需要经过三次握手，HTTP链接最显著的特点是客户端发送的每次请求都需要服务器会送响应，结束请求后，会主动释放链接。从建立链接到关闭链接的过程称为 一次连接 HTTP1.1以后可以在一次连接中处理多个请求，并且多个请求可以重叠进行，不需要等待一个请求结束后再进行下一次请求 HTTP连接是一种短连接，奥保持客户端程序的在线状态，及时客户端不需要请求任何数据也要每隔固定的时间像服务器发送一次保持连接的请求，弱服务器长时间无法接受到请求则认为网络已经断开
4. Socket是支持tcp/ip 协议的网络通信的基本操作单元，包含进行网络通信必须得五种信息：连接使用的协议，本地主机的IP地址，本地进程的协议端口，远地主机的IP地址，远地进程的协议端口。应用层和传输层通过套接字接口来区分来自不同应用程序或者网络连接的通信，实现数据传输的并发服务
5. 通常情况下套接字连接就是TCP连接，双方可以通信知道连接断开，但是在实际网络当中客户端和服务器之间要通过多个节点，比如路由器，网管，防火墙等，大部分防火墙会默认关闭长时间处于非活跃状态的连接而导致连接断开，因此要通过轮询告诉网络，该连接处于活跃状态，如果需要服务端像客户端推送消息的话，就需要建立套接字连接，因为HTTP连接需要客户端的请求，服务端才会给客户端发送数据
6. HTTP协议支持的服务不限于WWW，还可以是其他的服务，因而HTTP协议允许用户在同一的界面下，采用不同的协议访问不同的服务，另外HTTP协议还可恶意用于名字服务器和分布式对象管理 HTTP协议请求的方法通常有GET ,HEAD ,POST等请求方法 HTTP无连接的含义是：限制每次连接值处理一个请求，服务器处理完客户的请求，并受到客户的应答之后就断开连接 HTTP是无状态协议的含义是：对于事物处理没有记忆功能这个特点的缺点是：如果后续处理需要前面的信息，则他必须重新传送，这样可能会导致每次连接传送的数据量增大，但是服务器不需要先前信息的时候他的应答就比较的快 了
7. ;
8. @synthesize managedObjectContext = = \_managedObjectContext 这一行代码的作用是前者是类的真实属性，而后者只是一个加上——的一个实际的变量这样做得原因就是把前者的getter和setter的方法强制性的作用于后者和，用来保护前者变量不会被不恰当的访问
9. Abort（）是用来终止整个程序的也就是程序出现错误的情况后就直接终止整个程序
10. Coredata 不允许跨存储（文件）创建关系，
11. 在使用autolayout的时候一定要关闭autoresizing，因为这两个方式虽然不是同一个概念，但是默认情况下是会相互转换的，所以我们要关闭掉自动转换的功能
12. Autosizing masks在当俯视图改变大小的时候，他会告诉子视图调整大小，但是不会告诉子视图应该调整多少，所以很多的时候如果用storyboard上面的自动尺寸的功能的时候如果要保持父视图改变的时候子视图能跟着改变，并且会自动伸缩自己的尺寸的时候就必须要指定视图在视图改变的时候的视图的大小，不然话这个是不会自动的。
13. 其实在GCD中全局队列只有三个也就是高、中、低三个优先级的队列
14. 如果要获取某一个时间点在某一个历法下得表示，需要用到nsdatecomponents如果想要计算某一个时间点在某一个历法下对应的一个时间段前或者后的时间点，就需要用到nscalendar
15. 使用nsdatecomponents表示时间时，要确保需要使用的参数都已经添加了
16. 其实nscalendar、dateformatter有自动获取本地时区的功能 但是两者的local如果没有经过设置的话都是位空得
17. Nscalendar的firstweekday属性也就是设置一个星期的开始的时间是星期几，默认情况下 苹果官方认为一个星期开始是星期日也就是他得值是1，如果按照我们平常的习惯的话，把这个属性设置成1就行了
18. 如果应用数据很大，界面很多，关系很复杂的时候，如果使用故事版技术那我们在Interface Bulider设计器中得工作就会变得非常的庞大而且复杂，并且整个工程的性能也会受到一定的影响
19. 如果在一个类中建立一个数组属性并使用懒加载的时候那么 对这个数组增加对象的时候一定要使用点语法
20. 按钮圆角类型下 选中状态下左上角会出现一个小视图 不过改成custom就没有了？
21. Tableview的分界线如果不是自定义的话 就是按照 自己选得模式来算出自己选出的位置，所以如果不是自定义的cell的话 那么 一定要把原来的分界线弄掉
22. 如果是自定义的cell的话 那么就必须在返回cell的那个方法中把tableview的contentsize的大小设置成除了状态栏意外所有的地方的大小。（注意：在这个方法中设置后来发现 在开始的时候，也就是视图第一次出现的时候是有用 ，但是如果从其他的视图跳转回来的时候就会失效，最终发现可以在-(scroolviewdidscroll:这个方法中设置就不会出现上述BUG)但是上述情况是在想要改变tableview的contentsize的情况下作的，但是平常的导航条遮拦可以设置 self.edgesForExtendedLayout = UIRectEdgeNone 如果是最下面tabbar遮拦就可以把tableview的frame的高度减小49
23. dequeueReuseableCellWithIdentifier:与dequeueReuseableCellWithIdentifier:forIndexPath:的区别：

前者不必向tableView注册cell的Identifier，但需要判断获取的cell是否为nil；

后者则必须向table注册cell，可省略判断获取的cell是否为空，因为无可复用cell时runtime将使用注册时提供的资源去新建一个cell并返回

自定义cell时，记得将其他内容加到self.contentView 上，而不是直接添加到 cell 本身上

1. layoutSubviews在以下情况下会被调用：

1、init初始化不会触发layoutSubviews

   但是是用initWithFrame 进行初始化时，当rect的值不为CGRectZero时,也会触发

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

在苹果的官方文档中强调:

      You should override this method only if the autoresizing behaviors of the subviews do not offer the behavior you want.

 layoutSubviews, 当我们在某个类的内部调整子视图位置时，需要调用。

反过来的意思就是说：如果你想要在外部设置subviews的位置，就不要重写。

1. Webview 的didstartload 和shouldstartload两个方法在同一个网络请求的时候都不会只是执行一次但是didfinishload 方法会只是执行一次，所以在这两个方法中创建菊花的时候要设置一个值保证只是创建一个菊花 而不会创建两个
2. 基本上所有的视图空间都会由一个tag都可以作为区别两个同种类别的控件
3. 在使用tableview的时候一定要注意reloaddata的时候重新载入的不仅是cell的内容还包括header的内容 所以如果需要header的内容不要随着reloaddata的时候一定不要在viewforheaderinsection 这个方法中穿件header对象，因为reloaddata的时候这个方法也是会重新载入的 所以header会变回原来的样子，所以要把header放在控制器中保存下来。
4. 之所以需要在cell的contentview中添加子视图是因为如果直接在cell中添加的话，那么当需要解决重用问题的时候需要从cell中去掉子视图，那么会连分割线等其他的视图一块去掉，如果加载contentview中就不会出现这种情况。
5. launchImage 是在没有LaunchScreen.xib和ios8以下显示的
6. 同一个开发者帐号只能同时拥有一个没有用过的证书，如果有没有用过的证书那么就不能生成新的证书
7. 图片的名字不能加.9不然 真机的时候会不认识 名字有.9的都会不认识，但是模拟器认识
8. 用UIbutton生成的UIBarbutton ，UIbutton必须设置frame 但是frame中貌似只有宽和高有用

Block写递归

 今天又有同学面试被问到用 blcok 写递归了，写一个留在这里哈，有兴趣的同学面试前查阅一下。  
  
首先来一个 oc 的递归：

- (int)sum:(int)num {

    if (num == 0) {

        return num;

    }

    return num + [self sum:num - 1];

}  
  
写递归算法只需要记住两点即可：  
  
1. 有一个明确的出口  
2. 不满足条件出口时，自己调用自己  
  
按照以上思路用 block 改写一下：

static int (^sumBlock)(int) = ^ (int num) {

    if (num == 0) {

        return num;

    }

    return num + sumBlock(num - 1);

};   
  
注意，要做到自己调用自己，需要能够准确的在内存中找到 `block` 的函数入口，因此需要使用 `static` 修饰符号，其他就没啥了  
  
另外，面试中如果被问到，一定要说下：  
   
1. 每调用一次自己，系统都会开辟一个栈桢记录临时变量和参数  
2. 递归次数过多，会出现栈溢出错误  
3. 移动开发中不建议使用递归算法，现在主线程栈区只有 512K  
  
上面的测试代码调用 NSLog(@"%d", sumBlock(1024 \* 128)); 就会出现栈溢出错误

1. 如果把一个tableviewcontroller的tableview放在一个scollview中会出现tableview的frame 不正常

（1） 通过查找的方式来(这方式适合所有格式的子符串，推荐使用)

   NSString \*newStr =@"abdcdddccdd00大家好哦";

   NSString \*temp = nil;

   for(int i =0; i < [newStr length]; i++)

   {

       temp = [newStr substringWithRange:NSMakeRange(i, 1)];

       NSLog(@"第%d个字是:%@",i,temp);

   }

(2) 通过遍历字符的方式遍历字符串(只适合不包含中文的字符串)

   NSString \*newStr = @"abdcdddccdd00";

   for(int i =0; i < [newStr length]; i++)

   {

      NSLog(@"第%d个字符是:%@",i, [newStr characterAtIndex:i]);

   },其实这个方法中因为characteratindex：这个方法返回的是unichar格式的字符，可以用%C来输出，但是一定要是大写C

1. char   %c

short int  %hi %hx %ho

unsigned short int %hu %hx %ho

int  %i %x %o

unsigned int %u %x %o

long int  %li %lx %lo

unsigned long int  %lu %lx %lo

long long int  %lli %llx &llo

unsigned long long int  %llu %llx %llo

float %f %e %g %a

double  %f %e %g %a

long double %Lf $Le %Lg

id %p

bool %d 0表真，1表假

%@ 对象

%d， %i 整数

%u 无符号整形

%f 浮点/双字

%x, %X 二进制整数

%o 八进制整数

%zu size\_t

%p 指针

%e 浮点/ 双字 (科学计算)

%g 浮点/ 双字

%s C字符串

%.\*s Pascal字符串

%c 字符

%C unichar

%lld 64位长整数（long long）

%llu 无符号64位长整数

%Lf  64位双数

1. 如果需要对视图控件设置圆角且设置的不是背景颜色二十背景图片，此时如果背景图片也是圆角的情况下，那么很有可能原来的圆角就会出现漏色的现象，这是需要对空间的layer层设置圆角，一般情况下如果
2. Failed to lookup the process ID of cn.sinsoft.www after successful launch. Perhaps it crashed after launch.这种情况是在在模拟器上运行程序时候遇到的问题 ，这个时候不需要进行其他的操作就 直接把模拟器关了然后重新运行程序就行了
3. 有米所提供的第三方的广告SDK存在安全隐患，不能用，iOS8规定APP不能读取设备序列号
4. AFN 2.6中如果要访问 https 的网站，加一句 manager.securityPolicy.validatesDomainName = false 就行了
5. . scrollerview 最好不能作为导航控制器下得控制器下第一个view的第一个子视图，此时会有64pt 的不稳定性，此时可以对控制器self.automaticallyAdjustsScrollViewInsets = NO 设置，不然会有不同的差距。
6. NSTimer 用invalidate 停止，但是停止之后必须重新初始化nstimer
7. timer、preformselector都是runloop模式下的东西
8. runloop和GCD是互相利用关系 但不是丛书关系、uiEVENT是从属于runloop的，
9. 如果scrollView滑动的时候 加载在default模式下的timmer会停止的 用scheum方法创建的是加载在default下的 如果不想因为滑动就把model 放在common模式下。Model的模式的切换才能让UI运行顺畅
10. 50 GCD如果放在mainqueue中block是放在runloop的 但是其他的并不是，GCD的延迟调用和preformselector不一样 后者是在runloop中 前者不一样
11. 内存布局以一个 objc\_class 指针为开始的所有东东都可以当做一个 object 来对待!
12. 任意一个对象都是个结构类型的指针
13. 每个对象都会有一个用objc\_object类型指针指向的super\_class（这个类如果是根类那么这个属性为null） 类对象的ISA 指针指向 结构类型metaclass,metaclass的ISA指针指向跟metaclass如果metaclass是根metaclass那么其ISA指针指向自己，metaclass的super\_class指向自己的父类metaclass，根metaclass的super\_class则指向该meta\_class对应的类
14. sizeof计算数据（包括数组、变量、类型、结构体等）所占内存空间，用字节数表示。
15. (类型说明符\*)calloc(n,size)此方法的作用是在内存的动态存储区中分配n个size大小的连续空间
16. (类型说明符\*) malloc (size) 功能：在内存的动态存储区中分配一块长度为"size" 字节的连续区域。函数的返回值为该区域的首地址。 “类型说明符”表示把该区域用于何种数据类型。(类型说明符\*)表示把返回值强制转换为该类型指针。
17. calloc函数与malloc 函数的区别仅在于一次可以分配n块区域
18. strlen()计算字符数组的字符数，以"\0"为结束判断，不计算为'\0'的数组元素。

char \* strcpy(char \* dest, const char \* src) // 实现src到dest的复制

1. iOS 设计模式：观察者模式、代理模式、依赖注入、工厂模式、kvo模式，单例模式、策略模式
2. iOS中一般用const来修饰全局常量：在一个项目中任何一个文件中定义一个全局常量 ，然后再文件中其他的位置都能够访问，但是用static也可以用来修饰常量 但是此时的常量是 局部常量 在一个文件中定义一个局部常量 只能在这个文件的内部使用 ，但是在 项目的其他的任何地方使用都会报错 （找不到这个 常量）
3. 以下的代码证明const的右边的总是不能被修改

1.const NSString \*HSCoder = @"汉斯哈哈哈";

"\*HSCoder"不能被修改， "HSCoder"能被修改

2.NSString const \*HSCoder = @"汉斯哈哈哈";

"\*HSCoder"不能被修改， "HSCoder"能被修改

3.NSString \* const HSCoder = @"汉斯哈哈哈";

"HSCoder"不能被修改，"\*HSCoder"能被修改

1. 如果我们定义一个常量又不想这个常量被修改那么就定义：类名 \*const 对象名
2. iOS9 新特性nullable表示能为空 ，nonnull 表示不能为空 而且两个关键字只能修饰对象 可以用来修饰属性、参数、返回值
3. null\_resettable:setter可以为空 但是get方法不能为空 ，而且如果用这个关键词修饰属性就必须要重写这个属性的getter去处理get为空的情况
4. *\_null\_unsepcified不确定是否为空，但是不能再属性括号中修饰，只能在属性名前修饰*
5. *泛型对属性进行规定的时候可以防止赋值错误，而且取值的时候可以直接使用该泛型的基本方法，泛型一般适用于集合类型 ，也可以使用自定义泛型 在创建类的时候就直接自定义一个泛型，然后再创建此对象的时候为自定义泛型设置确定值，这样在赋值的时候就会有相应的泛型的提示*
6. *泛型中想要将子类赋值给父类属性 可以使用\_\_convariant来为泛型做协变，也就是泛型数据的强转；\_\_contravariant是逆变可使用父类赋值给子类属性*
7. *\_\_kindof一般用于方法的返回值中的限制，表示可以返回它修饰的类本身的对象或者子类对象*
8. *默认每一个方法都有两个隐式参数：self：方法调用者 \_cmd：调用方法的编号*
9. *Void \* 就是id*
10. *当有字符串常量的时候，苹果官方推荐使用const*
11. *Const只有在conmmand + b才会编译 而宏是预编译 的 也就是说不用 按键操作就已经编译了，前者会有错误检查 ，后者不会有错误检查，宏可以定义代码，但是const不能， 宏编译时间过长，但是const则只需要编译一次就行了*
12. *const作用是用来修饰右边的变量比如：指针变量、基本变量、对象变量等，而且用const修饰的变量无法修改 const int a 和 int const a没有区别*
13. *Static用于修饰局部变量的时候只要程序运行这个变量就存在，而且只会分配一次内存；static修饰的代码 只会运行一次 以后就不会执行了，用来修饰全局变量表示这个变量只能在当前文件中使用*
14. *Extrn修饰的变量不能初始化，他仅仅是表示声明，但是不能定义，也就是说不能使用这个变量，这个一般用于声明全局变量 ，用于在其他文件中申明全局变量后 在想要用的文件作用用extrn中告诉他有这个变量，找到这个变量*
15. *Super不是一个指针 他只是一个编译修饰符，不能打印，一般用这个表示用当前调用对象来调用父类方法*
16. Block当做属性的时候如果在ARC 环境的时候使用strong 在非ARC环境中用copy
17. 可以将block当做返回值使用 这样使用的话 可以使用点语法来调用方法 ，其实也就是相当于写一个getter
18. 链式编程思想 其实就是 多个点语法
19. 设置约束之前一定要把 view先添加到父控件上之后再弄
20. KVO的本质就是对象有没有调用改值的 set方法
21. @property 这个关键词 会检测类中是否有加\_的属性如果有同名的下划线属性 就只会生成getter并不会生成setter方法 并且用这个词生成的属性一般是私有属性 对象名->属性名
22. RACsubject默认如果一个信号发送完毕就会自动取消订阅 ，而且只要订阅者存在那么就不会取消 或者调用[disposeable dispose]
23. 只有在主运行循环 才能监听网络状态
24. 有时服务器传回来的string是nstaggedpointerstring类型可以对这个类型进行mutableCopy 来获取string类型