打造最受名企欢迎的iOS程序员（下）

本文编辑：竹_子

关注作者简书：小猿员

给广大iOS开发者的进阶一些方向性的帮助。

（iOS技术交流群624212887，欢迎大家进群多多交流）

51.在Obj-c中有没有私有方法？私有变量？一般采用什么方法实现？

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

```
@interfaceController : NSObject { NSString *something; }
+(void)thisIsAStaticMethod;
-(void)thisIsAnInstanceMethod;
@end
@interfaceController (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

52. obj-c的优缺点

答案：

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
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。

a) int a; // Aninteger
b) int *a; // A pointer to aninteger
c) int **a; // A pointer to apointer to an integer
d) int a[10]; // An array of10 integers
e) int *a[10]; // An array of10 pointers to integers
f) int (*a)[10]; // A pointerto an array of 10 integers
g) int (*a)(int); // A pointerto a function a that takes an integer argument and returns aninteger
h) int (*a[10])(int); // Anarray of 10 pointers to functions that take an integer argument andreturn an integer

53.HTTP协议中，POST和GET的区别是什么？
答案：1.GET 方法
GET 方法提交数据不安全，数据置于请求行，客户端地址栏可见;
GET 方法提交的数据大小有限
GET 方法不可以设置书签

54.POST 方法
POST 方法提交数据安全，数据置于消息主体内，客户端不可见
POST 方法提交的数据大小没有限制
POST 方法可以设置书签

55. 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。

56. 控件主要响应3种事件：（ 基于触摸的事件 ）、（ 基于值的事件 ）和（ 基于编辑的事件 ）。

57. xib文件的构成分为哪3个图标？都具有什么功能。（10分）
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。

58. UIView与CLayer有什么区别（10分）？
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。

59. UIView 有个重要属性 layer ，可以返回它的
主 CALayer 实例。

60. 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。

61. 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。

62. 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。

63. 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。

64. 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。

65. 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。

66. 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。

67. Quatrz 2D的绘图功能的三个核心概念是什么并简述其作用（10分）。
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。

68. iPhone OS主要提供了几种播放音频的方法（10分）？
答： SystemSound Services
AVAudioPlayer 类
Audio Queue Services
OpenAL

69. 使用AVAudioPlayer类调用哪个框架、使用步骤（10分）？
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。

70. CFSocket使用有哪几个步骤（10分）。
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；

71. Core Foundation中提供了哪几种操作Socket的方法（10分）？
答： CFNetwork 、 CFSocket 和 BSD Socket 。

72.线程与进程的区别和联系?
答案 ： 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。 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。

73. 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； core data 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 core data 的时候，你不用安装额外的数据库系统，因为 core data 使用内置的 sqlite 数据库。 core data 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反的改变，例如用户执行撤销命令。当 core data 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 core data 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。

74. 获取项目根路径，并在其下创建一个名称为userData 的目录。（10分）。

```
//  获取根路径
NSArray
*paths=NSSearchPathForDirectoriesInDomains(NSDocumentDirect
ory,NSUserDomainMask,YES);
NSString *documentsDirectory = [paths objectAtIndex:];
//  创建文件系统管理器
NSFileManager *fileManager = [[NSFileManageralloc] init];
//  判断userData 目录是否存在
if(![fileManagerfileExistsAtPath:[NSStringstringWithFormat:
@"%@/userData",documentsDirectory]]) {
//  不存在， 创建一个userData目录
[fileManagercreateDirectoryAtPath:[NSStringstringWithFormat:
@"%@/userData",documentsDirectory]withIntermediateDirectori
es:falseattributes:nilerror:nil];
   }
```

75.列举几种进程的同步机制，并比较其优缺点。
答案：原子操作 信号量机制　自旋锁　管程，会合，分布式系统
1.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
2.进程死锁的原因
答案：资源竞争及进程推进顺序非法
3.死锁的4个必要条件
答案：互斥、请求保持、不可剥夺、环路
4.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

76.进程间通信的方式有_____

（1）管道（Pipe）：管道可用于具有亲缘关系进程间的通信，允许一个进程和另一个与它有共同祖先的进程之间进行通信。

（2）命名管道（named pipe）：命名管道克服了管道没有名字的限制，因此，除具有管道所具有的功能外，它还允许无亲缘关系进程间的通信。命名管道在文件系统中有对应的文件名。命名管道通过命令mkfifo或系统调用mkfifo来创建。

（3）信号（Signal）：信号是比较复杂的通信方式，用于通知接受进程有某种事件发生，除了用于进程间通信外，进程还可以发送信号给进程本身；linux除了支持Unix早期信号语义函数sigal外，还支持语义符合Posix.1标准的信号函数sigaction（实际上，该函数是基于BSD的，BSD为了实现可靠信号机制，又能够统一对外接口，用sigaction函数重新实现了signal函数）。

（4）消息（Message）队列：消息队列是消息的链接表，包括Posix消息队列system V消息队列。有足够权限的进程可以向队列中添加消息，被赋予读权限的进程则可以读走队列中的消息。消息队列克服了信号承载信息量少，管道只能承载无格式字节流以及缓冲区大小受限等缺

（5）共享内存：使得多个进程可以访问同一块内存空间，是最快的可用IPC形式。是针对其他通信机制运行效率较低而设计的。往往与其它通信机制，如信号量结合使用，来达到进程间的同步及互斥。

（6）内存映射（mapped memory）：内存映射允许任何多个进程间通信，每一个使用该机制的进程通过把一个共享的文件映射到自己的进程地址空间来实现它。

（7）信号量（semaphore）：主要作为进程间以及同一进程不同线程之间的同步手段。

（8）套接口（Socket）：更为一般的进程间通信机制，可用于不同机器之间的进程间通信。起初是由Unix系统的BSD分支开发出来的，但现在一般可以移植到其它类Unix系统上：Linux和System V的变种都支持套接字。

77.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。

TCP全称是Transmission Control Protocol，中文名为传输控制协议，它可以提供可靠的、面向连接的网络数据传递服务。传输控制协议主要包含下列任务和功能：

* 确保IP数据报的成功传递。
* 对程序发送的大块数据进行分段和重组。
* 确保正确排序及按顺序传递分段的数据。
* 通过计算校验和，进行传输数据的完整性检查。

78、TCP和UDP的区别

TCP提供的是面向连接的、可靠的数据流传输，而UDP提供的是非面向连接的、不可靠的数据流传输。

简单的说，TCP注重数据安全，而UDP数据传输快点，但安全性一般

79.你了解svn,cvs等版本控制工具么？
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。

80. 为什么很多内置类如UITableViewController的delegate属性都是assign而不是retain的？
答：

会引起循环引用

所有的引用计数系统，都存在循环应用的问题。例如下面的引用关系：

* 对象a创建并引用到了对象b.
* 对象b创建并引用到了对象c.
* 对象c创建并引用到了对象b.

这时候b和c的引用计数分别是2和1。

当a不再使用b，调用release释放对b的所有权，因为c还引用了b，所以b的引用计数为1，b不会被释放。

b不释放，c的引用计数就是1，c也不会被释放。从此，b和c永远留在内存中。

这种情况，必须打断循环引用，通过其他规则来维护引用关系。我们常见的delegate往往是assign方式的属性而不是retain方式 的属性，
赋值不会增加引用计数，就是为了防止delegation两端产生不必要的循环引用。

如果一个UITableViewController 对象a通过retain获取了UITableView对象b的所有权，这个UITableView对象b的delegate又是a，

如果这个delegate是retain方式的，那基本上就没有机会释放这两个对象了。自己在设计使用delegate模式时，也要注意这点。

81.通信底层原理

答：OSI七层模型

7 应用层： ftp,smtp,http,telnet,tftp（通过各种协议，最终还是包装成TCP数据包，发送到网络中！）

6 表现层：

5 会话层：

4 传输层： tcp udp

3 网络层： ip,ICMP,IGRP,EIGRP,OSPF,ARP

2 数据链路层： STP,VT

1 物理层：

82、objective-c 是所有对象间的交互是如何实现的？

在对象间交互中每个对象承担的角色不同，但总的来说无非就是”数据的发送者”或”数据的接收者”两种角色，我们可以通过代理去进行通信，或者通过观察者消息模式，blocks，appdelegagte

83、TCP/IP 建立连接的过程？

在TCP/IP 协议中，TCP协议提供可靠的连接服务，采用三次握手建立连接；

第一次握手：建立连接时，客户端发送连接请求到服务器，并进入SYN_SEND状态，等待服务器确认；

第二次握手：服务器收到客户端连接请求，向客户端发送允许连接应答，此时服务器进入SYN_RECV状态；

第三次握手：客户端收到服务器的允许连接应答，向服务器发送确认，客户端和服务器进入通信状态，完成三次握手。

（所谓的三次握手，就是要有三次连接信息的发送、接收过程。TCP连的建立需要进行三次连接信息的发送、接收。）

84、如何引用一个已经定义过的全局变量？

extern

可以用引用头文件的方式，也可以用extern 关键字，如果用引用头文件的方式来引用某个在头文件中的全局变量，假定你那个变量写错了，那么编译期间会报错，如果用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。

85.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.

86. (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.

87. 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——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();});
```

__block和__weak修饰符的区别其实是挺明显的：
1.__block不管是ARC还是MRC模式下都可以使用，可以修饰对象，还可以修饰基本数据类型。
2.__weak只能在ARC模式下使用，也只能修饰对象（NSString），不能修饰基本数据类型（int）。
3.__block对象可以在block中被重新赋值，__weak不可以。

tableView 滑动卡的问题主要是因为：从缓存中或者是从本地读取图片给UIImage的时候耗费的时间。需要把下面的两句话放到子线程里面：

```
1 NSData  *imgData  =  [NSData  dataWithContentsOfURL:[NSUR
     L  URLWithString:app.icon]];  //得到图像数据
2 UIImage  *image  =  [UIImage  imageWithData:imgData];
```

把UIImage赋值给图片的时候在主线程。
子线程不能更新UI 所有的UI跟新都是主线程执行了。手指滑动屏幕了。或者屏幕的某个方法执行了。

子线程里面加入NSTimer 的时候需要 手动添加NSRunloop 否则不能循环。

单利里面添加 NSMutableArray 的时候，防止多个地方对它同时便利和修改的话，需要加原子属性。并且用strong，，，并且写一个遍历和修改的方法。加上锁。 Lock UnLock

```
__weak ViewController*  weakSelf = self;
```

GCD里面用 __weak 防止内存释放不了，循环引用。

88.id、nil代表什么？
id和void *并非完全一样。在上面的代码中，id是指向struct objc_object的一个指针，这个意思基本上是说，id是一个指向任何一个继承了Object（或者NSObject）类的对象。需要注意的是id是一个指针，所以你在使用id的时候不需要加星号。比如id foo=nil定义

了一个nil指针，这个指针指向NSObject的一个任意子类。而id *foo=nil则定义了一个指针，这个指针指向另一个指针，被指向的这个指针指向NSObject的一个子类。
nil和C语言的NULL相同，在objc/objc.h中定义。nil表示一个Objctive-C对象，这个对象的指针指向空（没有东西就是空）。
首字母大写的Nil和nil有一点不一样，Nil定义一个指向空的类（是Class，而不是对象）。
SEL是“selector”的一个类型，表示一个方法的名字
Method（我们常说的方法）表示一种类型，这种类型与selector和实现(implementation)相关

IMP定义为 id (*IMP)(id, SEL, …)。这样说来， IMP是一个指向函数的指针，这个被指向的函数包括id(“self”指针)，调用的SEL（方法名），再加上一些其他参数.说白了IMP就是实现方法。

冒泡排序

```
main()
{
int i,j,temp;
int a[10];
for(i=0;i<10;i++)
scanf ("%d,",&a[i]);
for(j=0;j<=9;j++)
{ for (i=0;i<10-j;i++)
if (a[i]>a[i+1])
{ temp=a[i];
a[i]=a[i+1];
a[i+1]=temp;}
}
for(i=1;i<11;i++)
printf("%5d,",a[i] );
printf("\n");
}
```

89.为什么很多内置类，如UITableView的delegate属性都是assign而不是retain的？
如果是retain会引起循环引用
一个对象没必要管理自己delegate的生命周期，或者说没必要拥有该对象，所以我们只要知道它的指针就可以了，用指针找到对象去调用方法，也就是委托实现的感觉。

或者我们换个角度，从内存管理方面也可以解释这个问题。delegate的生命周期不需要让该对象去控制，如果该对象对其使用retain很可能导致delegate所指向的对象无法正确的释放。

90.视图控制器的loadView方法是什么时候调用的？
视图控制器的view被访问到，并且为nil时，调用loadView方法创建视图

1.什么情况使用 weak 关键字，相比 assign 有什么不同？
什么情况使用 weak 关键字？
1)在ARC中,在有可能出现循环引用的时候,往往要通过让其中一端使用weak来解决,比如:delegate代理属性
2）自身已经对它进行一次强引用,没有必要再强引用一次,此时也会使用weak,自定义IBOutlet控件属性一般也使用weak；当然，也可以使用strong。
不同点：
1）weak 此特质表明该属性定义了一种“非拥有关系”（nonowning relationship)。为这种属性设置新值时，设置方法既不保留新值，也不释放旧值。此特质同assign类似， 然而在属性所指的对象遭到摧毁时，属性值也会清空(nil out)。 而 assign 的“设置方法”只会执行针对“纯量类型”(scalar type,例如 CGFloat 或 NSlnteger 等)的简单赋值操作。
2）assigin 可以用非OC对象,而weak必须用于OC对象

91.这个写法会出什么问题： @property (strong) NSMutableArray *array;

使用了atomic属性会严重影响性能。
该属性使用了同步锁，会在创建时生成一些额外的代码用于帮助编写多线程程序，这会带来性能问题，通过声明nonatomic可以节省这些虽然很小但是不必要额外开销。
在默认情况下，由编译器所合成的方法会通过锁定机制确保其原子性(atomicity)。如果属性具备nonatomic特质，则不使用同步锁。请注意，尽管没有名为“atomic”的特质(如果某属性不具备nonatomic特质，那它就是“原子的”(atomic))。
在iOS开发中，你会发现，几乎所有属性都声明为nonatomic。
一般情况下并不要求属性必须是“原子的”，因为这并不能保证“线程安全”（ thread safety)，若要实现“线程安全”的操作，还需采用更为深层的锁定机制才行。例如，一个线程在连续多次读取某属性值的过程中有别的线程在同时改写该值，那么即便将属性声明为atomic，也还是会读到不同的属性值。
因此，开发iOS程序时一般都会使用nonatomic属性。但是在开发Mac OS X程序时，使用 atomic属性通常都不会有性能瓶颈。

92.AFNetworking或SDWebImage 里面给 UIImageView 加载图片的逻辑是什么样的?
SDWebImage 中为 UIView 提供了一个分类叫做 WebCache, 这个分类中有一个最常用的接口,
sd_setImageWithURL:placeholderImage: , 这个分类同时提供了很多类似的方法, 这些方法最终会调用一个同时具有 option progressBlock completionBlock 的方法, 而在这个类最终被调用的方法首先会检查是否传入了 placeholderImage 以及对应的参数, 并设置 placeholderImage .
然后会获取 SDWebImageManager 中的单例调用一个 downloadImageWithURL:... 的方法来获取图片, 而这个 manager 获取图片的过程有大体上分为两部分, 它首先会在 SDWebImageCache 中寻找图片是否有对应的缓存, 它会以 url 作为数据的索引先在内存中寻找是否有对应的缓存, 如果缓存未命中就会在磁盘中利用 MD5 处理过的 key 来继续查询对应的数据, 如果找到了, 就会把磁盘中的缓存备份到内存中.

然而，假设我们在内存和磁盘缓存中都没有命中，那么 manager 就会调用它持有的一个 SDWebImageDownloader 对象的方法 downloadImageWithURL:... 来下载图片，这个方法会在执行的过程中调用另一个方法 addProgressCallback:andCompletedBlock:fotURL:createCallback: 来存储下载过程中和下载完成的回调，当回调块是第一次添加的时候，方法会实例化一个 NSMutableURLRequest 和 SDWebImageDownloaderOperation ，并将后者加入 downloader 持有的下载队列开始图片的异步下载.
而在图片下载完成之后，就会在主线程设置 image，完成整个图像的异步下载和配置.

93.GCD 里面有哪几种 Queue? 背后的线程模型是什么样的?
GCD 中 Queue 的种类还要看我们怎么进行分类，如果根据同一时间内处理的操作数分类的话，GCD 中的 Queue 分为两类

Serial Dispatch Queue
Concurrent Dispatch Queue
一类是串行派发队列，它只使用一个线程，会等待当前执行的操作结束后才会执行下一个操作，它按照追加的顺序进行处理. 另一类是并行派发队列，它同时使用多个线程，如果当前的线程数足够，那么就不会等待正在执行的操作，使用多个线程同时执行多个处理.

另外的一种分类方式如下:

Main Dispatch Queue
Global Dispatch Queue
Custom Dispatch Queue
主线程只有一个，它是一个串行的进程. 所有追加到 Main Dispatch Queue 中的处理都会在 RunLoop 在执行. Global Dispatch Queue 是所有应用程序都能使用的并行派发队列，它有 4 个执行优先级 High, Default, Low, Background. 当然我们也可以使用 dispatch_queue_create 创建派发队列.

94.什么是iOS中的沙盒机制。
IOS中的沙盒机制（SandBox）是一种安全体系，它规定了应用程序只能在为该应用创建的文件夹内读取文件，不可以访问其他地方的内容。
1.每个应用程序都在自己的沙盒内
2.不能随意跨越自己的沙盒去访问别的应用程序沙盒的内容
3.应用程序向外请求或接收数据都需要经过权限认证
默认情况下，每个沙盒含有3个文件夹：Documents, Library 和 tmp。
因为应用的沙盒机制，应用只能在几个目录下读写文件
Documents：苹果建议将程序中建立的或在程序中浏览到的文件数据保存在该目录下，iTunes备份和恢复的时候会包括此目录
Library：存储程序的默认设置或其它状态信息；
Library/Caches：存放缓存文件，iTunes不会备份此目录，此目录下文件不会在应用退出删除
tmp：提供一个即时创建临时文件的地方

95.nil, Nil, NSNULL, NULL区别
nil是指向obj－c中对象的空指针，是一个对象，在o－c中ni对象调用方法不会引起crash。
Nil是指向obj－c中的类的空指针，表示的是一个空类。
NULL是指向任何类型的空指针（如c／c++中的空指针），在objective－c中是一个数值。

NSNULL用于集合操作，在集合对象中，表示一个空值的集合对象。

96.iOS中处理音频和视频使用哪些框架？
AVFoundation（基于Core Audio、Core Video、Core Media等框架）、MediaPlayer、UIKit

97.如何监听View的触摸事件,事件是如何传递的、视图的响应者链是什么？
(1)覆写View类的touchBegin、touchMove、touchEnd系列方法监听视图的触摸。

(2)事件传递:当触摸一个视图时,首先系统会捕捉此事件,并为此事件创建一个UIEvent对 象,将此对象加入当前应用程序的事件队列中,然后由UIApplication对象从队列中,一个一个取出来进行分发,首首先分发给UIWindow对象,然后由UIWindow对象分发给触摸的视图对象,也就是第一响应者对象。!
(3)响应者链: 事件被交由第一响应者对象处理,如果第一响应者不处理,事件被沿着响应 者链向上传递,交给下一个响应者(next responder)。一般来说,第一响应者是个视图对 象或者其子类对象,当其被触摸后事件被交由它处理,如果它不处理,事件就会被传递给它 的视图控制器对象(如果存在),然后是它的父视图(superview)对象(如果存在),以 此类推,直到顶层视图。接下来会沿着顶层视图(top view)到窗口口(UIWindow对象)再 到程序(UIApplication对象)。如果整个过程都没有响应这个事件,该事件就被丢弃。一般 情况下,在响应者链中只要由对象处理事件,事件就停止传递。但有时候可以在视图的响应 方方法中根据一些条件判断来决定是否需要继续传递事件。

98.xml数据的解析方式,各有什么不同?
(1)xml数据解析有两种解析方方式:DOM解析与SAX解析!
(2)DOM解析必须先完成DOM树的构造,在处理规模较大的XML文档时就很耗内存,占用资 源较多!
(3)与DOM不同的是,它是用用事件驱动模型,解析XML文文档时每遇到一个开始或者结束标 签、 或者属性、或者一条指令时,程序就产生生一个事件来进行行相应的处理,因此,SAX相对于 DOM来说更适合操作大的XML文档!

99.设备状态栏(Device Status Bar)是什么?高度如何?是否透明?在手机通话或者导航状态下，它是如何显示的?
高度为20px，iOS7开始为透明状态，整合在导航栏（高度64px）中
通话或导航状态下，高度变高为40px

100.Core Graphics 和Quartz 2D的区别?

quartz是一个通用的术语，用于描述在IOS和MAC OS X ZHONG 整个媒体层用到的多种技术 包括图形、动画、音频、适配。
Quart 2D 是一组二位绘图和渲染API，Core Graphic会使用到这组API
Quartz Core 专指Core Animation用到的动画相关的库、API和类。

101.如何为APP添加启动页？
使用LaunchScreen.xib或者Images.xcassets中添加LaunchImage

102.UIView的ContentMode是如何实现的？
contentstretchmode的机理,当view的bounds改变时,ios的绘图系统只是简单的拉伸和重新排列缓存的layer图像.

103.layer的层级结构是什么？
layer的层级和view相似,一个layer有多个sublayer,一个sublayer只有一个最近的superlayer.

有一些方法控制layer的层级顺序,同view相似:
addsublayer;
insertsublayer: atindex(below/above);
replacesublayer:with;
removefromsuperlayer.
layer同时也有zposotion的属性,拥有低的zposition的layer比高的先绘制.当有时使用sublayer不便控制层级时,使用zpositon是个很好的选择.

104.如何确定layer的位置？
控制layer的位置使用两个property.
position 一个superlayer坐标系下的坐标;
anchorpoint 一个本身坐标系下的坐标;

105.transform中，可以使用kvc执行动画的属性有哪些？

rotation.x, rotation.y, rotation.z, rotation (same as rotation.z), scale.x, scale.y, scale.z, translation.x, translation.y, translation.z, and translation

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

107.请简述self.name= nil的机制，以及与[namerelease]的区别?
self.name =nil; //使用nil参数调用setName:方法
[name release] 生成的访问器将自动释放以前的name对象

108.使用sql语句查询出省名以湖开头，邮编为436001所在的市区？（5分）（表名及字段名自定义）
select＊fromcitys where postcode＝436001 and province＝’湖%’;

109.写一” 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))
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);
结果是:
((*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。

110.const char *p; charconst*p; char*const p; const char* const p;四个修饰指针有什么区别

答： （1）定义了一个指向不可变的字符串的字符指针

（2）和（1）一样

（3）定义了一个指向字符串的指针，该指针值不可改变，即不可改变指向

（4）定义了一个指向不可变的字符串的字符指针，且该指针也不可改变指向