iOS 部分
理论：
1.描述什么是MVC
【MVC简介】
是一种架构模式，它是苹果非常热衷的一种架构模式
M： model 模型 保存所有应用程序里要使用的数据，比如一款太空大战游戏，模型要负责保存飞船的大小、飞行速度、位置信息、装载了多少只枪等等这些信息。并且要处理数据之间的逻辑 比如飞船要打中敌机多少次能把敌机击落
模型只是负责记录数据，跟数据的显示是没关系的，数据的显示是控制器跟视图的任务

C： controller 控制器 负责控制视图如何去显示模型里要显示的数据 它要负责把模型里的数据传输给视图（控制器是通过视图控制器的【生命周期】来控制视图变化的）

V：view 视图 视图就是视图控制器的小跟班，它的任务就是负责显示视图，完全听命于视图控制器，视图控制器让视图做什么视图就做什么

2.列举常用网络协议的端口号，例如HTTP默认的是80

FTP 文本传输协议 20或21
TELNET 远程登录协议 23
POP3 发邮件协议 110

3、frame 与 bounds 的区别？Bounds 的大小改变 frame 改变吗
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）
会发生改变

4、UIViewController 的生命周期方法调用顺序
答：- (void)viewDidLoad;

- (void)viewDidUnload;

- (void)viewWillAppear:(BOOL)animated;

- (void)viewDidAppear:(BOOL)animated;

- (void)viewWillDisappear:(BOOL)animated;

- (void)viewDidDisappear:(BOOL)animated;
5、UITableView 的执行流程是怎么样的

答："numberOfSectionsInTableView:"返回 TableView 的 section 数目
"tableView:titleForHeaderInSection:"section1 是否有表头标题栏
"tableView:numberOfRowsInSection:"设置 section1 中行数
"tableView:heightForRowAtIndexPath:"设置 section1 中 row1 行的高度……row2 行的高度……逐行设置，直至当前 section1 属性设置完毕
"tableView:titleForHeaderInSection:"section2 是否有表头栏，之后同 4-5，设置 section2 的属性.同理，设置完毕所有的 section 的相关属性
"tableView:cellForRowAtIndexPath:"接下来设置的是每个 section 中每 row 添加的数据
这样整个 TableView 就设置完毕了.
6、如何设计一个可变高度（根据内容自适应高度）的 UITableViewCell
答：1) 创建并添加一个 UILabel 作为单元格 cell 的子视图；
2) 在 UITableView 的委托方法:
(CGFloat)tableView:(UITableView*)tableViewheightForRowAtIndexPath:(NSIndexPath *) indexPath 中计算高度
3) 在 UITableView 的委托方法:
(UITableViewCell*)tableView:(UITableView*)tableViewcellForRowAtIndexPath: (NSIndexPath *) indexPath 中计算 UILabel 的框大小。
7、UIView 的圆角属性设置方法
答：利用 setCornerRadius:
8. UIVIiewController 在现实过程中，各个方法的调用顺序
init->viewDidLoad->viewDidAppear->viewDidUnload
9.对于语句 NSString *obj = [[NSData alloc]init],obj 在编译时和运行时分别是什么类型的对象？
答：编译时为 NSString 类型，运行时为 NSData 类型。

10.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？
答:a、线程创建有三种方法：使用 NSThread 创建
(detachNewThreadSelector:toTarget:withObject:)、使用 GCD 的 dispatch、使用子类化的 NSOperation,然后将其加入 NSOperationQueue;
b、在主线程执行代码，方法是 performSelectorOnMainThread，
c、如果想延时执行代码可以用 performSelector:withObject:afterDelay:

11.描述一下 iOS SDKSDK 中如何实现 MVC 的开发模式？
答:iOS 开发中使用了很好的分层设计，数据都可以放在自定义类型、NSArray 及其子类型、NSDictionary 及其子类型中，视图的显示都用 UIView 及其子类来实现，控制器在 UIViewController 的子类中实现，在控制器的 ViewDidLoad、ViewWillAppear、ViewDidAppear、ViewDidDisappear、ViewWillDisappear 等方法中实现数据和视图的交互。

12、MVC 设计模式是如何体现在 iOS Appkai'fa 开发中的？三者之间有哪些常见消息传递方式？

答：iOS 开发中使用了很好的分层设计，数据都可以放在自定义类型、NSArray 及其子类型、NSDictionary 及其子类型中，视图的显示都用 UIView 及其子类来实现，控制器在 UIViewController 的子类中实现，在控制器的 ViewDidLoad、ViewWillAppear、ViewDidAppear、ViewDidDisappear、ViewWillDisappear 等方法中实现数据和视图的交互。三者之间常见消息传递方式有：代理、通知中心、kvc/kvo 等。

13. 关于自定义 Cell 中，图片下载用到的方法?
如果有添加了第三方库 SDWebImage/AFNetWorking，可以使用 UIImageView 的类别方法 setImageWithURL:直接异步加载，如果没有的话可以使用 NSURLConnection 发起 request 请求，或者使用 ASI 将下载请求添加到下载队列中，将图片下载之后，在回调方法里，回调主线程，设置图片。

14. UITableViewCell 怎样使用更流畅?
首先 cell 的复用机制节约了系统资源；其次应当注意有些复杂的大数据或网络数据应采用异步加载的方式进行加载，以免 cell 刷出时发生卡顿。

15. 在 UIWebView 上点击回复，如何使用 UITextField 进行回复?
UIWebView 有一个 stringByEvaluatingJavaScriptFromString 方法可以将 javascript 嵌入页面中，通过这个方法我们可以在 iOS 中与 UIWebView 中的网页元素交互。通过 JAVAScript 获取点击时间，弹出 UITextField，输入字符串后，可以通过 post 请求发送回复。

16. 关于图文混排是如何排版的?
图文混排应当以图片为起点，首先明确图片的位置，文字可以选择位于图片的下方，或者是环绕效果。环绕效果可以采用 2 个以上 label 或 textView 来实现；也可以通过 AttributeString 的属性设置，或者 CoreText 重绘 UIVIew，添加文字的方式，修改文字间的间隔，达到让出图片的效果。如果图文显示在高度可变的视图中，如 tableView 的 Cell 中，可以计算文字占位 Rect，动态修改视图或 cell 的高度。

17. @property (nonatomic, retain) IBOutlet UIView * view 这个对象需要 release 吗，如果需要，如何做?
其实 XIB 文件所生成的视图对象是无法真正释放的，因此关联 XIB 的属性也可以设置为 assign 属性，而视图不会释放。从这个角度上讲，这个对象不释放也可以，但是既然用 retain 做修饰符，出于尊重内存管理法则，理应进行释放，最简单的方式就是将视图在 dealloc 方法中设为 nil，适用于非 ARC，但 ARC 中这样写也没关系。

18. 两种传输协议在什么时候使用?
所谓的两种传输协议，是指网络协议中的传输层协议，即 TCP 协议和 UDP 协议。TCP 协议会在收发数据的两端建立稳定可靠，有序的链接，传输数据稳定可靠，但系统资源消耗较大，适合，数据或文件的下载或上传，Http 网络协议就是采用 TCP 传输协议传输数据的。UDP 是通俗讲得漂流瓶协议，发出协议的主机就

像扔出漂流瓶的鲁滨逊一样，不再负责数据的跟踪和校验，优点是系统资源占用低，缺陷是没有在收发两端建立稳定的传输路线，数据有丢包，损坏或后发先至等缺陷，优秀的 UDP 传输服务器，会反复发送校验序列，以保证数据的正确。

19. push 推送机制
iOS 在系统级别有一个推送服务程序使用 5223 端口。使用这个端口的协议源于 Jabber 后来发展为 XMPP ，被用于 Gtalk 等 IM 软件中。所以， iOS 的推送，可以不严谨的理解为：
苹果服务器朝手机后台挂的一个 IM 服务程序发送的消息。
然后，系统根据该 IM 消息识别告诉哪个 Apps 具体发生了什么事。
然后，系统分别通知这些 Apps 。

20. iOS 播放音频的几种方法?
iPhone OS 主要提供以下了几种播放音频的方法：
System Sound Services
AVAudioPlayer 类
Audio Queue Services
OpenAL

21. 代理的作用
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。
22. int retVal=UIApplication(argc,argv,nil,nil):是什么意思

对 UIApplication 对象进行了初始化，这个方法除了 argc 和 argv 参数外，另外这个函数还有 2 个两个字符串参数来识别 UIApplication 类和 UIApplication 代理类，在这里默认是 2 个 nil,第一个参数为 nil 就默认把 UIApplication 类作为缺省值进行初始化，可以在这里不填 nil 而是使用自己定义的 UIApplication 子类。至于第二个参数 nil 就设置为 nil 就把模板生成的 HelloWorldAppdelegate 类作为默认值。

23. 保存一个变量到本地，列举两个简单的方法
    1，用NSUserDefaults存储小量数据
    2，直接writeTofF ile
    3，存数据库
    4，归档

24. 如果UIView * view 已经实例化，在view仅添加了n 个UIButton 类的实例，这些button不是全局的，并且button已经用tag区分开，如何快速找出指定的一个button改变他的属性？

```
button=(UIButton*)[view viewWithTag:tag]
```

25. 当A类 中的某个方法执行到某处时，这时想在B类中执行某个方法，如何做？并做简单说明

用代理执行代理方法

说明：在b类中实现协议方法，设置a的代理为b，在指定方法内 调用代理的协议方法

26. oc中加号方法与减号方法的区别

加号方法是类方法，用类名直接调用

减号方法为实例方法，需要创建一个实例对象调用

27. 建一个工程用到的最基本的两个框架是？

Foundation　UIKit

28，一个UITableview的实例，重新加载数据的方法是什么？

reloadData　刷新整个表格　和 reloadSections:　withRowAnimation 刷新一组数据

29. XML有哪几种解析方式，他们各有什么优点

答：有Sax和Dom两种解析方式，sax是逐行解析。dom是一次性全部加载xml文件，然后解析

29.iOS平台怎么做数据持久化

有以下方式做

1、NSUserDefaults

2、Plist

3、数据库

4、文件保存

5、归档与反归档

30.int　retVal=UIApplication(argc,argv,nil,nil):是什么意思

对UIApplication对象进行了初始化，这个方法除了argc 和argv 参数外，另外这个函数还有2个两个字符串参数来识别UIApplication类和UIApplication代理类，在这里默认是2个nil,第一个参数为nil就默认把UIApplication类作为缺省值进行初始化，可以在这里不填nil而是使用自己定义的UIApplication子类。至于第二个参数nil就设置为nil就把模板生成的HelloWorldAppdelegate类作为默认值。

31.iOS平台怎么做数据的持久化?Core Data和SQLite有无必然联系？Core Data是一个关系型数据库吗？

答：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与sqlite还是有联系的，core data 是对sqlite的封装，因为sqlite是c语言的api，然而有人也需要obj-c 的api，所以有了 core data 另外，core data不仅仅是把c的api翻译成oc 的api，还提供了一些管理的功能，使用更加方便
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

32.runloop 是什么？在主线程中的某个函数里调用了异步函数，怎么样 block 当前线程，且还能响应 timer 事件，touch 事件等？
RunLoop

RunLoop从字面上看是运行循环的意思,这一点也不错,它确实就 是一个循环的概念,或者准确的说是线程中的循环。 本文一开始就 提到有些程序是一个圈,这个圈本质上就是这里的所谓的RunLoop, 就是一个循环,只是这个循环里加入很多特性。 首先循环体的开始需要检测是否有需要处理的事件,如果有则去处理,如果没有则进入睡眠以节省CPU时间。 所以重点便是这个需要 处理的事件,在RunLoop中,需要处理的事件分两类,一种是输入 源,一种是定时器,定时器好理解就是那些需要定时执行的操作,

输 入源分三类:performSelector源,基于端口(Mach port)的源, 以及自定义的源。编程的时候可以添加自己的源。RunLoop还有一 个观察者Observer的概念,可以往RunLoop中加入自己的 观察者以 便监控着RunLoop的运行过程,CFRunLoop.h中定义了所有观察者的

类型: 1 enum CFRunLoopActivity { kCFRunLoopEntry = (1 << 0), kCF

RunLoopBeforeTimers = (1 << 1), kCFRunLoopBeforeSour
ces = (
如果你使用过select系统调用写过程序你便可以快速的理解runloop事

件源的概念,本质上讲事件源的机制和select一样是一种多路复用IO 的 实现,在一个线程中我们需要做的事情并不单一,如需要处理定 时钟事件,需要处理用户的触控事件,需要接受网络远端发过来的 数据,将这些需要做的事情统统注册到事件源中,每一次循环的开 始便去检查这些事件源是否有需要处理的数据,有的话则去处理。 拿具体的应用举个例子,NSURLConnection网络数据请求,默认是 异步的方式,其实现原理就是创建之后将其作为事件源加入到当前 的RunLoop,而等待网络响应以及网络数据接受的过程则在一个新 创建的独立的线程中完成,当这个线程处理到某个阶段的时候比如 得到对方的响应或者接受完了网络数据之后便通知之前的线程去执 行其相关的delegate方法。所以在Cocoa中经常看到 scheduleInRunLoop:forMode: 这样的方法,这个便是将其加入到事件 源中,当检测到某个事件发生的时候,相关的delegate方法便被调用。对于CoreFoundation这一层而 言,通常的模式是创建输入源,然后 将输入源通

过CFRunLoopAddSource函数加入到RunLoop中,相关事 件发生后,相关的回调函数会被调 用。如CFSocket的使用。 另外 RunLoop中还有一个运行模式的概念,每一个运行循环必然运行在 某个模式下,而模式的存在是为了过滤事件源和观察者的,只有那 些和当前 RunLoop运行模式一致的事件源和观察者才会被激活。每一个线程都有其对应的 RunLoop,但是默认非主线程的 RunLoop 是没有运行的,需要为 RunLoop 添加至少一个事件源,然后去 run 它。一般情况下我们是没有必要去启用线程的 RunLoop 的,除非你 在一个单独的线程中需要长久的检测某个事件。

33.使用 UITableView 必须要实现的两个方法?

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

34. 简述以下在 iOS 中使用 SQLIte

```
_fmdb = [[FMDatabase alloc] initWithPath:path];
[_fmdb open];
[_fmdb executeUpdate:@"create table Article(ArticleId integer primary
key autoincrement,ArticleTitle text,ArticleContent text)"];
[_fmdb close];
```

35. 设计一个新闻浏览需要用到哪些技术

网络下载，数据解析，表单控件等

36. 谈谈你对多线程的理解

iOS 多线程分为三种 NSThread，NSOperation，GCD，NSThread 以线程为导向，NSOperation 以任务为导向，GCD 是 Block 模式的 NSOperation

36. 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做什么工作？

loadView：

每次访问UIViewController的view(比如controller.view、self.view)而且view为nil，loadView方法就会被调用。

自定义UIViewController的view用的

viewDidLoad：

无论你是通过xib文件还是重写loadView方法创建UIViewController的view，在view创建完毕后，最终都会调用viewDidLoad方法

一般我们会在这里做界面上的初始化操作，比如往view中添加一些子视图、从数据库或者网络加载模型数据装配到子视图中。

viewDidUnload：

发出内存警告且view被释放的时候就会调用viewDidUnload方法

一般在此释放资源，主要是释放界面元素相关的资源，将相关的实例都赋值为nil

37. 你用过NSOperationQueue么？如果用过或者了解的话，你为什么要使用NSOperationQueue，实现了什么？请描述它和GCD的区别和类似的地方（提示：可以从两者的实现机制和适用范围来描述）。

使用NSOperationQueue用来管理子类化的NSOperation对象，控制其线程并发数目。GCD和NSOperation都可以实现对线程的管理，区别是 NSOperation和NSOperationQueue是多线程的面向对象抽象。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。

38. 谈谈对 swift 的看法

Swift 作为 Apple 钦定的 objc 的继承者，作为 iOS/Mac 开发者的话，是觉得必须和值得学习和使用的。现在 Swift 可以和原来的 objc 或者 c 系的代码混用。因为在很多语法特性上 Swift 确实和一些脚本非常相似。但是首先需要明确的是，至少在 Apple 开发中，Swift 不是以一种脚本语言来运行的，所有的 Swift 代码都将被 LLVM 编译为 native code，以极高的效率运行。按照官方今天给出的 benchmark 数据，运行时比 Python 快 3.9 倍，比 objc 快 1.4 倍左右。我相信官方数据肯定是有些水分，但是即使这样，Swift 也给人带来很多遐想和期待。Swift 和原来的 objc 一样，是类型安全的语言，变量和方法都有明确的返回，并且变量在使用前需要进行初始化。而在语法方面，Swift 迁移到了业界公认的非常先进的语法体系，其中包含了闭包，多返回，泛型和大量的函数式编程的理念，函数也终于成为一等公民可以作为变量保存了（虽然具体实现和用法上来看和 js 那种传统意义的好像不太一样）。初步看下来语法上借鉴了很多 Ruby 的人性化的设计，但是借助于 Apple 自己手中强大的 LLVM，性能上必须要甩开 Ruby 不止一两个量级。

另一方面，Swift 的代码又是可以 Interactive 来“解释”执行的。新的 Xcode 中加入了所谓的 Playground 来对开发者输入的 Swift 代码进行交互式的相应，开发者也可是使用 swift 的命令行工具来交互式地执行 swift 语句。细心的朋友可能注意到了，我在这里把“解释”两个字打上了双引号。这是因为即使在命令行中， Swift 其实也不是被解释执行的，而是在每个指令后进对从开始以来的 swift 代码行了一遍编译，然后执行的。这样的做法下依然可以让人“感到”是在做交互解释执行，这门语言的编译速度和优化水平，可见一斑。同时 Playground 还顺便记录了每条语句的执行时候的各种情况，叫做一组 timeline。可以使用 timeline 对代码的执行逐步检查，省去了断点 debug 的时间，也非常方便。

39. 简述一下 IOS 中线程同步机制

1：原子操作 不同线程如果通过原子操作函数对同一变量进行操作，可以保证一个线程的操作不会影响到其他线程内对此变量的操作，因为这些操作都是原子式的。因为原子操作只能对内置类型进行操作，所以原子操作能够同步的线程只能位于同一个进程的地址空间内。2：锁 iOS 平台下的锁对象为 NSLock 对象，进入锁通过调用 lock 函数，解锁调用 unlock 函数（因为 iOS 中大部分的线程同步类都继承自 NSLocking 协议，所以其加锁/解锁的操作基本都为 lock/unlock 函数），同一个 NSLock 对象成功调用 lock 函数后，在其显式 unlock 之前任何线程都不能再对此 NSLock 对象加锁，以达到互斥访问的目的。3：事件 NSConditon 类型提供了 wait 与 signal 函数，分别代表了等待事件的操作以及触发事件的操作。除了 wait 函数，NSCondition 还提供了 waitUntilDate 函数，其功能与 NSLock 中的 lockBeforeDate 大致相同，简要来说就是提供了一个带超时的 wait 函数。

40. 启动一个线程，在子线程中如何刷新界面。

可以跳转到主线程中进行界面的刷新，如[self performSelectorOnMainThread:@selector(updateUI) withObject:nil waitUntilDone:YES];

41. 简述开发中使用过的设计模式。

（一）代理模式

应用场景：当一个类的某些功能需要由别的类来实现，但是又不确定具体会是哪个类实现。

优势：解耦合

敏捷原则：开放-封闭原则

实例：tableview 的 数据源 delegate，通过和 protocol 的配合，完成委托诉求。

列表 row 个数 delegate

自定义的 delegate

（二）观察者模式

应用场景：一般为 model 层对，controller 和 view 进行的通知方式，不关心谁去接收，只负责发布信息。

优势：解耦合

敏捷原则：接口隔离原则，开放-封闭原则

实例：Notification 通知中心，注册通知中心，任何位置可以发送消息，注册观察者的对象可以接收。

kvo，键值对改变通知的观察者，平时基本没用过。

（三）MVC 模式

应用场景：是一中非常古老的设计模式，通过数据模型，控制器逻辑，视图展示将应用程序进行逻辑划分。

优势：使系统，层次清晰，职责分明，易于维护

敏捷原则：对扩展开放-对修改封闭

实例：model-即数据模型，view-视图展示，controller 进行 UI 展现和数据交互的逻辑控制。

（四）单例模式
应用场景：确保程序运行期某个类，只有一份实例，用于进行资源共享控制。
优势：使用简单，延时求值，易于跨模块
敏捷原则：单一职责原则
实例：[UIApplication sharedApplication]。
注意事项：确保使用者只能通过 getInstance 方法才能获得，单例类的唯一实例。
java，C++中使其没有公有构造函数，私有化并覆盖其构造函数。
object c 中，重写 allocWithZone 方法，保证即使用户用 alloc 方法直接创建单例类的实例，
返回的也只是此单例类的唯一静态变量。

（五）策略模式
应用场景：定义算法族，封装起来，使他们之间可以相互替换。
优势：使算法的变化独立于使用算法的用户
敏捷原则：接口隔离原则；多用组合，少用继承；针对接口编程，而非实现。
实例：排序算法，NSArray 的 sortedArrayUsingSelector；经典的鸭子会叫，会飞案例。
注意事项：1，剥离类中易于变化的行为，通过组合的方式嵌入抽象基类
2，变化的行为抽象基类为，所有可变变化的父类
3，用户类的最终实例，通过注入行为实例的方式，设定易变行为
防止了继承行为方式，导致无关行为污染子类。完成了策略封装和可替换性。

（六）工厂模式
应用场景：工厂方式创建类的实例，多与 proxy 模式配合，创建可替换代理类。
优势：易于替换，面向抽象编程，application 只与抽象工厂和易变类的共性抽象类发生调用关系。
敏捷原则：DIP 依赖倒置原则
实例：项目部署环境中依赖多个不同类型的数据库时，需要使用工厂配合 proxy 完成易用性替换
注意事项：项目初期，软件结构和需求都没有稳定下来时，不建议使用此模式，因为其劣势也很明显，
增 加了代码的复杂度，增加了调用层次，增加了内存负担。所以要注意防止模式的滥用。
42. UIViewController 中的 viewDidLoad,viewWillAppear,viewDidUnload,dealloc 分别是在什么时候调用？
viewDidLoad:方法
在视图加载后被调用：
如果是在代码中创建的视图加载器，他将会在 loadView 方法后被调用；
如果是从 nib 视图页面输出，他将会在视图设置好后后被调用。

重载重写该方法以进一步定制 view
在 iPhone OS 3.0 及之后的版本中，还应该重载重写 viewDidUnload 来释放对 view 的任何索引
viewDidLoad 后调用数据 Model viewWillAppear: 方法
Called when the view is about to made visible. Default does nothing
视图即将可见时调用。默认情况下不执行任何操作 viewDidUnload:方法
当系统内存吃紧的时候会调用该方法（注：viewController 没有被 dealloc）
内存吃紧时，在 iPhone OS 3.0 之前 didReceiveMemoryWarning 是释放无用内存的唯一方式，但是 OS 3.0 及以后 viewDidUnload 方法是更好的方式
在该方法中将所有 IBOutlet（无论是 property 还是实例变量）置为 nil（系统 release view 时已经将其 release 掉了）
在该方法中释放其他与 view 有关的对象、其他在运行时创建（但非系统必须）的对象、在 viewDidLoad 中被创建的对象、缓存数据等 release 对象后，将对象置为 nil（IBOutlet 只需要将其置为 nil，系统 release view 时已经将其 release 掉了）
一般认为 viewDidUnload 是 viewDidLoad 的镜像，因为当 view 被重新请求时，viewDidLoad 还会重新被执行
viewDidUnload 中被 release 的对象必须是很容易被重新创建的对象（比如在 viewDidLoad 或其他方法中创建的对象），不要 release 用户数据或其他很难被重新创建的对象

dealloc:方法
viewDidUnload 和 dealloc 方法没有关联，dealloc 还是继续做它该做的事情
43.navigationbar 的背景颜色设置

```
UINavigationController* nav = [[UINavigationController alloc] init];

self.nav.navigationBar.tintColor = [UIColor blackColor];
```

44.tableviewcell 的那几个函数

```
 // Designated initializer.  If the cell can be reused, you must pass in a reuse identifier.  You should use the same reuse identifier for all cells of the same form.
- (id)initWithStyle:(UITableViewCellStyle)style reuseIdentifier:(NSString *)reuseIdentifier

- (void)prepareForReuse;
// if the cell is reusable (has a reuse identifier), this is called just before the cell is returned from the table view method dequeueReusableCellWithIdentifier:.  If you override, you MUST call super.
```

```
- (void)setSelected:(BOOL)selected animated:(BOOL)animated;
// animate between regular and selected state
- (void)setHighlighted:(BOOL)highlighted animated:(BOOL)animated;
// animate between regular and highlighted state
- (void)setEditing:(BOOL)editing animated:(BOOL)animated;
```

45.还有动画
在 iOS 中动画实现技术主要是：Core Animation。

Core Animation 负责所有的滚动、旋转、缩小和放大以及所有的 iOS 动画效果。其中 UIKit 类通常都有 animated：参数部分，它可以允许是否使用动画。

Core Animation 主要是使用

我们知道每个 UIView 都关联到一个 CALayer 对象，CALayer 是 Core Animation 中的图层。

Core Animation 主要就是通过修改图层来改变 UI 的大小，位置，从而实现动画效果。

可以说，任何一个应用程序都离不开动画！
就连苹果各个 UI 控件中的切换操作，都有它内在的动画。

了解一下，关于动画的一些知识。

任何知识点，都会迁出一系列的知识点。

```
[UIView beginAnimations:@"dropDownloadLabel"
context:UIGraphicsGetCurrentContext()];
[UIView setAnimationDuration: 0.5];
[UIView setAnimationBeginsFromCurrentState: NO];

// 执行的动画 code

[UIView commitAnimations];
```

就将这段段代码作为知识的切入点，开始了解吧。

```
[UIView beginAnimations:@"dropDownloadLabel"
context:UIGraphicsGetCurrentContext()];
[UIView commitAnimations];
```

这两句代码，标记了一个动画的开始和结束。在中间我们可以写我们的一些动画操作！

beginAnimations 方法

+ (void)beginAnimations:(NSString *)animationID context:(void *)context
用来，表示动画的开始。
animationID：作为动画的标识
context：自定义的一些动画数据，这些数据将发送给动画的代理方法：setAnimationWillStartSelector:方法和 setAnimationDidStopSelector:方法。
这个，参数，通常为 nil。我们可以直接设置为 nil。
这里，我们使用 UIGraphicsGetCurrentContext()；因为此方法默认也会返回 nil。

该方法告诉系统，我们将开始动画。并且，在该方法后，我们可以通过 setAnimationXXX（一系列方法）来设置我们进行的动画的一些参数。
完成动画后，调用 commitAnimations 方法来通知系统，动画结束。

至此，我们知道，就是设置动画的一些列参数的方法即 setAnimationXXX 方法。

```
[UIView setAnimationDuration: 0.5];
[UIView setAnimationBeginsFromCurrentState: NO];
```

动画是可以嵌套的。

```
[UIView beginAnimations:@"animation_1"
context:UIGraphicsGetCurrentContext()];
// code1
[UIView beginAnimations:@"animation_2"
context:UIGraphicsGetCurrentContext()];
// code2
[UIView commitAnimations];

[UIView commitAnimations];
```

如果我们为动画设置了，setAnimationWillStartSelector:方法和 setAnimationDidStopSelector:方法。
那么当动画开始或者停止的时候，动画的 animationID 参数和 context 参数，会传递给 setAnimationWillStartSelector:方法和 setAnimationDidStopSelector:方法。

悲剧总是要发生的！
苹果 API 在最后的描述中，给了这么一句话：

Use of this method is discouraged in iOS 4.0 and later. You should use the block-based animation methods to specify your animations instead.

可见，在 iOS 4.0 后，block 语法，大大增多了。这种方式，是不建议的，需要我们使用 block 的方式。

于是，动画的 block 方式：

```
[UIView animateWithDuration:0.3f delay:0.0f
options:UIViewAnimationOptionCurveLinear
                              animations:^{ // 执行的动画 code}
                              completion:^(BOOL finished){
                                  // 完成后执行 code
                              }];
```

在尽量用 block 来完成动画，因为说不定啥时候，老的动画方式，将被废除。

到此，可以告一段落。但是，我想将这简单的动画代码，一查到底！

commitAnimations 方法：
+ (void)commitAnimations

标记动画结束。与 beginAnimations 方法成对使用。
例如：
[UIView commitAnimations];

一系列的 setAnimationXXX 方法：

setAnimationDuration 方法：

+ (void)setAnimationDuration:(NSTimeInterval)duration

设置动画持续时间（秒）

例如：
[UIView setAnimationDuration: 0.5];

setAnimationBeginsFromCurrentState 方法
+ (void)setAnimationBeginsFromCurrentState:(BOOL)fromCurrentState

设置动画开始时的状态。

我们构想一个场景：一般，我们按下一个按钮，将会执行动画一次。

当 YES 时：当上一次动画正在执行中，那么当下一个动画开始时，上一次动画的当前状态将成为下一次动画的开始状态。
当 NO 时：当上一个动画正在执行中，那么当下一个动画开始时，上一次动画需要先恢复到完成时的状态，然后在开始执行下一次动画。

setAnimationStartDate 方法

+ (void)setAnimationStartDate:(NSDate *)startTime

设置动画开始时间。

setAnimationDelay 方法

+ (void)setAnimationDelay:(NSTimeInterval)delay

设置画开始的延迟时间（秒）。

setAnimationCurve 方法

+ (void)setAnimationCurve:(UIViewAnimationCurve)curve

设置动画的曲线方式（就是动画的总体变化的时间曲线：开始快最后慢，开始慢最后快，最后慢，均匀线性）。

curve 参数如下：

```
typedef NS_ENUM(NSInteger, UIViewAnimationCurve) {
        UIViewAnimationCurveEaseInOut,          // slow at beginning
and end
        UIViewAnimationCurveEaseIn,             // slow at beginning
        UIViewAnimationCurveEaseOut,            // slow at end
        UIViewAnimationCurveLinear
    };
```

setAnimationRepeatCount 方法

+ (void)setAnimationRepeatCount:(float)repeatCount

设置 动画重复次数

setAnimationRepeatAutoreverses 方法

+ (void)setAnimationRepeatAutoreverses:(BOOL)repeatAutoreverses

设置动画是否做一次反向的执行。
如果设置为 YES：动画将执行：动画初始状态》动画》动画完成状态》动画》动画初始状态。
如果设置为 NO：默认值

setAnimationsEnabled 方法

+ (void)setAnimationsEnabled:(BOOL)enabled

设置动画是否可用！
YES：默认值。
NO：动画效果被禁用
注意：仅仅是动画是否可用，在动画中被改变的 UI 对象依然是起作用的。仅仅是动画效果被禁用了。

areAnimationsEnabled 方法

+ (BOOL)areAnimationsEnabled

返回动画效果是否被禁用。

46.还有一个按着导航栏颜色变亮的

```
_myNav.navigationBar.translucent = YES;

 _myNav.navigationBar.barStyle = UIBarStyleBlack;
```

47.还有个版本的问题

一、配置 SVN 服务器

1、创建 Svn 服务工作路径同时新建我们的 App 工程，入下图所示，
SVN_Project 是 SVN 服务的工作路径，MyProject 是我们的 iOS 工程

2，在 Mac 下有自带的 svn 服务功能（Windows 下是没有的），直接在终端打开 svn 的服务即可，在打开服务的同时指定 svn 的工作路径

（1）启动 svn 服务命令：svnserve -d -r 工作路径：

（2）终端截图如下：

3、svn 服务启动后，要创建 svn 管理文件，管理文件有关于 svn 的各种配置

(1)在工作目录中创建管理文件命令：svnadmin create MySVNProject

(2)管理文件创建成功后，其目录结构如下：

(3)接下来要配置我们的 svn,打开 conf 文件夹如下：

(4)配置 svnserve.conf 文件，把带一个#的临时注释去掉即可：

（5）在 passwd 中添加用户 名和密码

(6)authz 中是用户组的管理

二.把工程导入 SVN

1.想把我们的工程导入 svn 的话，需要用到一个工具 svnx，svnx 连接 svn 服务器，后面跟的文件是 SVN 的管理文件,用户名和密码就是在配置文件中添加的用户名和密码

2.登陆成功以后，导入我们的 iOS 工程。

三、在 Xcode 中 check out 工程（下面用的时 Xcode6.1 的测试版本）

1.在 Welcome Xcode 中选中 Check out an existing Project,入下图所示：

2.连接 svn 服务器（ip 后面的仍然是 svn 管理文件）：

3、check out 工程

4.在本地打开工程，在 Source Control 中进行项目的管理

有自己的产品也有外包

编程：

1、给定字符串的长度，还有换行方式 算出高度。

```
UIFont *font = [UIFont systemFontOfSize:14];
CGSize size = [text sizeWithFont:font
constrainedToSize:CGSizeMake(140, 1000)
lineBreakMode:UILineBreakModeCharacterWrap];
```

2. 写一个发送同步 http 请求，并获得返回结果的方法。

```
NSMutableURLRequest *request = [[NSMutableURLRequest alloc] init];

 [request setURL:[NSURL URLWithString:urlStr]];

 [request setHTTPMethod:@"GET"];

 NSData *returnData = [NSURLConnection sendSynchronousRequest:request
                    returningResponse:nil error:nil];

 [request release];
```