PS：答案只供参考。

技术基础

1. 为什么说Objective-C是一门动态的语言？

|  |
| --- |
| 静态、动态是相对的，这里动态语言指的是不需要在编译时确定所有的东西，在运行时还可以动态的添加变量、方法和类  Objective-C 可以通过[Runtime](https://baike.baidu.com/item/runtime/4723760?fr=aladdin" \t "_blank) 这个运行时机制，在运行时动态的添加变量、方法、类等，所以说Objective-C 是一门动态的语言  **其他说法**  Objective-C 是C 的超集，在C 语言的基础上添加了面向对象特性，并且利用[Runtime](https://baike.baidu.com/item/runtime/4723760?fr=aladdin" \t "_blank) 这个运行时机制，为Objective-C 增添了动态的特性。  Objective-C 使用的是 “消息结构” 并非 “函数调用”：使用消息结构的的语言，其运行时所应执行的代码由运行期决定；而使用函数调用的语言，则由编译器决定   1. 动态类型语言：动态类型语言是指在运行期间才去做数据类型检查的语言，也就是说，在用动态类型的语言编程时，永远也不用给任何变量指定数据类型，该语言会在你第一次赋值给变量时，在内部将数据类型记录下来。Python和Ruby就是一种典型的动态类型语言，其他的各种脚本语言如VBScript也多少属于动态类型语言。 2. 静态类型语言：静态类型语言与动态类型语言刚好相反，它的数据类型是在编译其间检查的，也就是说在写程序时要声明所有变量的数据类型，C/C++是静态类型语言的典型代表，其他的静态类型语言还有C#、JAVA等。   因为在运行期可以继续向类中添加方法，所以编译器在编译时还无法确定类中是否有某个方法的实现。对于类无法处理一个消息就会触发消息转发机制  消息转发分为两大阶段：   * “动态方法解析”：先征询接收者，所属的类，能否动态添加方法，来处理这个消息，若可以则结束，如不能则继续往下走 * “完整的消息转发机制”： 请接收者看看有没其他对象能处理这条消息，若有，就把这个消息转发给那个对象然后结束运行时系统会把与消息有关细节全部封装到NSInvocation 对象中，再给对象最后一次机会，令其设法解决当前还未处理的这条消息   **Objective-c的动态性**  Objective-C的动态性，让程序在运行时判断其该有的行为，而不是像c等静态语言在编译构建时就确定下来。它的动态性主要体现在3个方面：   * 动态类型：如id类型。实际上静态类型因为其固定性和可预知性而使用的特别广泛。静态类型是强类型，动态类型是弱类型，运行时决定接收者。 * 动态绑定：让代码在运行时判断需要调用什么方法，而不是在编译时。与其他面向对象语言一样，方法调用和代码并没有在编译时连接在一起，而是在消息发送时才进行连接。运行时决定调用哪个方法。 * 动态载入。让程序在运行时添加代码模块以及其他资源。用户可以根据需要执行一些可执行代码和资源，而不是在启动时就加载所有组件。可执行代码中可以含有和程序运行时整合的新类。 |

1. 讲一下MVC和MVVM，MVP？(具体举例在项目中如何使用)

|  |
| --- |
| **MVC**全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写,Model 呈现数据，View 呈现用户界面，而 View Controller 调节它两者之间的交互。   * **视图**(View)：构建UI的类；例如：按钮，标签，进度条等。 * **模型**(Model)：视图类所需要的数据；例如：表格需要显示的文字。 * **控制器**(Controller)：连接视图类和模型类，任务是使数据显示在屏幕上。   **MVVM**全名是Model View View-Model，ViewModel将表示逻辑从 Controller 移出放到一个新的对象里，即 View Model。就是那些将 Model 数据转换为 View 可以呈现的东西的事情，例如将一个 NSDate 转换为一个格式化过的 NSString。  **MVP**全名是Model-view-presenter，是使用者界面设计模式的一种。   * **Model** 定义使用者界面所需要被显示的数据模型，一个模型包含着相关的业务逻辑。 * **View** 视图为呈现使用者界面的终端，用以表现来自 Model 的数据，和使用者命令路由再经过 Presenter 对事件处理后的数据。 * **Presenter** 包含着元件的事件处理，负责检索 Model 取得数据，和将取得的数据经过格式转换与 View 进行沟通。 * **MVP** 设计模式通常会再加上 Controller 做为整体应用程序的后端程序工作。 |

1. 线程间通信？

|  |
| --- |
| 使用全局变量主要由于多个线程可能更改全局变量，因此全局变量最好声明为violate  使用消息实现通信在Windows程序设计中，每一个线程都可以拥有自己的消息队列（UI线程默认自带消息队列和消息循环，工作线程需要手动实现消息循环），因此可以采用消息进行线程间通信sendMessage,postMessage。   1. 定义消息#define WM\_THREAD\_SENDMSG=WM\_USER+20; 2. 添加消息函数声明afx\_msg int OnTSendmsg(); 3. 添加消息映射ON\_MESSAGE(WM\_THREAD\_SENDMSG,OnTSM) 4. 添加OnTSM()的实现函数； 5. 在线程函数中添加PostMessage消息Post函数   使用事件CEvent类实现线程间通信  Event对象有两种状态：有信号和无信号，线程可以监视处于有信号状态的事件，以便在适当的时候执行对事件的操作。   1. 创建一个CEvent类的对象：CEvent threadStart;它默认处在未通信状态； 2. threadStart.SetEvent();使其处于通信状态； 3. 调用WaitForSingleObject()来监视CEvent对象   线程间的通信、同步方式与进程间通信方式 |

1. 数据持久化的几个方案(针对某一种简述如何使用)

|  |
| --- |
| 1. plist文件（属性列表） 2. preference（偏好设置） 3. NSKeyedArchiver（归档） 4. SQLite 3 5. CoreData |

5、objc使用什么机制管理对象内存？

|  |
| --- |
| MRC(manual retain-release)手动内存管理  ARC(automatic reference counting)自动引用计数  Garbage collection (垃圾回收)。但是iOS不支持垃圾回收, ARC作为LLVM3.0编译器的一项特性, 在iOS5.0 (Xcode4) 版本后推出的。  ARC的判断准则, 只要没有强指针指向对象, 对象就会被释放. |

Runtime

1、objc在向一个对象发送消息时，发生了什么？

|  |
| --- |
| objc在向一个对象发送消息时，runtime库会根据对象的isa指针找到该对象实际所属的类，然后在该类中的方法列表以及其父类方法列表中寻找方法运行，然后在发送消息的时候，objc\_msgSend方法不会返回值，所谓的返回内容都是具体调用时执行的  如果向一个nil对象发送消息，首先在寻找对象的isa指针时就是0地址返回了，所以不会出现任何错误。 |

2、runtime如何实现weak变量的自动置nil？

|  |
| --- |
| runtime 对注册的类， 会进行布局，对于 weak 对象会放入一个 hash 表中。 用 weak 指向的对象内存地址作为 key，当此对象的引用计数为0的时候会 dealloc， 在这个 weak 表中搜索，找到所有以a为键的 weak 对象，从而设置为 nil。  weak 修饰的指针默认值是 nil （在Objective-C中向nil发送消息是安全的） |

高级

1、SDWebImage的缓存策略？（提出缓存的不足或针对某一个过程讲解一下，比如图片缓冲实现）

|  |
| --- |
| SDweSDWebImage提供一个UIImageView的类别以支持加载来自互联网的远程图片。具有缓存管理、异步下载，同一个URL下载次数控制和优化等特征。  **独立的异步图像下载**  可能会用到单独的异步图片下载，则一定要用 - (id <SDWebImageOperation>)downloadImageWithURL:(NSURL \*)url options:(SDWebImageOptions)options progress:(SDWebImageDownloaderProgressBlock)progressBlock completed:(SDWebImageCompletionWithFinishedBlock)completedBlock; 来建立一个SDWebImageDownLoader 的实例。这样就可以有下载进度的回调和下载完成的回调，可以在回调完成进度条相关的操作和显示图片相关的操作。  **独立的异步图像缓存**  SDImageCache类提供一个管理缓存的单例类。 SDImageCache \*imageCache = [SDImageCache sharedImageCache] 查找和缓存图片时以URL作为key。(先查找内存，如果内存不存在该图片，再查找硬盘；查找硬盘时，以URL的MD5值作为key). 查找图片： UIImage \*cacheImage = [imageCache mageFromKey:myCacheKey]; 缓存图片： [ imageCache storeImage:myImage forKey:myCacheKey]; 默认情况下，图片是被存储到内存缓存和磁盘缓存中的。如果仅仅是想缓存到内存中，可以用下面方法： storeImage:forKey:toDisk: 第三个参数传NO即可。  **主要用到的对象：**   * 1.UIImageView(WebCache)，入口封装，实现读取图片完成后的回调。 * 2.SDWebImagemanager,对图片进行管理的中转站，记录那些图片正在读取。向下层读取Cache（调用SDImageCache），或者向网络读取对象（调用SDWebImageDownloader）。实现SDImageCache和SDWebImageDownLoader的回调。 * 3.SDImageCache,根据URL作为key，对图片进行存储和读取（存在内存（以URL作为key）和存在硬盘两种（以URL的MD5值作为key））。实现图片和内存清理工作。   **SDWebImage加载图片的流程**   * 1.入口 setImageWithURL:placeholderImage:options:会先把 placeholderImage显示，然后 SDWebImageManager根据 URL 开始处理图片。 * 2.进入SDWebImageManager 类中downloadWithURL:delegate:options:userInfo:，交给 SDImageCache从缓存查找图片是否已经下载 queryDiskCacheForKey:delegate:userInfo:. * 3.先从内存图片缓存查找是否有图片，如果内存中已经有图片缓存，SDImageCacheDelegate回调 imageCache:didFindImage:forKey:userInfo:到 SDWebImageManager。 * 4.SDWebImageManagerDelegate 回调 webImageManager:didFinishWithImage: 到 UIImageView+WebCache,等前端展示图片。 * 5.如果内存缓存中没有，生成 ｀NSOperation ｀ 添加到队列，开始从硬盘查找图片是否已经缓存。 * 6.根据 URL的MD5值Key在硬盘缓存目录下尝试读取图片文件。这一步是在 NSOperation 进行的操作，所以回主线程进行结果回调 notifyDelegate:。 * 7.如果上一操作从硬盘读取到了图片，将图片添加到内存缓存中（如果空闲内存过小， 会先清空内存缓存）。SDImageCacheDelegate'回调 imageCache:didFindImage:forKey:userInfo:`。进而回调展示图片。 * 8.如果从硬盘缓存目录读取不到图片，说明所有缓存都不存在该图片，需要下载图片， 回调 imageCache:didNotFindImageForKey:userInfo:。 * 9.共享或重新生成一个下载器 SDWebImageDownloader开始下载图片。 * 10.图片下载由 NSURLConnection来做，实现相关 delegate 来判断图片下载中、下载完成和下载失败。 * 11.connection:didReceiveData: 中利用 ImageIO做了按图片下载进度加载效果。 * 12.connectionDidFinishLoading: 数据下载完成后交给 SDWebImageDecoder做图片解码处理。 * 13.图片解码处理在一个 NSOperationQueue完成，不会拖慢主线程 UI.如果有需要 对下载的图片进行二次处理，最好也在这里完成，效率会好很多。 * 14.在主线程 notifyDelegateOnMainThreadWithInfo: 宣告解码完成 imageDecoder:didFinishDecodingImage:userInfo: 回调给 SDWebImageDownloader`。 * 15.imageDownloader:didFinishWithImage:回调给 SDWebImageManager告知图片 下载完成。 -16. 通知所有的 downloadDelegates下载完成，回调给需要的地方展示图片。 * 17.将图片保存到 SDImageCache中，内存缓存和硬盘缓存同时保存。写文件到硬盘 也在以单独 NSOperation 完成，避免拖慢主线程。 * 18.SDImageCache 在初始化的时候会注册一些消息通知， 在内存警告或退到后台的时 候清理内存图片缓存，应用结束的时候清理过期图片。 |

2、KVO的使用？实现原理？（为什么要创建子类来实现）

|  |
| --- |
| **一、KVO 是什么？**  KVO 是 Objective-C 对***观察者设计模式***的一种实现。【另外一种是：通知机制（notification），详情参考：*[iOS 趣谈设计模式——通知](https://www.jianshu.com/p/ebd1953a655d" \t "_blank)*】； KVO 提供一种机制，指定一个被观察对象(例如 A 类)，当对象某个属性(例如 A 中的字符串 name)发生更改时，对象会获得通知，并作出相应处理；【且不需要给被观察的对象添加任何额外代码，就能使用 KVO 机制】  在 MVC 设计架构下的项目，KVO 机制很适合实现 mode 模型和 view 视图之间的通讯。 例如：代码中，在模型类A创建属性数据，在控制器中创建观察者，一旦属性数据发生改变就收到观察者收到通知，通过 KVO 再在控制器使用回调方法处理实现视图 B 的更新；(本文中的应用就是这样的例子.)  **二、实现原理？**  KVO 在 Apple 中的 API 文档如下：  Automatic key-value observing is implemented using a technique called  isa-swizzling… When an observer is registered for an attribute of an object the  isa pointer of the observed object is modified, pointing to an intermediate class  rather than at the true class …  KVO 的实现依赖于 Objective-C 强大的 Runtime，从以上 Apple 的文档可以看出苹果对于 KVO 机制的实现是一笔带过，而具体的细节没有过多的描述，但是我们可以通过 Runtime 的所提供的方法去探索【可参考：*[Runtime的几个小例子](https://www.jianshu.com/p/ed65518ec8db" \t "_blank)*】，关于KVO 机制的底层实现原理。为此啊左从网上的一些关于 KVO 的资料总结了有关的内容：  **基本的原理：**  当观察某对象 A 时，KVO 机制动态创建一个对象A当前类的子类，并为这个新的子类重写了被观察属性 keyPath 的 setter 方法。setter 方法随后负责通知观察对象属性的改变状况。 |

编辑人： 韩旗

时间：2018年9月13日