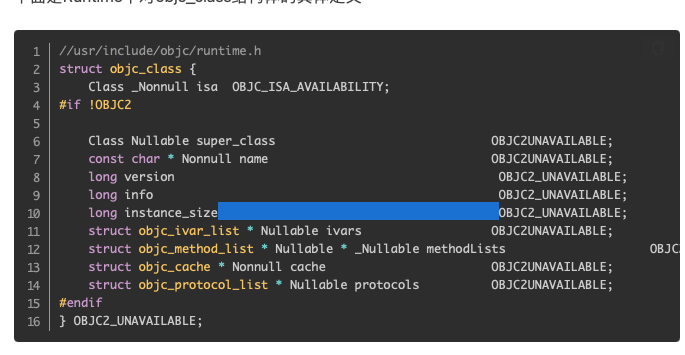
Runtime

一、runtime的概念  
 runtime（简称运行时），是一套 纯C(C和汇编写的) 的API。而 OC 就是 运行时机制，也就是在运行时候的一些机制，其中最主要的是 消息机制。  
OC的函数调用成为消息发送，属于 动态调用过程。在编译的时候并不能决定真正调用哪个函数，只有在真正运行的时候才会根据函数的名称找到对应的函数来调用。（简单来说就是在编译时可以调用未实现的方法，只有用到时进行消息查找未找到时才会报错）

事实证明：在编译阶段，OC 可以 调用任何函数，即使这个函数并未实现，只要声明过就不会报错，只有当运行的时候才会报错，这是因为OC是运行时动态调用的。而 C 语言 调用未实现的函数 就会报错。所以runtime也涉及到了 oc的消息调用机制。objc\_msgSend是runtime核心，所有的方法调用都是用这个方法。并不是直接调用方法实现(IMP)而是发送消息，让类的结构体去动态查到方法实现，所以在为查找到方法实现之前我们可以动态的去修改这个方法的实现

Isa指针，不管id，还是Class最后指向的都是objc\_class这个结构体  
objc\_class结构体中的定义如下：



结构体数据分析：

* 1. isa Class对象，指向objc\_class结构体的指针，也就是这个Class的MetaClass(元类)。类的实例对象的 isa 指向该类;该类的 isa 指向该类的 MetaClass。MetaCalss的isa对象指向RootMetaCalss。 super\_class Class对象指向父类对象。如果该类的对象已经是RootClass，那么这个super\_class指向nil。MetaCalss的SuperClass指向父类的MetaCalss。MetaCalss是RootMetaCalss，那么该MetaClass的SuperClass指向该对象的RootClass。
  2. ivars 类中所有属性的列表，使用场景：我们在字典转换成模型的时候需要用到这个列表找到属性的名称，去取字典中的值，KVC赋值，或者直接Runtime赋值。
  3. methodLists 类中所有的方法的列表，类中所有方法的列表，使用场景：如在程序中写好方法，通过外部获取到方法名称字符串，然后通过这个字符串得到方法，从而达到外部控制App已知方法。
  4. cache 主要用于缓存常用方法列表，每个类中有很多方法，我平时不用的方法也会在里面，每次运行一个方法，都要去methodLists遍历得到方法，如果类的方法不多还行，但是基本的类中都会有很多方法，这样势必会影响程序的运行效率，所以cache在这里就会被用上，当我们使用这个类的方法时先判断cache是否为空，为空从methodLists找到调用，并保存到cache，不为空先从cache中找方法，如果找不到在去methodLists，这样提高了程序方法的运行效率
  5. protocols 故名思义，这个类中都遵守了哪些协议，使用场景：判断类是否遵守了某个协议上

二、运行时阶段的消息发送。：

2.1、检测selector 是不是需要忽略的。比如 Mac OS X 开发，有了垃圾回收就不理会retain,release 这些函数了。

2.2、检测target 是不是nil 对象。ObjC 的特性是允许对一个 nil对象执行任何一个方法不会 Crash，因为会被忽略掉。

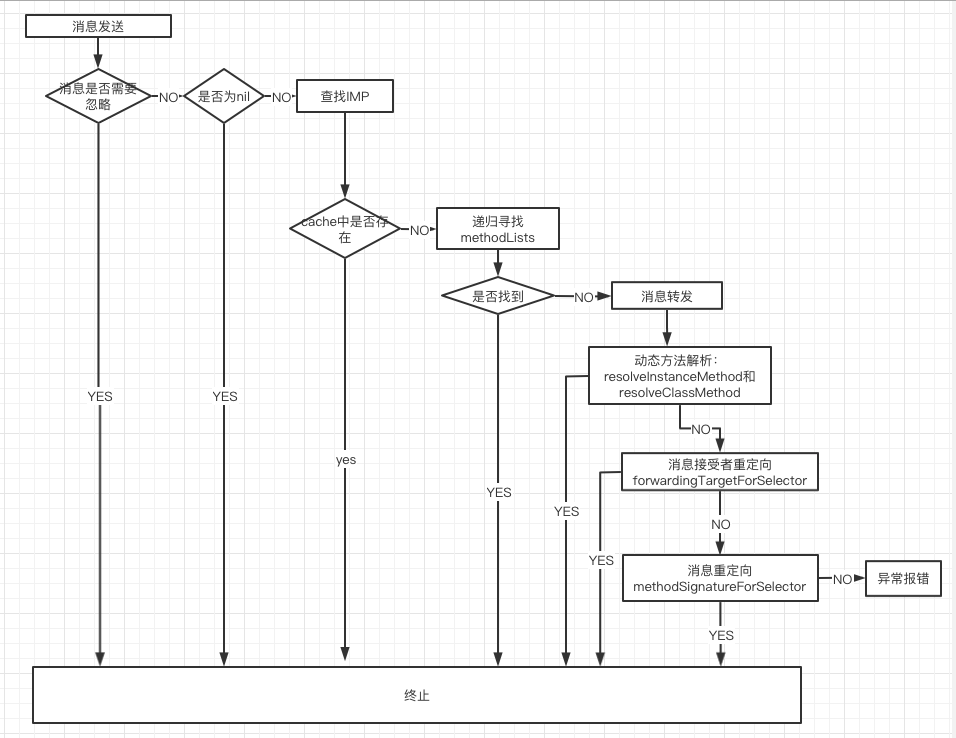
2.3、如果上面两个都过了，那就开始查找这个类的 IMP，先从 cache 里面找，若可以找得到就跳到对应的函数去执行。

2.4、如果在cache里找不到就找一下方法列表methodLists。

如果methodLists找不到，就到超类的方法列表里寻找，一直找，直到找到NSObject类为止。

2.5、如果还找不到，Runtime就提供了如下三种方法来处理：动态方法解析、消息接受者重定向、消息重定向。

消息发送流程图：



对于方法防错处理也就是在消息转发中，系统给出了三次防错机会。如果三次都没成功则会报错，未找到方法。

三、runtime的使用场景  
3.1、动态交换两个方法的实现

3.2、动态添加属性

3.3、实现字典转模型的自动转换

3.4、动态添加方法

3.5、拦截并替换方法

四、runtime 的示例。详细见runtimeDemo的代码  
五、沃阅读中的使用场景有。

5.1、数组越界和字典为空防错。通过方法交换进行防错判断。  
5.2、NSNull对象防错。动态方法解析进行防错处理。

5.3、字典转模型，实际是通过MJModel 封装的处理，里面用到的技术也是runtime。  
5.4、南研sdk也面打点，因需将所有页面切换时进行打点，所以通过方法交换。在viewWillAppear和viewDidDisappear进行打点。  
5.5、未找到方法防错处理。通过方法交换forwardingTargetForSelector然后将消息对象重定向到特殊类，然后resolveInstanceMethod中添加默认方法即可  
  
  
  
  
  
https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ObjCRuntimeGuide/Articles/ocrtTypeEncodings.html#//apple\_ref/doc/uid/TP40008048-CH100 数据类型。