weak实现原理：Runtime维护了一个weak表，用于存储指向某个对象的所有weak指针。weak表其实是一个hash（哈希）表，Key是所指对象的地址，Value是weak指针的地址（这个地址的值是所指对象的地址）数组。

当weak指向的对象释放之后，weak指针置为空；当__unsafe_unretained指向的对象释放之后，指针仍然存在，此时使用可能造成崩溃。

假如Controller太臃肿，如何优化？

构造viewmodel　viewmodel的指针指向vc(weak)，vc的指针指向viewmodel(Strong)

使用分类的方法

工具类封装

网络请求封装

UI的封装

整合常量

消息转发机制原理：

消息转发机制基本分为三个步骤：

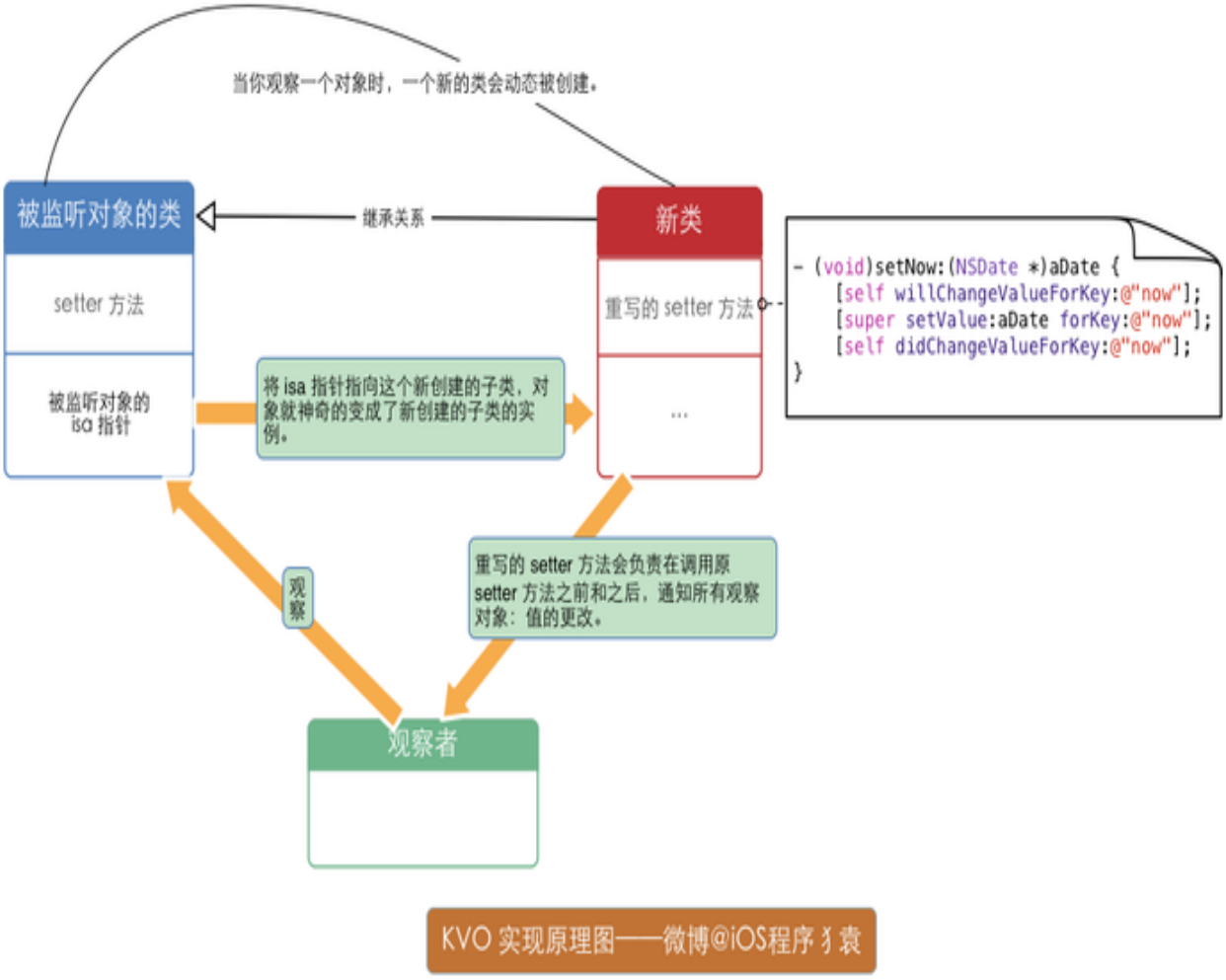
1、动态方法解析 在resolveInstanceMethod方法中可以通过class_addMethod补救

2、备用接受者 在forwardingTargetForSelector可以renturn一个备用的接受者

3、完整转发 forwardInvocation把这个消息转发

KVO的理解：

基于runtime实现，大概就是新建一个类(新类)，新类继承于旧类；将旧类对象的isa指针指向新类；重写新类的set方法，set方法的作用是一是执行父类的set方法，一是通知观察者



KVO 实现原理图——微博@iOS程序犭袁

NSString为什么用copy不用strong：

当要赋值的对象是NSMutableString类型时，这时如果使用strong，这只是一个简单的指针拷贝，当NSMutableString类型的数据改变时，用strong修饰的属性也会发生改变，同理用copy时会把NSMutableString类型的数据深拷贝，不会发生这种情况

为什么ios是动态语言：

1.动态类型：

即运行时再决定对象的类型(根据isa指针找到对应的类)。简单说就是id类型；像内置的明确的基本类型都属于静态类型(int、NSString等)。静态类型在编译的时候就能被识别出来。而动态类型就编译器编译的时候是不能被识别的，要等到运行时(run time)，即程序运行的时候才会根据语境来识别。

2.动态绑定：

基于动态类型，在某个实例对象被确定后，其类型便被确定了。该对象对应的属性和响应的消息也被完全确定，这就是动态绑定。比如我们一般向一个NSObject对象发送-respondsToSelector:或者 -instancesRespondToSelector:等来确定对象是否可以对某个SEL做出响应，而在OC消息转发机制被触发之前，对应的类 的+resolveClassMethod:和+resolveInstanceMethod:将会被调用，在此时有机会动态地

向类或者实例添加新的 法，也即类的实现是可以动态绑定的；

3.动态加载：

所谓动态加载就是我们做开发的时候icon图片的时候在Retina设备上要多添加一个张@2x的图片,当设备更换的时候,图片也会自动的替换。

***MVVM和MVC：**

MVC:简单来说就是，逻辑、试图、数据进行分层，实现解耦。

MVVM:是Model-View-ViewMode模式的简称；比MVC更加释放控制器臃肿，将一部分逻辑(耗时，公共方法，网络请求等)和数据的处理等操作从控制器里面搬运到ViewModel中

MVVM的特点：

低耦合，View可以独立于Model变化和修改，一个ViewModel可以绑定到不同的View上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。

可重用性，可以把一些视图的逻辑放在ViewModel里面，让很多View重用这段视图逻辑。

独立开发，开发人员可以专注与业务逻辑和数据的开发(ViewModel)。设计人员可以专注于界面(View)的设计。

load和initialize：

load:

1.当类被引用进项目的时候就会执行load函数(在main函数开始执行之前),每个类的load函数只会自动调用一次

2.当父类和子类以及分类都有load方法时，先调用父类的load方法时，后调用子类的load方法，然后调用分类的load方法

3.如果有多个category实现了load方法，那么这些load方法都会执行，但是顺序不一定

initialize：

1.当类初始化时调用，本类和父类的initialize都会调用,且只会调用一次；但是子类如果没有实现initialize方法，父类会调用两次

2.当类有 category 时，category 的 initialize 会执行，而本类的不会执行