> 听说最近女生都喜欢找程序员做男朋友，说是我们程序员一般都稳重，专一，工资高，如果你长得还挺帅那估计就抢手货了。本来我计划今天出去跑步的，但是下雨没去成，因为如果我再减掉15斤体重，那我就具备上面长得帅优秀条件了，估计我离我女神就越来越近了。

上面我减掉15斤就能帅了的话，如果你信了的话，我就感谢下你对我的肯定，因为我自己都不信，我怎么可能那么帅呢。但是有女生说喜欢找我们程序员却不是谣言，因为我听我女友说的，我女友听朋友说的。是不是有点小激动!还没有女友的伙伴动起来，加把劲，代码敲起来，程序跑起来。

正题:

开发中，我们有时会使用runtime进行swizzle方法(如果不知道swzzle请看这里)。多数时候我们会

这样写，swizzle第一种写法:

```
Method originalMethod = class_getInstanceMethod(aClass, originalSel);
Method swizzleMethod = class_getInstanceMethod(aClass, swizzleSel);
method_exchangeImplementations(originalMethod, swizzleMethod);
```

或者是下面这种方式，swizzle第二种写法:

```
Method originalMethod = class_getInstanceMethod(aClass, originalSel);
Method swizzleMethod = class_getInstanceMethod(aClass, swizzleSel);
BOOL didAddMethod = class_addMethod(aClass, originalSel,
method_getImplementation(swizzleMethod), method_getTypeEncoding(swizzleMethod));
if (didAddMethod) {
class_replaceMethod(aClass, swizzleSel, method_getImplementation(originalMethod),
method_getTypeEncoding(originalMethod));
}else{
method_exchangeImplementations(originalMethod, swizzleMethod);
}
```

网上资料查找时发现第二种方式网上有称呼为最佳的swizzle方式，具体可见资料(最佳实践相关资料).
上面两种方式具体哪种好呢，我下不定论，因为在使用过程中，发现他们各自都有自己的问题(将在下面指出)，具体还得看自己需求觉得哪种好，下面将解释这两种方式的一些问题，本文代码见代码地址。
先说说第一种情况(当swizzle一个自身没有实现而父类实现了的方法时)下两种方式比较：
首先我们建立两个类：father类以及继承fahter类的son类

```
@interface Father : NSObject
@end
@implementation Father
-(void)work{
    NSLog(@"父亲得赚钱养家😯");
}
@end

@interface Son : Father

@end

@implementation Son
```

```
@end
```

然后新建Son分类：

```
@implementation Son (Swizzle)
BOOL simple_Swizzle(Class aClass, SEL originalSel,SEL swizzleSel){

    Method originalMethod = class_getInstanceMethod(aClass, originalSel);
    Method swizzleMethod = class_getInstanceMethod(aClass, swizzleSel);
    method_exchangeImplementations(originalMethod, swizzleMethod);

    return YES;
}
BOOL best_Swizzle(Class aClass, SEL originalSel,SEL swizzleSel){

    Method originalMethod = class_getInstanceMethod(aClass, originalSel);
    Method swizzleMethod = class_getInstanceMethod(aClass, swizzleSel);
    BOOL didAddMethod = class_addMethod(aClass, originalSel, method_getImplementati
on(swizzleMethod), method_getTypeEncoding(swizzleMethod));
    if (didAddMethod) {
        class_replaceMethod(aClass, swizzleSel, method_getImplementation(originalMe
thod), method_getTypeEncoding(originalMethod));
    }else{
        method_exchangeImplementations(originalMethod, swizzleMethod);
    }

    return YES;
}
@end
```

这里我们将第一种方式称为simple_Swizzle，第二种方式为best_Swizzle。
情况一：子类里swizzle一个自身没有实现而父类实现了的方法。
我们在分类中添加一个自己的work方法准备swizzle

```
@implementation Son (Swizzle)
-(void)son_work{
    [self son_work];
    NSLog(@"son分类里的son_work");
}

@end
```

然后再load方法中进行替换,我们先使用simple_Swizzle。
```
```

+(void)load{

```
static dispatch_once_t onceToken;
dispatch_once(&onceToken, ^{
         simple_Swizzle([self class], NSSelectorFromString(@"work"), @selector(son_
work));
};
```

}
```

嗯，好了，运行程序后,调用如下代码

```
    Son *son = [Son new];
    [son work];
```

控制台打印如下:

```
2017-06-18 16:11:20.443302 runtime学习[45823:16276777] 父亲得赚钱养家😯
2017-06-18 16:11:20.448797 runtime学习[45823:16276777] son分类里的son_work
```

也许这里你已经发现问题，如果没有我们继续，再调用下面代码:

```
    Father *aFather = [Father new];
    [aFather work];
```

程序崩溃了，控制台输出:

```
[Father son_work]: unrecognized selector sent to instance 0x1700162f0'
```

what?(黑人问号？)，什么情况！！[aFather work]里调用了son_work。
原来，刚刚的simple_Swizzle将子类的son_work的IMP替换给类父类的work方法，导致父类方法work里有[self son_work];而父类并没有这样的方法所以over了。
具体为什么会替换掉父类的方法，是因为class_getInstanceMethod方法在找方法时会从自己类到父类到根类一直查找下去，一直到根类为止，所以上面在查找work方法时就找到father类里了。这就是simple_Swizzle的一个问题。
现在我们使用simple_Swizzle来进行替换。
```

```
best_Swizzle([self class], NSSelectorFromString(@"work"), @selector(son_work));
```

然后

```
    Son *son = [Son new];
    [son work];
```

控制台打印如下:

```
2017-06-18 16:27:43.491944 runtime学习[45833:16282207] 父亲得赚钱养家😯
2017-06-18 16:27:43.493439 runtime学习[45833:16282207] son分类里的son_work
```

没问题，接着:

```
    Father *aFather = [Father new];
    [aFather work];
```

控制台打印如下:

```
2017-06-18 16:30:39.622756 runtime学习[45836:16282868] 父亲得赚钱养家😯
```

可以看到这时父类没有调用son_work。因为best_Swizzle在进行swizzle时会先尝试给自己添加work方法方法实现

```
    BOOL didAddMethod = class_addMethod(aClass, originalSel, method_getImplementation(swizzleMethod), method_getTypeEncoding(swizzleMethod));
```

如果son类没有实现work方式时，class_addMethod就会成功添加一个新的属于son自己的work方法，同时将本来要swizzle的方法的实现直接复制进work里,然后再将父类的IMP给swizzle。如果son已经已经实现了则会添加失败，直接进行swizzle具体可见资料(),逻辑为下面代码。

```
    if (didAddMethod) {
        class_replaceMethod(aClass, swizzleSel, method_getImplementation(originalMethod), method_getTypeEncoding(originalMethod));
    }else{
        method_exchangeImplementations(originalMethod, swizzleMethod);
    }
```

这时simple_Swizzle会有问题，best_Swizzle没有问题。
下面我们来说另外一种情况(swizzle一个子类到父类到根类都没有实现过的方法):
我们给son分类添加一个son_Cry方法:

```
-(void)son_Cry{
    NSLog(@"我是son分类,WZ_Cry方法");
    [self son_Cry];
}
```

同时声明一个没有实现的方法cry;

```
@interface Son : Father

-(void)cry;
@end
```

然后先用simple_Swizzle替换:

```
simple_Swizzle([self class], @selector(cry), @selector(son_Cry));
```

接着调用cry方法:

```
Son *son = [Son new];
[son cry];
```

控制台打印:

```
[Son cry]: unrecognized selector sent to instance 0x170018900'
```

嗯，调用失败，正常因为都没有实现如何swizzle。然后用我们用回自动添加一个实现的best_Swizzle方式，控制台打印如下:

```
2017-06-18 16:59:33.815223 runtime学习[45845:16289311] 我是son分类,WZ_Cry方法
2017-06-18 16:59:33.815243 runtime学习[45845:16289311] 我是son分类,WZ_Cry方法
2017-06-18 16:59:33.815319 runtime学习[45845:16289311] 我是son分类,WZ_Cry方法
...此处省略无数次上面的log打印
```

再一次（what? 黑人问号❓）,程序死循环了...
问题出在这两行代码:

```
//因为cry方法没有实现过，class_getInstanceMethod无法找到该方法，所以originalMethod为nil。
 Method originalMethod = class_getInstanceMethod(aClass, originalSel);

 //当originalMethod为nil时这里class_replaceMethod将不做替换，所以swizzleSel方法里的实现还是自己原来的实现。
    class_replaceMethod(aClass, swizzleSel, method_getImplementation(originalMethod), method_getTypeEncoding(originalMethod));
```

所以最终导致在下面方法里重复调用自己。

```
-(void)son_Cry{
NSLog(@"我是son分类,WZ_Cry方法");
[self son_Cry];
}
```

这里就是best_Swizzle的一个问题，当然这个问题很少出现，因为我们一般swizzle时都是swizzle一个已知的方法，所以一般都有方法实现。
但是也不是不会出现这种问题，这里我在一次进行swizzle一个类实现的协议方法时出现了，比如AppDelegate类里的协议，我想在某个代理回调里插入一段自己的逻辑，又不想对已有的项目里直接加入，我想它时可以直接拿到另外一个项目也可以直接用的，所以我建了一个分类swizzle协议方法，如果我直接swizzle一个没有实现过的协议方法，就会出现死循环。
具体可见demo
里的解决方法是在originalMethod为nil时，替换后将swizzleSel复制一个不做任何事的空实现,代码如下:

```
    if (!originalMethod)
    {
        class_addMethod(aClass, originalSel, method_getImplementation(swizzleMethod), method_getTypeEncoding(swizzleMethod));
        method_setImplementation(swizzleMethod, imp_implementationWithBlock(^(id self, SEL _cmd){ }));
    }
```

喝口水 总结:

对于simple_Swizzle和best_Swizzle的选择没有具体要求，大家结合自己的场景使用，我一般直接用simple_Swizzle，因为我知道这个方法一定有，所以能简单就粗暴吧。

对于runtime，大家使用起来还是的谨慎些，一不小心就会入坑。

如果喜欢点个喜欢吧😍，因为你的喜欢就是你的喜欢，哈哈哈哈。

一些资料

这里有6篇文章，点击吧，Go!Go!GO!