iOS 编码规范

> 2019.1.3
>
> 本文档并不会涉及到代码优化：NSInteger 替代 int，NSArray 类型属性需要用 copy 特性等。
>
> 本文档为纯编码规范，主要是为了统一编码习惯。包含核心原则，命名规范，通用规范，代码组织，注释五个主题。

核心原则

1. 语言，使用 US 英语命名

> 文件名，类，方法，宏，ImageSet，Localizable.string Key，配置，#pragma mark，model 字段）

```
Colour，manzengId，颜色 //不建议(英式英语，拼音，中文)
Color //推荐

#pragma mark - 懒加载 //不建议
#pragma mark - Property Method //推荐
```

2. 代码风格，简单易懂，逻辑清晰

> 清晰永远比简洁更重要。

```
insertObject:at: //不建议
insertObject:atIndex: //推荐

setBGColor: //不建议
setBackgroundColor:  //推荐

[button setTitle:@"title" forState:0]; //不建议
[button setTitle:@"title" forState:UIControlStateNormal]; //推荐

result = a > b ? x = c > d ? c : d : y; //不建议(多个表达式嵌套)
//推荐
msg = code > 0 ? @"Error" : @"Success";
result = reachable > NotReachable ? msg : @"Fail";
```

命名规范

1. 遵从驼峰命名法

- 变量

```
NSUInteger Index = 2; //不建议
NSUInteger index = 2; //推荐
```

- 方法

```
+ (void)ResetStandardUserDefaults; //不建议
+ (void)resetStandardUserDefaults; //推荐
//特殊情况
+(NSURLSessionDataTask *)GET:(NSString *)URLString;
+(instancetype)PDFWithImage:(UIImage *)image;
```

- 类

```
@interface myCouponVC () //不建议(首字母没有大写，且没有前缀)
@interface RGMyCouponViewController () //推荐
```

- 常量

```
static NSString * const CouponCellID = @"CouponCell"; //不建议
static NSString * const kCouponCellID = @"kCouponCellID"; //推荐
```

- 宏

```
//常量宏
#define useLocalhost 0 //不建议
#define kUseLocalHost 0 //推荐

//操作宏
#define NullFilterString(s) //不建议
#define RGNullFilterString(s) //推荐
```

2. ImageSet，以下划线分割模块，依次递进。全小写。

```
//不建议
bg
backImage

//推荐
product_detail_price_background
nav_bar_back
```

3. Localizable.string，以下划线分割模块，依次递进。首字母大写。

```
//不建议
EDIT
tryAgain

//推荐
BuyerShow_MyStyle_Edit
Feedback_Alert_TryAgain
```

4. 前缀

- 前缀以全大写命名。比如 RG、ZF；
- 类、分类、协议、操作宏、枚举使用前缀，但不要为成员变量，方法使用前缀；
- 命名前缀不要与 Cocoa 框架与第三方组件冲突。比如 UI、NS、CG。

6. 不要使用 “and” 和 “width” 来连接参数

```
initWithTitle:withImage:andColor: //不建议

initWithTitle:image:color: //推荐
```

7. 组件名称不建议缩写

```
//不建议
UILabel * nameLbl;
UIViewController * homeVC;
UICollectionViewCell * productCell; //TODO:待定

//推荐
UILabel * nameLabel;
UIViewController * homeViewController;
UICollectionViewCell * productCCell;
```

8. 文件命名

- 如果头文件内只定义了单个类或者协议，直接用类名或者协议名来命名头文件；

```
//RGHomeViewController.h
@interface RGHomeViewController
@end
```

- 如果头文件内定义了一系列的类、协议、类别，使用其中最主要的类名来命名头文件；

```
//NSString.h
@interface NSString
@end
@interface NSMutableString
@end
```

- 分类的命名应该依据其主要的功能。

```
//不建议
UIColor+ZHCategory.h
UIButton+Extension.h

//推荐
UIColor+ZHRandom.h
UIButton+RGPointBound.h
```

通用规范

1. 大括号

- 开始于行尾，结束于行首，包括 `Method`， `if`， `else`， `switch`， `while` 等；

```
//不建议
if (condition)
{
    ...
}
//推荐
if (condition) {
    ...
}
```

- 所有流程控制语句必须加上括号，即使只有一行。

```
//不建议
if (condition) ... else ...
//推荐
if (condition) {
    ...
} else {
    ...
}
```

2. 代码长度

- **一个方法**内部最多 **50** 行，如果超过就需要精简代码，为了方便以后阅读、热修复和代码重构；
- 每行**代码长度**不超过 **80** Column

```
//不建议
[self.contentLabel.text
boundingRectWithSize:CGSizeMake(MAXFLOAT,MAXFLOAT)
options:NSStringDrawingUsesLineFragmentOrigin |
NSStringDrawingUsesFontLeading
attributes:@{NSFontAttributeName:self.contentLabel.font}
context:nil];


//推荐
NSDictionary * attributes =
@{NSFontAttributeName:self.contentLabel.font};
```

```objc
NSStringDrawingOptions options =
NSStringDrawingUsesLineFragmentOrigin |
NSStringDrawingUsesFontLeading;
CGSize size = CGSizeMake(MAXFLOAT,MAXFLOAT);
//冒号对齐
[self.contentLabel.text boundingRectWithSize:size
                                     options:options
                                  attributes:attributes
                                     context:nil];
//如果方法名过长，推荐使用 Tab 缩进
[self shorts:@"shorts"
        middlelongKeywords:@"more"
       eventLongerKeywords:@"event"];


//不建议，Block 无需冒号对齐
[UIView animateWithDuration:1.0
                 animations:^{
                     //Do something
                 }
                 completion:^(BOOL finished) {
                     //Do something
                 }];
//推荐
[UIView animateWithDuration:1.0 animations:^{
    //Do something
} completion:^(BOOL finished) {
    //Do something
}];



//推荐
NSArray *array = @[
                   @"This",
                   @"is",
                   @"an",
                   @"array"
                   ];
```

> Xcode -> Preferences -> Text Editing -> Page guide at column: **80**

3. 语句中留有合适的空格

- 类方法和实例方法的 “+” 和 “-” 需要留一个空格

```
//不建议
+(void)resetStandardUserDefaults;
//推荐
+ (void)resetStandardUserDefaults;
```

- 运算符之间留有空格

```
//不建议
NSString * title=self.isViewLoaded?@"title":@"loading";
//推荐
NSString *title = self.isViewLoaded ? @"title" : @"loading";
```

- 方法调用之间留有空格

```
//不建议
[[NSUserDefaults standardUserDefaults]objectForKey:@"name"];
//推荐
[[NSUserDefaults standardUserDefaults] objectForKey:@"name"];
```

- NSDictionary，key : value 之间留有空格

```
//不建议
@{NSForegroundColorAttributeName:fontColor}

//推荐(冒号之间留有空格)
@{NSForegroundColorAttributeName : fontColor}
```

4. @Class

```
//不建议
@class UIView, UINavigationItem, UIBarButtonItem, UITabBarItem,
UISearchDisplayController;

//推荐
@class UIView;
@class UINavigationItem, UIBarButtonItem, UITabBarItem;
@class UISearchDisplayController;
```

5. @property

- 保持属性的特性的顺序一致

```
//不建议
@property(readonly, copy, nullable, nonatomic) NSString *nibName;

//推荐(保证特性关键字顺序一致)
@property(nullable, nonatomic, copy, readonly) NSString *nibName;
```

- 控制属性的可访问性

```
//.h 不建议
@property(nonatomic, strong) UILabel * titleLabel;

//推荐
//.h
@property(nonatomic, strong, readonly) UILabel * titleLabel;
//.m
@property(nonatomic, strong) UILabel * titleLabel;
```

6. 变量

- 尽量用变量取代属性

```
//TODO:待定
@interface ViewController (){
    UILabel * nameLabel;
}
@property (nonatomic, strong) UIView * nameLabel;
@end
```

- 变量定义的位置

```
//TODO:待定
@interface ViewController (){
    UILabel * nameLabel;
}
@end

@implementation ViewController{
    UILabel * nameLabel;
}
@end
```

7. 布尔值

- OC 使用 YES 和 NO
- nil 会解析成 NO

```
//不建议
if (obj == nil) {}
//推荐
if (obj) {}
```

- 不要拿变量跟 YES 和 NO 比较

```
//不建议
if (isAwesome == YES) {}
//推荐
if (isAwesome) {}
```

8. 枚举，`NS_ENUM` 不要全部写在一个文件里面，定义在各自的模块里面。

9. #import 的顺序

```
//模板
#import <系统库>
#import <第三方库>
#import "其它类"

//不建议
#import "RGModel.h"
#import <UIKit/UIKit.h>
#import <Google/Analytics.h>

//推荐
```

```
#import <UIKit/UIKit.h>
#import <Google/Analytics.h>
#import "RGModel.h"
```

10. If 嵌套

```
//不建议
if ([obj boolValue]){
    if (valid){
        //Do something
    }
}

//推荐
if (![obj boolValue]){
    return;
}
if (!valid){
    return;
}
//Do something
```

代码组织

1. 生命周期

```
#pragma mark - Life Cycle
- (instancetype)init;
- (void)dealloc {}
- (void)viewDidLoad {}
- (void)viewWillAppear:(BOOL)animated {}
- (void)viewDidLayoutSubviews {}
- (void)didReceiveMemoryWarning {}
```

2. 协议实现

```
#pragma mark - NSCopying
+ (id)copyWithZone:(struct _NSZone *)zone

#pragma mark - NSObject
- (BOOL)isEqual:(id)object;

#pragma mark - UITableViewDelegate
#pragma mark - WKNavigationDelegate
#pragma mark - YYModel
```

3. 事件处理（UIContol Action，Notification，KVO，UIGestureRecognizer）

```
#pragma mark - UIControl Action
-(void)submitAction:(id)sender {}

#pragma mark - Notification Action
- (void)userSignOutAction:(NSNotification *)notification {}

#pragma mark - Touch Action
- (void)touchesBegan:(NSSet<UITouch *> *)touches withEvent:(UIEvent
*)event {}
```

4. 私有方法，#pragma mark - Private Method

5. 公共方法，#pragma mark - Public Method

6. 属性方法

```
#pragma mark - Property Method
- (UILabel *)centerLabel {
    if (!_centerLabel) {
        _centerLabel = [UILabel new];
    }
    return _centerLabel;
}
```

注释

1. h 文件建议所有的公共属性，方法，变量都必须加上注释；

2. `Protocol` 必须加上注释；

3. 属性或变量有默认值，必须要注明；

```
//推荐
/** The backdrop for your app’s user interface and the object that
dispatches events to your views. */
@property (strong, nonatomic) UIWindow *window;
```

4. Block 和 方法注释必须注明其主要用途，参数和返回值必须也要注明。

```
//不建议
//Note that NSArray objects are not like C arrays...

//推荐
/**
Note that NSArray objects are not like C arrays...

 @param anObject <#anObject description#>
 @param index <#index description#>
 @return <#return value description#>
 */
- (NSUInteger)insertObject:(ObjectType)anObject atIndex:
(NSUInteger)index;
```

> 快捷键：alt + command + / 自动生成上述注释模板

5. 其它建议注释方式

- 标记未完成的功能

```
//TODO: Test Data
```

- 标记追踪问题的位置

```
//FIXME: Bug 1101
```