IOS 代码规范

代码规范说明书

佳都数据

二〇一九年二月

文档版本

序号	创建人	创建时间	当前版本	备注
1	文伟佳，吴丰收	2019-02-14	1.0.0	
2	修改人	修改时间	当前版本	修改内容

目 录

1. 核心原则

1.1. 代码简洁易懂，逻辑清晰

- 不要过分追求技巧，降低程序的可读性。
- 简洁的代码可以让 bug 无处藏身。要写出明显没有 bug 的代码，而不是没有明显 bug 的代码。

1.2. 先保证程序的正确性，其次考虑效率

编程优先考虑满足正确性，再考虑扩展问题，可重用问题，测试问题

1.3. 面向变化编程，而不是面向需求编程

需求是暂时的，只有变化才是永恒的。 本次迭代不能仅仅为了当前的需求，写出扩展性强，易修改的程序才是负责任的做法，对自己负责，对公司负责。

2. 通用规范

2.1. 运算符

2.1.1. 一元运算符与变量之间不需要空格

例子：

++iCount

!blaue

2.1.2. 二元运算符与变量之间需要空格

例子：
for(int i = 0; i < 5; i++)

2.2. 变量

2.2.1. 变量使用前应初始化

例子：

int a = 0;

2.2.2. 局部变量尽量命令在使用范围

例子：

int a = 0;

//a do something

int b = 0;

//b do something

2.2.3. 常量使用

常量是容易重复被使用和无需通过查找和代替就能快速修改值。常量应该使用 static 来声明而不是使用#define，除非显式地使用宏。

应该:

static NSString * const JPushKey = @"066fcjsdagjdadalskdad9288";

static CGFloat const PCIItemSpace = 50.0;

不应该:

#define JPushKey @"066fcjsdagjdadalskdad9288"

#define PCIItemSpace 50

2.3. IF 语句

2.3.1. 必须列出所有分支

例子：

```
int a = 0;

if (a < 5) {

    a = 2;

}else{

    a = 3;

}
```

2.3.2. 要善于使用 return 提前结束分支

例子：

```
- (void)someThing:(NSInteger)a

{

    if (a < 0) {

        return;

    }

}
```

2.3.3. 变量在右，常量在左

例子：

```
if (0 == a) {

    return;
```

```
}
```

2.4. 函数

2.4.1. 一个函数的长度必须限制 50 行

通常情况下，在不翻页或者滚动的情况下看完一个函数，有利于理解这个函数，更方便阅读

2.4.2. 一个函数要符合单一原则

一个函数里面与第二个函数无直接关系，最好不要出现有第二个函数在里面调用

2.4.3. 对输入参数进行有效性判断，参数错误立即返回

例子：

```
- (void)someThing:(NSInteger)a
{
    if (a <= 0) {
        return;
    }
    //a do something
}
```

2.5. 注释

2.5.1. 公共接口注释

注释需告诉当前类所实现的功能

2.5.2. 深层专业知识代码

注释体验出实现代理和思想

2.5.3. 容易产生歧义的代码

严格来说这种是不允许的，若出现这种情况需注释说明一下情况

2.5.4. 注释方式

多行使用 “/**/”，单行使用“//…”，注释语言简洁，准确，易懂

例子如下：

```
/**
 初始化

 @param data          数据
 @param identifiers 复用

 @return 实例

 */
```

2.5.5. #Paragma Mark、#warning 的使用

#pragma mark –标记能快速定位查找相关内容

#warking 有助于记忆需要注意的问题

3. IOS 规范

3.1. 变量

3.1.1. 变量名需使用驼峰命名格式

例子：

```
NSString *s = @"";
```

```
PCIHomeVC.h
```

3.1.2. 变量的名称需同时包含名称和功能

例子：

```
UIButton * addBtn;
```

3.1.3. 系统常用类加入后缀

例子：

UIViewController	VC
UIView	View
UILabel	Label
UIButton	Btn
UIImage	Img
UIImageView	ImageView
NSString	Str
NSArray	Array
NSDictionary	Dict
NSet	Set

3.2. Block

3.2.1. Block 简单命名

例子：（使用 copy 修饰）

```
typedef void(^closeBtnClick)(void);

@property(nonatomic,copy)closeBtnClick closeBtnClickBlock;
```

3.2.2. Block 循环引用问题

1. Block注意防止循环引用，导致实例无法释放。

定义一个weakSelf，然后在block体里面使用该weakSelf就可以避免循环引用的问题

2.防止因为弱引用导致Block内部引用变量时，变量被提前被释放。

block内部使用strong，然后执行的时候判断下self是否还在，如果在就继续执行下面的操作，否则return或抛出异常.

例子：

```
WeakSelf();
[_bottomView setSubmitBtnClickBlock:^{
    StrongSelf();
    if ([strongself checkAvailability]) {
        [strongself submitPictureToSFTP];
     }
}];
```

3.3. Delegate

3.3.1. Delegate 简单命名

例子：(使用weak修饰，以Delegate为后缀)

```
@property(nonatomic,weak)id <TGSignalPickDelegate> delegate;
```

3.3.2. 修饰代理方法

使用 optional 修饰可不实现方法，使用 required 修饰为必须实现方法，同时回调方法两个参数需以类名字为开头，已区分此方法属于那类，例子：

```
- (NSInteger)tableView:(UITableView *)tableView

numberOfRowsInSection:(NSInteger)section {

}

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView {

}
```

3.4. 枚举

3.4.1. 枚举的命名

Enum 中枚举的内容的命名需要已该类类型名称开头

例子:

```
typedef enum : NSUInteger {

    ClockWiFiNone,

    ClockWiFiCompany,
```

```
    ClockWiFiOther,

} ClockWiFi;
```

推荐使用下面这种

```
typedef NS_ENUM (NSUInteger, YYTextTruncationType) {
    YYTextTruncationTypeNone   = 0,
    YYTextTruncationTypeStart  = 1,
    YYTextTruncationTypeEnd    = 2,
    YYTextTruncationTypeMiddle = 3,
};
```

3.5. 宏

3.5.1. 宏的命名需要使用大写，单词用_分隔

例子：

```
#define INPUT_GES_PWD  @"输入手势密码"
```

3.6. 属性

3.6.1. 属性命名使用小驼峰

例子：

```
@property (nonatomic,strong) NSArray *sectionTitle;        //分区头
```

3.6.2. BOOL 需加上 getter 修饰

例子：

```
@property(nonatomic,assign,getter=isSwitchGesture) BOOL switchGesture;
```

3.6.3. 使用懒加载实例 getter 方法

```
-(NSArray *)sectionTitle{

    if (!_sectionTitle) {

        _sectionTitle = @[@"",@"",@"系统通知"];

    }

    return _sectionTitle;

}
```

懒加载调用使用点语法来读取数据。

3.6.4. 尽量使用不可变对象

对于公开文件，例子如下：

```
@property(nonatomic,copy,readonly) NSString *userGesturePWDKey;
```

对于内部文件 例子如下：

```
@property(nonatomic,copy,readWrite) NSString *userGesturePWDKey;
```

作用：外部只能读取该数据而不能修改它

3.7. 函数(方法)

3.7.1. 函数命名使用小写开头

例子：

```
- (void)someThing:(NSInteger)a
{
    //do something
}
```

3.7.2. 方法名前缀需特出作用

例子：（刷新）

```
- (void)refreshData:(NSData*)data
{
    //do something
}
```

3.7.3. 方法参数过长需换行处理，同时不建议用 and 连接

例子：

```
- (void)doSomethingWith:(NSString *)theFoo
                   rect:(CGRect)theRect
               interval:(CGFloat)theInterval
{
    //do shomething
```

```
}
```

注：若参数过多，不妨把参数定位为一个 model

3.8. 类

3.8.1. init 初始化方法

例子 ：

```
-(instancetype)initWithType:(PCIVerificationCodeType)type{

    self = [super init];
    if (self) {
        _type = type;
    }
    return self;
}
```

3.8.2. 类的头文件尽量少引用其他头文件，可使用 class 代替

例子：

```
@class PCISetListModel;

@interface PCISetCell : UITableViewCell
```

.m文件使用

```
#import "PCISetListModel.h"
```

作用：减少编译时间，避免循环引用

3.8.3. 类的命名

团队合作，尽量在命名是加入前缀，针对工程使用工程的缩写

例子：

```
@interface PCIHomeVC : PCIBaseCollectionviewVC
```

3.9. 单例

3.9.1. 使用 dispatch_once 来生成

例子：

```
+ (instancetype)Config {

    static dispatch_once_t onceToken;

    dispatch_once(&onceToken, ^{

        enviromnet = [[JDEnvironment alloc] initEnv:CURENV];

    });

    return enviromnet;

}
```

3.10. 防止空值导致闪退

3.10.1. 字典空值闪退问题

不建议方式：

```
[parameters setObject:orderTime forKey:@"txnTime"];
```

建议方式

```
parameters[@"texTime"] = orderTime;
```

3.10.2. 防止空值措施

先用宏判断是否空值，例子如下：

```
#define NotNilAndNull(_ref)  (((_ref) != nil) && (![(_ref) isEqual:[NSNull
null]]))
#define IsNilOrNull(_ref)   (((_ref) == nil) || ([(_ref) isEqual:[NSNull null]]))
```

3.11. 代码结构

3.11.1. 工程在 viewController 常用结构

例子：

```
@implementation WWJDemoViewController

#pragma mark - Lifecycle ViewController生命周期
- (instancetype)initWithFoo:(Foo *)foo;
- (void)dealloc;

#pragma mark - View Lifecycle view 页面 生命周期
- (void)viewWillAppear:(BOOL)animated;
- (void)viewDidLoad;
```

```
#pragma mark - prepareData 准备数据

- (void)prepareData;

#pragma mark - Layout UI准备和布局相关

- (void)prepareUI;

- (void)makeViewConstraints;

#pragma mark - Public Interface 公共接口

- (void)startFooing;

- (void)stopFooing;

#pragma mark - User Interaction 用户交互 按钮事件相关

- (void)foobarButtonTapped;

#pragma mark - XYZFoobarDelegate Delegate代理

- (void)foobar:(Foobar *)foobar didSomethingWithFoo:(Foo *)foo;

#pragma mark - Internal Helpers 内部公用方法

- (NSString *)displayNameForFoo:(Foo *)foo;

#pragma mark - Lazy 懒加载

- (NSArray *)dataArray;

@end
```

3.12. 工程布局

3.12.1. 使用方式

团队项目建议使用纯代码或者 XIB

3.12.2. 项目文件目录结构

自己负责的模块 M—V—C 集中放到模块文件夹的 Model View Controller 中

项目最好对文件夹进行架构命名：例子如下：

Class,Common,Category,baseClass,Marcos,Network,ThirdParty,Utils 等文件夹进行功能分类(可参考云匙开发架构)

3.13. 版本规范

采用 ABC 三位数字命名，如 1.0.1，有版本更新参照如下例子：

版本号	说明	示例
A.b.c	属于重大更新内容	**1**.0.1 -> **2**.00
a.**B**.c	属于小部分更新内容	1.**2**.1 -> 1.**2**.2
a.b.**C**	属于补丁更新内容	1.0.**1** -> 1.0.**2**