iOS编码规范

# 1、目的

统一规范XCode编辑环境下iOS开发的编码风格和标准

# 2、适用范围

适用于所有用Objective-C语言开发的项目。

Objective-C编码约定

该约定大致内容参照自NYTimes、Apple、Google等编码风格约定，作为团队内  
 开发使用，   
 不定期进行更新

[NYTimes Objective-C Style Guide](https://github.com/NYTimes/objective-c-style-guide" \t "http://note.youdao.com/share/_blank)

[NYTimes Objective-C Style Guide 中文](https://github.com/NYTimes/objective-c-style-guide/blob/master/README_zh-Hans.md" \l "%E5%8F%98%E9%87%8F" \t "http://note.youdao.com/share/_blank)

[The official raywenderlich.com Objective-C style guide](https://github.com/raywenderlich/objective-c-style-guide" \t "http://note.youdao.com/share/_blank)、

# 3、编码规范

## 3.1、文件

1) 项目文件必须使用一个有意义的名字且前缀以QS\_。例如：Xcode中下拉刷新的  
 项目文件被命名为’QS\_ PullDownRefresh.xcodeproj’。

2) 公共头文件统一命名为’Public.h’。任何文件的命名尽量不要以中文命名

3) 对于文件的目录要按如下结构创建：

-Document（文档所在路径）

Help（帮助文件所在路径）

-图片等资源文件放在单独的目录与组(Group)中，如Images。

-所有的AppDelegate文件、Main.storyboard、Assets等统一放在一个Base组中，  
 包括公共头文件、cofig配置文件等

-项目中每一个主页面放在单独的组(Group)中，组名允许使用中文，每一个主  
 页面的view和viewController放在此组下。

-所有的view放在单独的组(Group)中，如Custom View。

-所有的viewController放在单独的组中，如viewControllers。

-引用外部文件或者某个单独的功能时，放在单独的组中，

例如：

程序中使用了coverflow功能，引用openflow的文件时，将其全部文件放在openFlow这个组中。其view和viewcontroller等文件的组织不受上面所规定的影响。

## 3.2、注释

1) 在一些需特别强调或代理方法等（.m文件中），方便直接在快速搜索中定位方法位  
 置的，使用mark 标记，且统一前面加-，如：#pragma mark – 标记

2) 注释可以采用’ /\* \*/ ’和’ // ’两种注释符号，涉及到多行注释时，尽量使用：’ /\* \*/ ’。

3) 对于一行代码的注释可放在前一行及本行上，不允许放在下一行，更不允许在一行

语句的中间加入注释。

4) 单元文件的文件头注释说明应按如下格式：

//

// 文件名

// 工程名

//

// Created by 创建者 on 日期.

// Copyright 2010 xxx有限公司. All rights reserved.

//

// 系统名称：

// 功能描述：

// 修改记录：(仅记录功能修改)

// 张三 2012-02-02 创建该单元

// 小明 2010-03-02 增加本地点单功能。

//

5) 对于需重点说明的方法，方法前面的注释遵循以下格式：如果某项没有，则以N/A 表示

例如：

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称：-(BOOL)showFiveAndSixStairRoomCountByStatu

函数描述：显示特定状态下五楼和六楼的房间数

输入参数：(NSString \*)statu：某状态。

输出参数：(int \*)roomCount ：该状态房间数量。

返回值：BOOL：操作是否成功。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

(void)ShowFiveAndSixStairRoomCountByStatu:(NSString \*)  
staturoomCount:(int \*)\_roomCount;

或

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

函数名称：-(NSUInteger)showFiveAndSixStairRoomCount

函数描述：显示五楼六楼的房间数。

输入参数：N/A

输出参数：N/A

返回值：NSUInteger :房间数量。

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

(NSUInteger)showFiveAndSixStairRoomCount

6) 不必每行都加注释，在3~10行左右的段落做注释要好于每行都做注释，显而易见的代码不加注释。

例如：

If (!returnValue) //调用登录过程失败 🡨无用的注释

{

NSLog(@”登录失败”);

}

## 3.3、编码排版格式

1) 代码的缩进应使用空格（SPACE），不能使用制表符（TAB），并且缩进以2 个字符为单位。

2) 中括弧的每一个括弧在源程序中要单独占一行。

例如

//不正确用法

for (int i = 0; i < 10 ; i++){

}

//正确用法

for (int i = 0; i < 10; i++)

{

……

}

3) 空格的使用

a) 关键字与其后的表达式之间要有空格，如：

if (expr)

或

for (expr)

b) 单目操作符不应与它们的操作数分开（如’!’和’^’等）。

c) 除’ , ’外，其它双目操作符应与它们的操作数用空格隔开，特别是=号两边。

例如

i=i+1; //错误的写法，操作符两端没有空格

i = i + 1; //正确的写法，

if(a>b) //错误的写法，逻辑判断符号两端没有空格

if(a > b) //正确的写法

d) .h中协议<>前面有一个空格。

e) .h中成员声明时，类型与变量之间有至少1个空格。\*号靠近变量，  
 不靠近类型。

f) @property后留1个空格，（）里面，逗号紧跟前一变量，与后一变量  
 之间留1个空格。（）外面，先留1个空格，再声明属性。

g) 方法的+、-后面与（）返回值之间留1个空格。

h) 返回类型与\*之间留1个空格，方法参数中返回类型与\*之间留1个空格。

i) 在多参数方法中，每个参数后面都有1个空格。

4) 每行只能有一个语句

例如

//不正确写法

NSUInteger objectIndex, stuffCount;

或

objectIndex = objectIndex + 10, stuffCount = stuffCount + 20;

或

@synthesize MyView, MyLabelView;

//正确写法

NSUInteger objectIndex;

NSUInteger stuffCount;

或

objectIndex = objectIndex + 10;

stuffCount = stuffCount + 20;

或

@synthesize MyView;

@synthesize MyLabelView;

5) 关于空行

a) .h中的空行

、文件说明与头文件包含(#import)之间空1行

、头文件包含(#import)之间，如果需要分类区别，各类别之间空1行。

、头文件包含(#import)与@class之间空2行。

、@interface与@class之间空1行。

、头文件{}里面，空1行开始声明对象成员，如果需要分类区别，各类别之间空1行。

、头文件{}外，空1行书写属性，如果需要分类区别，各类别之间空1行。

、属性下面空1行开始写方法，如果需要分类区别，各类别之间空1行。

、方法完成后，空1行@end。

、如果需要声明protocol，空2行接着写。通常protocol写在@end后面，但是声明在@interface之前。

b) .m中的空行

、文件说明与头文件包含(#import)之间空1行

、头文件包含(#import)之间，如果需要分类区别，各类别之间空1行。

、@implementation和@synthesize之间空1行, 如果需要分类区别，各类别之间空1行。

、@synthesize与方法之间空1行。

、方法与方法之间空1行。

C) 方法里面的空行

、变量声明后需要空1行，如果需要分类区别，各类别之间空1行。

、条件、循环，选择语句，整个语句结束，需要空1行。

、各功能快之间空1行。

、最后一个括弧之前不空行。

、注释与代码之间不空行。

、#pragma mark 与方法之间空1行。

d) 每行代码最多不得操作100个字。  
设置如下：Xcode => Preferences => TextEditing => Page Guide at column   
/输入 100即可。

## **3.4、命名规范**

### 3.4.1、保留字

Objective-c语言的保留字或关键词应全部使用小写字母，除下表中保留字外， private、protected、public、在类型说明中也作为保留字使用。还有 nonatomanic,retain,readwrite,

readonly等也有特殊的使用场合。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_Bool | \_Complex | \_Imaginary | auto | break |
| bycopy | byref | case | char | const |
| continue | default | do | double | else |
| enum | extern | float | for | goto |
| if | in | inline | inout | int |
| long | oneway | out | register | restrict |
| return | self | short | signed | sizeof |
| static | struct | super | switch | typedef |
| union | unsigned | void | volatile | while |

### 3.4.2、方法

1) 方法的名称应全部使用有意义的单词组成，（原则是看见方法名即知方法用处）且以小写字母开头，多单词组合时，后面的单词首字母大写。对于重要需重点说明的，按照注释规则注释。

例如：- (void)getUserInformation……

2) 设置类变量的内容的方法应使用set作为前缀，读取变量的内容的方法应使用get  
 作为前缀。

例如：

- (void)getUserName;

- (void)setUserName:(NSString \*)userName;

3) 方法中的参数：第一个参数名称要从函数名称上携带出来，第二个参数的首字母大 写，且以and开头多个单词组合时，后面单词首字母大写。

例如：

- (void)myFunctionWithSizeA:(CGSize)sizeA andSizeB:(CGSize)sizeB;

### 3.4.3、变量

1) 变量必须起有意义的名字，使其他组员可以很容易读懂变量所代表的意义，变量命名必须是英文，可以采用同义的英文命名，可使用几个英文单词，第一个单词首字母小写，其他单词首字母大写。

例如：

NSString \*username;

2) 对于一些特殊类型的变量，命名时要带上类型，如NSArray 的变量命名为xxxArray，其他的如xxxDictionary，xxxSize等。这样就可以从名称上知道是什么类型的变量。变量类型可以简写，但不能不写，如xxxArr,xxxLab,xxxBtn,xxxDict

3) 对于要和interface builder关联的的输出口变量，命名时要后缀以特定的控件名。

例如：

IBOutlet UILabel \*userNameLabel;

4) 对于使用c语言形式声明的变量，一些特定类型可采用一定的简写：

例如：

指针类型：P

结构体类型：Rec

数组类型：Arr

Core Graphic：CG等。

循环控制变量通常使用单一的字符如：i、j、k等。使用有意义的名字，如 objectIndex也是可以的。

5) 尽量避免使用全局变量，如果必须使用全局变量则必须加前缀 ‘ Pub\_’,同时应在  
 变量名称中体现变量的类型。如Pub\_index\_Integer

6) 私有实例变量前加一个下划线，如\_myPrivateVarible。

7) 枚举变量也要有相应的前缀来区分不同的enum变量。

例如：

typedef NS\_ENUM(NSInteger, UIViewAnimationTransition) {

UIViewAnimationTransitionNone,

UIViewAnimationTransitionFlipFromLeft,

UIViewAnimationTransitionFlipFromRight,

UIViewAnimationTransitionCurlUp,

UIViewAnimationTransitionCurlDown,

};

枚举类型命名要加相关类名前缀并且枚举值命名要加枚举类型前缀。

3.4.4、常量

对于常量的命名，包括常用变量与宏定义变量等，统一在常量前面加上小写字母k  
 作为标记. 如:

static const NSTimeInterval kAnimationDuration = 0.3;

定义作为NSDictionary或者Notification等的Key值字符串时，必须加上const  
 关键字, 以防止被修改. 如:

NSString \*const UIApplicationDidEnterBackgroundNotification

1) 若常量作用域超出编译单元（实现文件）, 需要在类外可见时, 使用extern  
 关键字, 并加上该类名作为前缀。

如 extern NSString \*const kThumbnailSize

2) 全局常量（通知或者关键字等）必须用const来定义. 因为如果使用宏定义,   
 一来宏可能被重定义. 二来引用不同的文件可能会导致宏的不同。

3) 避免在程序中直接出现常数，使用超过一次的应以const的形式来替代，不建议  
 使用宏定义。

4) 常数的定义应与它实际使用时的类型相一致。如以3.0来定义浮点类型，用3  
 表示整型。

5) 一些常量前加特殊前缀，可以作为不同常量的区分，全部大写字母，

例如：

UserDefaultsKey的变量前加UDKEY\_,

NotificationNameKey前面加NNKEY\_,

DictionaryKey前面加DICTKEY\_,

### 3.4.5、类

1) 所有的类名，接口名(Protocol)均以大写字母开头，多单词组合时，后面的单词首字母大写。类，接口名必须是有意义的。

2) 继承自UIView的类以View结尾。

例如：

QS\_OperatorUsersInfomationView，QS\_LabelView等。

3) 继承自ViewController的类以viewController或VC结尾。

例如：

QS\_HomePageViewController，QS\_LoginViewController，QS\_LogoutVC等。

4) 所有保存数据的实体以Model结尾。

例如：

QS\_UserModel

5）对于公共的、被其他界面复用的类，在类名后加上后缀\_COM,如QS\_LogoutVC\_COM

## 3.5、修改规范

在项目完成后的维护阶段，代码若有修改，均需按照修改规范操作

### 3.5.1、新增代码行

新增代码行的前后应有注释行说明。

//修改人，修改时间，修改说明

新增代码行

//修改结束

### 3.5.2、删除代码行

删除代码行的前后用注释行说明

//修改人，修改时间，修改说明

要删除的代码行(将要删除的语句进行注释)

//修改结束

### 3.5.3、修改代码行

修改代码行以注释旧代码行后再新增代码行的方式进行。

//修改人，修改时间，修改说明

//修改前代码

//修改前代码(将要修改的代码进行注释)

//修改后代码行开始

修改后代码行

//修改结束