**iOS开发规范文档**

命名原则

1.基本原则

（1）清晰。 命名应该是以清晰为主、简洁为辅。总的来讲不要使用单词的简写，除了使用非常常见的简写以外，尽量使用单词的全称。不可使用拼音、数字、容易让人看不懂易混淆的词命名。自定义API（造轮子）时，API的名称不要有歧义并且不要与苹果原生API产生冲突，让使用者一看你的API就知道是以什么方式用来做什么的。

（2）一致性。 本项目采用GL作为类前缀，对于通知的名称也应采用GL为前缀Notification为尾缀，宏定义应以k为前缀，对于枚举常量方法名的定义参考苹果原生API，总体来讲所有命名都应尽可能的与苹果API保持一致。

2.类命名

类名应该遵循驼峰命名原则。类名中应该包含一个或多个单词来描述这个类是做什么的。

3.类别命名

类名+个人标识+拓展名

例如UIView+GLExtension

类别的方法应该使用个人标识前缀加下划线加方法名避免与项目中的其他方法产生冲突。

详情参考SDWebImage的sd\_setImage：方法...

4.方法命名

方法使用小驼峰法命名, 一个规范的方法读起来应该像一句完整的话，读过之后便知函数的作用。执行性的方法应该以动词开头，小写字母开头，返回性的方法应该以返回的内容开头，但之前不要加get。如果有参数，函数名应该作为第一个参数的提示信息，若有多个参数，在参数前也应该有提示信息（一般不必加and）一些经典的操作应该使用约定的动词，如initWith,insert,remove,replace,add等等。

5.变量命名

（1）变量名使用小驼峰命名规则，使变量名尽可能可以推测其用途。

（2）类的成员变量用小驼峰命名法并加上下划线开头的方式命名。

（3）一般变量命名请使用简洁明了的方式并且开头不要使用下划线。

6.常量命名

常量(预定义，枚举，局部常量等)使用小写k开头的驼峰法。

7.图片资源文件命名

这个图片资源命名方式，以功能为组织形式，是一个很好的习惯，有利于查看资源文件。

原则：

1）采用单词全拼，或者大家公认无岐义的缩写(比如：nav，bg，btn等)

2）采用“模块+功能”命名法，模块分为公共模块、私有模块。公共模块主要包括统一的背

景，导航条，标签，公共的按钮背景，公共的默认图等等；私有模块主要根据app的业务

功能模块划分，比如用户中心，消息中心等

备注：建议背景图采用以bg作前缀，按钮背景采用btn作前缀（不作强制要求，项目实际

负责人根据团队特点确定即可）

公共模块命名示例：

导航条背影图片：bg\_nav\_bar@2x.png

导航返回按钮：bg\_nav\_back\_normal@2x.png，bg\_nav\_back\_selected@2x.png

标签item背景：bg\_tabbar\_record\_normal@2x.png，bg\_tabbar\_record\_selected@2x.png

私有模块命名示例：

以Joggers APP的用户中心图片资源为例说明，

uc——user center

用户中心头像默认图：bg\_uc\_avatar@2x.png

用户中心顶部默认背景图：bg\_uc\_top\_defaut@2x.png

用户中心底部背景图：bg\_uc\_bottom@2x.png

文件组织结构

1.类文件组织

iOS工程文件结构分物理结构和逻辑结构，建议逻辑结构和物理结构保持一致，以便方便有效地管理类文件。类文件组织要遵循以下两大原则：

基于MVC设计模式原则，至少要保证controller与数据处理，网络请求相对独立基于功能模块原则，功能模块分包括数据/网络处理，UI前端界面两部分，数据/网络处理应该在数据/网络处理的框架下，而UI前端界面比如用户中心，消息中心，它们的专有的controller，view等应该在属于文件夹。还会遇到一些公共的view，可以开辟出公共的文件夹来管理。

2.图片资源文件组织

图片资源文件采用Images.xcassets管理，不要使用自己创建的文件夹管理。

代码规范

1.删除多余的空行

（1） 所有方法与方法之间空1行

（2）所有代码块之间空1行

2.删除多余的注释

（1）删除注释掉的代码

（2）删除没有意义的注释

3.删除多余的方法

（1）如果方法没有使用到，请删除它

（2）如果方法没有执行任何业务逻辑，请删除它或者给出一定注释

4.删除未被使用的资源文件

5.添加必要的注释

（1）所有 .h 文件中的property 需要给出注释

（2）所有自定义的方法需要给出注释

（3）比较大的代码块需要给出注释

（4）所有代码中出现的阿拉伯数字需要给出注释

（5）程序中出现加密／解密 逻辑的操作地方，需要给出注释说明过程（无论是系统还是自定义）

6.整体代码风格需要统一

（1）代码后面的”{“ 不需要单独占用一行

（2）逻辑运算符 与 代码之前空一格

    \* “#pragma mark -” 与下面的代码之前不要空行

（3）遵循一般性的代码规范

通用规则

1 下面所有规则对第三方类库无约束

    \* 所有类、方法、属性等命名，做到见名知意，采用驼峰式命名规则

    \* 根据资源类型或者所属业务逻辑对项目资源进行分组，使得整个项目结构清晰明了

    \* 整个项目保持一种代码书写风格，让你的代码变的优雅！

2. 命名规范

    \* 所有类名称以项目工程开头命名，eg：“XP”、“ZJG”、“SZ”

    \* 针对不同视图控制器，在末尾添加后缀，eg：

    \* UIViewController  后缀添加“ViewController”

    \* UIView 后缀添加“View”

    \* UIButton 后缀添加“Button"

    \* UILabel 后缀添加“Label"

3. 单页代码最好控制在800行以内，每个方法最好不要超过100行，过多建议对代码进行重构

4. 相同的逻辑方法定义避免在多个地方出现，尽量将公用的类、方法抽取出来

5. 删除未被使用的代码，不要大片注释未被使用的代码，确定代码不会使用，请及时删除

6. 对其他项目中copy过来的代码，根据具体需要更新代码风格，及时删除未被使用的代码

7. 项目中所有Group或者文件名称（图片名字等），不要使用汉字命名，尽量使用英文命名，国内特有名词可以使用拼音。

8. 项目中所有Group都需要在项目目录中存在一个真实的目录，Group中的文件与真实目录中文件一一对应。

9. 请在项目中写必要代码的注释

10. 请多使用 #pragma mark - Mark Name 对方法进行分组 eg:

    \* #pragma mark - View lifeCycle

    \* #pragma mark - View lifeTerm

    \* #pragma mark - Init methods

    \* #pragma mark - Action methods

    \* #pragma mark - Common methods

    \* #pragma mark - UIActionSheetDelegate

    \* #pragma mark - UIImagePickerControllerDelegate

    \* #pragma mark - UITableViewDelegate Methods

    \* #pragma mark - UITableViewDataSource Methods

    \* #pragma mark - UIScrollViewDelegate Methods

    \* #pragma mark - UITextFieldDelegate Methods