安居客iOS代码规范手册

在阅读本手册前请先阅读NYTimes Objective-C Style Guide。

语言规范

命名规范

1. [强制]命名约定通用准则：清晰、一致性、不能自我指涉。

    - 清晰：命名应该既清晰又简短，但拒绝为了追求简短而丧失清晰性，拒绝为了简洁进行随意缩写。
    - 一致性：命名含义应该具有前后，全局的一致性，同个功能也应该使用同个名称。
    - 不能自我指涉：除掩码常量、通知外，名称不应该自我指涉（self-reference）。

2. [强制]尽量不用缩写，除以下已经长期使用形成共识的内容。

命名	说明
alloc	allocate
dealloc	deallocate
info	information
init	initialize
min	minimum
max	maximum
temp	temporary
int	integer
msg	message

3. [强制]禁止使用系统前缀开头，以防止冲突，原则上两字母前缀都是系统保留前缀，尽量使用三个大写字符最为前缀。

4. [强制]方法名、参数名、成员变量、局部变量都采用小写字符开头，驼峰命名法。

5. [强制]如果方法代表执行某个动作，应该以动词开头。

6. [强制]如果方法返回某个属性，应该直接以属性作为方法名，只有当间接返回对象或返回多个对象时，

可以使用get。

7. [建议]同一个类的一系列相关方法命名应该有一致性，入参更多的方法应该在入参更少的方法名后面增加新关键字。

```
例如：
- (instancetype)initWithURL:(NSURL *)URL;
- (instancetype)initWithURL:(NSURL *)URL cachePolicy:(NSURLRequestCachePolicy)ca
```

8. [建议]动词前可以根据具体情况增加 `can, should, will等` 情态动词使方法更明确。

9. [强制]不要使用一到两个字符的名称。

10. [建议]delegate方法名称的开头应标识出发送消息的对象所属的类。

```
例如TableViewDelegate
- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPa
```

常量定义

1. [强制]使用NS_ENUM声明枚举类型，枚举项以枚举类型为前缀。

```
typedef NS_ENUM(NSInteger, UIViewAnimationCurve) {
    UIViewAnimationCurveEaseInOut,         // slow at beginning and end
    UIViewAnimationCurveEaseIn,            // slow at beginning
    UIViewAnimationCurveEaseOut,           // slow at end
    UIViewAnimationCurveLinear,
};
```

2. [强制]使用NS_OPTIONS声明位移枚举常量，位移枚举常量可以组合使用。

```
typedef NS_OPTIONS(NSUInteger, UIViewAutoresizing) {
    UIViewAutoresizingNone                 = 0,
    UIViewAutoresizingFlexibleLeftMargin   = 1 << 0,
    UIViewAutoresizingFlexibleWidth        = 1 << 1,
    UIViewAutoresizingFlexibleRightMargin  = 1 << 2,
    UIViewAutoresizingFlexibleTopMargin    = 1 << 3,
    UIViewAutoresizingFlexibleHeight       = 1 << 4,
    UIViewAutoresizingFlexibleBottomMargin = 1 << 5
};
```

3. [强制]通常情况下，不要使用宏创建数值和字符串常量，而使用static声明变量，并在头文件中以extern

的方式暴露给外部。

```
例如：
.m文件：
NSString * const AFNetworkingReachabilityNotificationStatusItem = @"AFNetworking
.h文件：
extern NSString * const AFNetworkingReachabilityNotificationStatusItem;
```

4. [强制]Notification消息使用全局的NSString对象进行标识，名称按如下方式组合：

```
[Name of assoicated class] + [Did|Will] + [Unique part of name] + Notification
例如：UIApplicationDidEnterBackgroundNotification
```

类定义

1. [强制]使用property代替成员变量。
2. [建议]需要时在property声明里自定义getter或setter。

```
如
@property(nonatomic, getter=isHidden) BOOL hidden;
```

3. [强制]需要对外公开的声明放在.h中，其他都放在.m中。

4. [强制]+ (void)initialize必须判断class类型，因为任何继承类也会执行父类的initialize。

```
+ (void)initialize {
  if (self == [ClassName self]) {
    // ... do the initialization ...
  }
}
```

5. [建议]使用 `nonnull, nullable, __kindof等` 来修饰入参、返回值、属性。

6. [建议]使用designated initializer指定初始化方法：

   - designated initializer提供所有参数，secondary initializer提供一些默认参数来调用designated initializer。
   - 一个类应该仅有一个designated初始化方法，其他初始化方法应该调用该方法。
   - 继承时应该注意：
     - 子类designated应该调用**直接父类**的designated initializer方法。
     - **重载**直接父类的designated initializer，调用你的designated initializer。
     - 可以用 `__attribute((unavailable))` 标记不可用的父类designated initializer。

注释

1. [建议]使用Xcode自带工具插入默认格式，Option+Command+/即可自动插入。
2. [建议]复杂难懂的逻辑添加注释。
3. [建议]公共类头文件中暴露的方法都应该添加注释。

代码组织

1. [建议]一个类功能较多时，可以使用Category的方式进行功能划分，这些Category可以放在同一个文件中。
2. [建议]使用#pragma marks - 进行方法分组。
3. [强制]合理使用group或folder组织工程结构，文件folder结构和工程group结构要对应。
4. [建议]过期方法不要直接删除，先标记为deprecated。
5. [建议]函数内嵌套不要太深，大括号嵌套大括号不要超过三层。

Switch

1. [强制]Switch一定要实现default，防止case遗漏处理。
2. [强制]每个case必须写break。
3. [强制]每个case不管多长，用大括号括起来。

工程规范

CocoaPods使用规范

1. [强制]Podfile.lock文件必须提交到git版本控制，以保证版本可回溯。
2. [强制]Podfile中必须指定三方库的版本或SHA值，以防意外更新。
3. [强制]区分pod install和pod update的使用场景，默认使用pod install。
4. [强制]开发pod库在提交repo前，本地进行lint校验。

Git使用规范

1. [强制]commit message必须简洁清晰，禁止一两个单词或是语义模糊的内容。

   ```
   反例:
   fix bug
   update UI
   ```

2. [强制]本地分支提交前，**使用git pull --rebase将本地分支进行rebase**。

3. [强制]使用gitlab上的用户名和邮件，不要有多套用户名和邮件。

```
$ git config --global user.name "输入你的名字"
$ git config --global user.email abc@example.com
```

最佳实践

多线程

1. [强制]自建线程必须命名。
2. [强制]单例创建要使用dispatch_once确保线程安全。
3. [强制]在多线程环境下使用lazy load方式加载变量有crash风险，必须**加锁保护**
4. [强制]performSelector:withObject:afterDelay:要在有Runloop的线程里调用，否则无效。

```
异步线程默认是没有runloop，除非手动创建。
```

5. [强制]禁止随意创建常驻线程，除非在整个App生命周期有必要且有任务运行。
6. [强制]Notification在哪个线程中post，就在哪个线程中被转发，如果有UI操作，注意dispatch到主线程。
7. [强制]禁止在主线程进行文件IO操作，必须异步处理。
8. [强制]剪贴板读取必须放在异步线程处理。

内存管理

1. [建议]慎重使用单例，避免造成不必要的常驻内存。
2. [强制]Delegate需要使用weak引用。
3. [强制]使用block访问self时，使用weakify和strongify避免Retain Cycle。
4. [强制]strong引用子对象，weak引用父对象，基础类型使用assign，NSString、NSArray、block使用copy。
5. [强制]在dealloc方法中remove observer。
6. [强制]指定repeat参数是YES的timer，必须在合适的时机调用invalidate方法。
7. [强制]在init和dealloc中除父类属性外，禁止使用self访问属性，只允许通过成员变量直接访问。
   - 在init和dealloc截断，self是一个不完整的对象。
   - 子类可以重写accessor方法，在某些情况下可能导致异常。
8. [强制]在使用到UIScrollView，UITableView，UICollectionView的class中，需要在dealloc中将delegate，dataSource置为nil。

```
iOS9以上类的delegate和datasource由assign改为了weak，如果只支持iOS9以上，可以不手动置nil。
```

9. [强制]禁止一次性申请超过10MB的内存。

集合

1. [强制]插入对象和使用literals需要做非空判断。

```
@{@"key" : INCASE_EMPTYSTRING(value)}
@[@"value1", INCASE_EMPTYSTRING(value2)]
```

2. [强制]注意线程安全问题，必要时加锁，保障线程安全。
3. [强制]遍历可变集合时，先copy，再遍历临时变量。

```
NSArray *array = [mutableArray copy];
[array enumerateObjectsUsingBlock:xxxx];
```

4. [强制]禁止mutable对象作为入参传递，禁止返回mutable对象。
5. [建议]使用NSCache而不是NSMutableDictionary作为缓存。
6. [建议]容器使用泛型指定对象类型。

```
NSArray <NSString *> *stringArray;
```

字符串

1. [建议]使用keypath时，尽量使用NSStringFromSelector(@selector())，防止某个属性被删除后没有编译器检查。
2. [建议]取substring时考虑emoji字符问题，防止截到中间crash。

锁

1. [强制]专锁专用，一个lock对象只负责一个任务。
2. [建议]根据不同使用场景和性能要求，使用不同类型的锁。
3. [强制]使用锁return前，记得unlock，如果在try catch中，记得在finally中unlock。

IO

1. [建议]经常被读取的文件做好内存缓存，减少IO开销。
2. [强制]文件存储路径遵守一下规则：
    - Documents目录：存储用户数据，该路径可以通过iTunes存取，会被iTunes备份。
    - Library目录：
        - Preferences目录：包含应用程序偏好设置文件。
        - Caches目录：存放缓存信息，不会被备份。

- tmp目录：存放临时文件，不会被备份。
- 创建其他子目录：放置不希望被用户看到的数据，会被备份。

Category

1. [强制]category方法加上自定义前缀，防止冲突覆盖。
2. [强制]禁止category方法覆盖系统方法。