# iOS编码规范精简版－根据apple、google等规范总结而来

**1.格式和换行**

1.1 只使用2个空格来缩进，不使用tab.   
1.2 方法长度不超过100行，建议不超过80行。   
1.3 方法- 和 + 和返回值之前为1个空格；方法参数之间有一个空格，其他地方不出现多余的空格。   
1.4 条件语句的格式，推荐如下:

if (user.isHappy)

{

//Do something

}

else

{

//Do something else

}

或:(苹果官方代码缩进方式)

if (user.isHappy) {

//Do something

} else {

//Do something else

}

## 2.命名

2.1 命名统一使用驼峰命名法；只采纳有广为人知含义的缩写，比如info、msg、UI、HTTP这类。自造的缩写不被认可。总体的命名原则是清晰和一致，避免歧义。   
2.2 命名类、协议、常量和typedef结构体时考虑使用前缀,主要目的是为了避免冲突。方法名存在特定类的命名空间内，无需使用前缀。   
2.3 方法首字母小写（方法以大写缩略词开始除外）；文件夹名和类名首字母大写。不使用下划线作为私有方法的前缀，此方式被苹果保留。   
2.4 全局常量尽量不要使用宏定义。宏很可能被重定义，而且引用不同的文件可能会导致宏的不同，所以尽量使用const来定义常量。   
2.5 避免使用newXXX、getXXX、setXXX来命名变量和方法。

## 3.oc下的cocoa编码规范

3.1 使用#pragma mark来分类方法，参考以下结构：

#pragma mark - Lifecycle

- (instancetype)init {}

- (void)dealloc {}

- (void)viewDidLoad {}

- (void)viewWillAppear:(BOOL)animated {}

- (void)didReceiveMemoryWarning {}

#pragma mark - Custom Accessors

- (void)setCustomProperty:(id)value {}

- (id)customProperty {}

#pragma mark - IBActions

- (IBAction)submitData:(id)sender {}

#pragma mark - Public

(void)publicMethod {}

#pragma mark - Private

- (void)privateMethod {}

#pragma mark - Protocol conformance

#pragma mark - UITextFieldDelegate

#pragma mark - UITableViewDataSource

#pragma mark - UITableViewDelegate

#pragma mark - NSCopying

- (id)copyWithZone:(NSZone \*)zone {}

#pragma mark - NSObject

- (NSString \*)description {}

* 1

3.2 重载系统方法时，不要忘记先调用super的方法;比如init、viewdidload等。   
3.3 属性必须指定其类型：比如(nonatomic,strong)。   
3.4 使用#import引入oc/oc++头文件，使用#include引入c/c++头文件。   
3.5 尽量精简你的公开api接口。无需公开的方法应为私有。   
3.6 init和dealloc方法中避免使用self.来访问成员变量。   
3.7 dealloc方法中对变量的释放应和初始化时的顺序相反。   
3.8 对nsstring使用copy。   
3.9 不要手动抛出objective-c的异常（但你可以尝试捕获异常以增加应用的健壮性）。   
3.10 不同类型的强制转换需要注意是否造成数据丢失、溢出等问题。   
3.11 推荐使用@property来自动合成属性，如非必要无需再写@synthesize相关代码。

## 4.注释要求

注释一般用来解释代码的意图。要保持注释和代码同步更新，表达准确，避免误导。尽可能写自注释的代码。尽量使用单行注释而不是块注释；需要文档化的代码可以使用vvdocumenter-xcode插件来帮助生成。