# iOS编程规范

## 0.1前言

为􏰀高产品代码质量,开发人员编写出简洁、可维护、可靠、可测试、高效 、 可移植的代码,、规范的解释以及参考材料(what &why)。列出了一些业界比较优秀的编程规范,作为延伸阅读参考材料。

## 0.2代码总体原则

1、清晰第一清晰性是易于维护 、易于重构的程序必需具备的特征 。代码首先是给人读的,好的代码应当可以像文章一样发声朗诵出来。一般情况下,代码的可阅读性高于性能,只有确定性能是瓶颈时,才应该主动优化。

2、简洁为美简洁就是易于理解并且易于实现。代码越长越难以看懂,也就越容易在修改时引入错误。写的代码越多,意味着出错的地方越多,也就意味着代码的可靠性越低 。因此,提倡大家通过编写简洁明了的代码来提升代码可靠性。废弃的代码(没有被调用的函数和全局变量)要及时清除,重复代码应该尽可能提炼成函数 。

3、选择合适的风格,与代码原有风格保持一致产品所有人共同分享同一种风格所带来的好处,远远超出为了统一而付出的代价。

## 0.3术语定义

原则:编程时必须坚持的指导思想。

规则:编程时强制必须遵守的约定。

建议:编程时必须加以考虑的约定。说明:对此原则/规则/建议进行必要的解释。

示例:对此原则/规则/建议从正、反两个方面给出例子。

### 1.Cocoa和Objective-c特性头文件引用

1.1规则:#import引用Objective-C/Objective-C++头文件; #include引用C/C++头文件

1.2规则:使用#include时要注意#define头保护。

1.3建议:#import头文件按模块分类。

说明:对头文件按模块分类,使代码易读,也便于理解。

1.4建议:#import的顺序,应该是先引用自定义类,再引用系统类/框架。

说明:将系统类放在最后引入,可避免和检验自定义类的头文件引用不完整 漏

洞

1.5建议:#import根框架而不是单独的零散文件

说明:当你试图从框架(如Cocoa或者Foundation)中包含若干零散的系统头

文件时,实际上包 含顶层根框架的话,编译器要做的工作更少。根框架通常已

经经过预编译,加载更快。

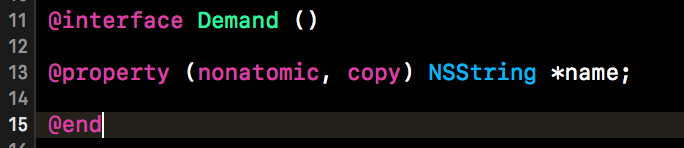
1.6建议:@class与#import的区别使用

说明:如需要继承类或执行协议,可以在.h中进行#import类或协议;其他情况下,在.h中声明用@classs声明此类即可。这样可以减少因头文件依赖引起重复编译,提高编译速度。

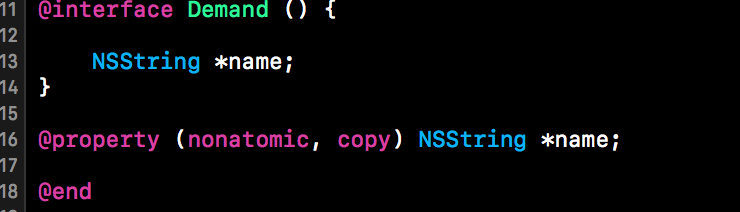
1.7规则:禁止对属性做出错误的内部成员变量声明

说明: xcode4.4开始会自动@synthesize ,为@ property声明一个前缀下划线的内部成员变量。因 此,不需要为属性重复做出内部成员变量声明。同时,禁止对属性做出错误的内部成员变量声明 。

示例：良好的风格：



错误的声明：



1.9建议:使用常类型变量,而不是内嵌的字符串字面值或数字

说明:常类型变量便于复用常用的变量值(如π),同时可以快速地修改值而无

需查找替换。

1.10建议:不用在init方法中,将成员变量初始化为0或者nil

说明:刚分配的对象,默认值都是0,除了isa指针(译者注: NSObject的isa识对象的类型)。所以不用在初始化器里面写一堆将成员初始化为0或者nil的代码。

1.12建议:字符串应使用copy属性(Attribute)

说明:应该用copy属性(attribute)声明NSString属性(property)。 从逻辑上,确保遵守NSString的setter必须使用copy而不是retain的原则。

1.13建议:谨慎声明属性的原子性

说明:一定要注意属性(property)的开销。缺省情况下,所有synthesize的setter和getter都是原子的。这会给每个get或者set带来一定的同步开销。将属性( property )声明为nonatomic,除非你需要原子性。

1.14建议:点引用只用于简单的属性set、get操作

说明:点引用只用于简单的属性set、get操作,但不应该用它来调用对象的其

它操作。