编码规范

1. 在类的头文件中尽量少引入其它头文件

(1) 除非确有必要，否则不要引入头文件。一般来说，应在某个类的头文件中使用向前声明来提及别的类，并在实现文件中引入那些类的头文件。这样可以降低类之间的耦合。

(2) 有时无法使用向前声明，如声明某个类遵循一项协议。可把这声明移至类的拓展，如果不行，把协议单独放至一个头文件并引入。

2. 多用字面量语法，少用与之等价的方法

(1) 字符串值：NSString *someStr = @” ”;

(2) 字面量数值：NSNumber **number = @1、@2.0f、@YES、@(a*b)**

(3) 字面量数组：NSArray *animals = @[@” cat”,@” dog”];

NSString *dog = animals[1]

(4) 字面量字典：NSDictionary *person = @{@” firstname” :@” fadf”}

插入 nil 会抛出异常 NSString *lastName = @” lastName”;

(5) 可变数组与字典:mutableArray[1] = @” dog”;

mutableDictionary[@” lastName”] = @” gally”;

(6) 使用字面量语法创建的都是不可变的，若想创可变对象需 copy 一份:

NSMutableArray *mutable = [@[@1,@2] mutableCopy];

得多调一方法，多创一对象，这是个缺点。

(7) 应该使用字面量语法创建字符串、数值、数组、字典，因为这比常规方法简明扼要

3. 多用类型常量，少用 define 预处理指令

(1) 用 static const NSTimeInterval kAnimationDuration = 0.3;替代#define ANIMATION_DURATION 0.3,，这样能令别人更易理解此定义意图

(2) 常用常量命名法：若常量仅在.m 使用，则前面加 k，若常量在类外可见，则以类名为前缀；

(3) 不要在头文件中声明预定义处理指令，因为可能名称冲突。Static const 也一样

(4)若不打算公开某常量，则应将其定义在.m 文件里

(5)若一变量声明为 static，又声明为 const，那么编译器不会创建符号，而像 #define 预处理命令一样仅做替换，不过，这种方式定义的常量带类型信息

(6)若要公开某常量：如派发通知，仅需使用字符串表示通知名称，可声明为一个外界可见的常量。这样，注册者无须知实际字符串值，只需以常量字符串来注册自己想接收的通知即可。

.h extern NSString *const EOCStringConstant’

.m NSString *const EOCStringConstant = @”“;

[[NSNotificationCenter defaultCenter]
postNotificationName:EOCStringConstant object:nil];

(7)预处理指令常量不包含类型信息，即使重定义相同常量，编译器也不会报错，这会导致常量值不一致

4. 用枚举表示状态、选项、状态码

(1)enum EOCConnectionState{

EOCConnectionStateDisconnected,

EOCConnectionStateConnected
};

用法：enum EOCConnectionState state = ;

typedef enum EOCConnectionState EOCConnectionState;

EOCConnectionState state=;

(2)enum UIViewAutoresizing{

UIViewAutoresizingNone = 0,

UIViewAutoresizingLeft = 1 << 0,

UIViewAutoresizingRight = 1<<1
}

这种定义每一值对应一个二进制表示只有 1 个二进制值为 1，则用按位或操作符可以组合多个选项：left|right

例：MYSearchType type = MYSearchTypeHome|MYSearchTypeHotRecruit;

if(type&MYSearchTypeHome)

//火热招聘

(3) typedef enum EOCConnectionState : NSInteger EOCConnectionState;

```
enum EOCConnectionState:int{
 EOCConnectionStateNone = 0,
 EOCConnectionStateLeft = 1 << 0
}
```

(4) typedef NS_ENUM(NSUInteger,EOCConnectState){

```
 EOCConnectStateNone = 0,
 EOCConnectStateLeft
}
```

(5) 在枚举的 switch 语句中尽量不要出现 default 语句，这样加入新的枚举后，编译器会提示开发者处理该枚举

5. 在对象内部尽量直接访问实例变量

(1) _name 直接访问实例变量，速度快，编译器生成的代码会直接访问保存对象变量的那块内存

(2) 直接访问实例变量时，不调用 set 方法，绕过了其定义的内存管理语义，如 copy 属性的变量，则不会进行 copy 操作

(3) 直接访问实例变量不会触发键值观察 kvo 通知

(4) 通过属性来访问好处：可打断点，易排察

(5) 合理折中方案：写入实例变量时用设置方法，读取实例变量时直接读取

(6) 在初始化方法及 dealloc 方法中，应直接通过实例变量进行读写操作

(7) 有时用了惰性初始化技术配置数据，则必须通过属性来访问(.语法)

6. 以类族模式隐藏实现细节(工厂方法为其一种形式)

(1) 自定义抽象基类

.h 文件 typedef NS_ENUM(NSUInteger,EOCExampleType){

```
        EOCExampleTypeA,
        EOCExampleTypeB
    }
```

```
@property (copy, nonatomic) NSString *name;
@property (assign, nonatomic) NSUInteger salary;
+ (EOCEmployee *)employeeWithType:(EOCEmplyeeType)type;
- (void)doADaysWork;
```

.m 文件

```
+employeeWithType:{
    case EOCExampleTypeA:
    return [[NSClassFromString(@“EOCExampleTypeA”) class] new
}
- doADaysWork{
    //由子类具体实现
}
```

注：系统框架常用类族