iOS 코딩 스타일 가이드

for Objective-C

2014년 7월 27일

개정이력

개정번호	작성자	개정 내역	개정일자
0.5.0	정낙천	최초작성	2014/07/29
0.6.0	정낙천	주석 스타일 추가	2014/07/30

목차

이름짓기 및 선언

공통

- 이름은 full name으로 만든다.
- 약자를 쓰지 않는다.
- 필요시 전치사도 사용한다.
- Camel 형태로 작명하며, 언더바(_)를 사용하지 않는다.
- 탭은 스페이스 4칸이며, 기본 들여쓰기도 스페이스 4칸이다.
- 콤마(,)뒤에 스페이스 한칸을 둔다. (ex => a, b, c)
- C, C++ 스타일의 NULL을 사용하지 않고 nil을 사용한다.
- 3항 연산자는 되도록 쓰지 않는다. (권고사항)
- 포인터는 타입 옆에 두고 한칸 띈다. (권고사항)(ex => Type* variableName)

```
// 좋지 않은 예
return _numberOfItems? (index - floorf((CGFloat)index / (CGFloat)_numberOfItems) * _numberOfItems): 0;
```

`@property (nonatomic, strong) NSMutableSet *itemViewPool;`	나쁨
`@property (nonatomic, strong) NSMutableSet* itemViewPool;`	좋음
`-(NSString *)getCurrentTime`	나쁨
`-(NSString*)getCurrentTime`	좋음

주석

- 함수와 클래스 이름만으로도 역할을 분명히 알수 있도록 이름을 짓는다. 그러면 주석을 안달아도 나중에 읽기 쉽다.
- 만약에 주석을 구구절절이 달아야하는 경우가 생긴다면 보통 함수를 잘잘히 나누어서 문제를 해결할 수 있다.

클래스 이름

- 대문자로 시작하며 Prefix로 'MS'를 추가한다. 추가적인 카테고리가 필요하면 MS뒤에 임의로 Prefix를 추가해준다.
- 한 파일에 한개의 클래스만 선언한다.
- 클래스 이름과 파일 이름은 동일해야 한다.

MSDownloadViewController	MS 가 prefix
MSViewerViewController	MSViewer가 prefix
MSDebuggingInfoLoggingMgr	나쁨
MSDebuggingInfoLoggingManager	좋음

함수 이름

- 소문자로 시작한다.
- 함수 특성에 따라서 'selector', 'private', 'create'와 같은 prefix를 붙여준다.

- 헤더파일에 선언되서 외부에 노출되는 함수는 prefix를 붙이지 않는다.
- 같은 prefix가 붙은 함수끼리 붙여서 배치한다.
- 헤더에서 @property 로 선언되지 않는 객체에 대해서 set/get으로 시작하는 함수를 쓰지 않는다.

`-(void)uploadException`	내부에서만 쓰일경우 나쁨, 헤더에 선언된 것이면 좋음
`-(void)privateUploadException`	내부에서만 쓰일경우 좋음, 헤더에 선언된 것으면 나쁨
`-(void)selectorRefreshList`	좋음
`-(void)setCurDic:(NSMutableDictionary *)curDic`	나쁨, property가 선언되지 않았는데 set/get을 사용함

함수 선언

- 함수 선언사이에는 스페이스(‘ ‘)가 들어가지 않도록 한다.
- 외부에 노출되지 않는 private 함수는 category를 m 파일 상단에 prototype 을 선언하고, 코드에서 사용하고 구현한다.
- 함수 구현 선언후 시작 중괄호는 다음 라인에 한다.

`- (BOOL) createListViewItems;`	좋음, 스페이스가 중간에 없음
`-(BOOL)createListViewItems;`	좋음, 스페이스가 중간에 없음

`// MSQAViewerViewController.m 파일, 함수 선언 및 사용, 구현 좋은예`
`@interface MSQAViewerViewController(selectors)`
`-(void)selectorRefreshList;`
`@end`
`…`
`UIBarButtonItem* refreshButton = [[UIBarButtonItem alloc]`
`initWithBarButtonSystemItem:UIBarButtonSystemItemRefresh target:self`
`action:@selector(selectorRefreshList)];`
`…`
 	
`-(void)selectorRefreshList`
`…`	

`- (BOOL)privateStopAudio`
`{`
`  …`
`}`
	좋음

`- (BOOL)privateStopAudio {`
`  …`
`}`
	나쁨

멤버변수 이름

- 헤더 또는 빈 카테고리에 선언하는 변수들에게 해당하는 규칙이다.
- 언더바(_)로 시작한다.

- @property 로 선언된 객체는 self.object로 쓰지 않고 언더바(_)로 접근한다.
- id 선언된 객체는 protocol을 준수하도록 한다.
- 탭을 잔뜩 넣어서 변수끼리 줄맞춤을 하지 않는다. 변수 타입 선언후 중간에 스페이스 한칸만 넣는다.

```
// MSBookmarkToggleView.h 헤더에 선언시
@interface MSBookmarkToggleView : UIView
{
    IBOutlet UIView* _viewBookmarked;
    IBOutlet UIView* _viewBookmarkCanceled;

    id<ZipArchiveDelegate> _delegate;

    BOOL _bIsBookmarked;
}
@end
```

```
// MSQAViewerViewController.m 구현 파일에 선언시
@interface MSQAViewerViewController()
{
    UITableView* _tableList;
    MSQAViewerListManager* _listManager;
}
@end
```

`@property (nonatomic, assign) id subViewControlDelegate;`	나쁨
`@property (nonatomic, assign) id<MSSubViewControlDelegate> subViewControlDelegate;`	좋음
`self.delegate = target;`	나쁨
`_delegate = target;`	좋음

`NSString * hostName;`
`NSNumber * ipAddress;`
`NSArray * accounts;`
	나쁨

`NSString* _hostName;`
`NSNumber* _ipAddress;`
`NSArray* _accounts;`
	좋음

지역변수 이름

- 모두 대문자로 쓰며 중간에 언더바(_)를 넣지 않는다.
- 탭을 잔뜩 넣어서 변수끼리 줄맞춤을 하지 않는다. 변수 타입 선언후 중간에 스페이스 한칸만 넣는다.

```
// 나쁨
NSString * streetAddress  = @"1 Infinite Loop";
NSString * cityName       = @"Cupertino";
NSString * countyName     = @"Santa Clara";
```

```
// 좋음
NSString* streetAddress = @"1 Infinite Loop";
NSString* cityName = @"Cupertino";
NSString* countyName = @"Santa Clara";
```

전역변수 이름

- 모두 대문자로 쓰며 중간에 언더바(_)를 넣는다.

- 스트링하고 값들만 전역변수로 선언하며 코드 조각을 전역 매크로로 작성하지 않는다.

`#define IPad_Resolution @"1024x768"`	나쁨
`#define IPAD_RESOLUTION @"1024x768"`	좋음
`#define APP_VERSION [[[NSBundle mainBundle] infoDictionary] objectForKey:@"CFBundleShortVersionString"]`	나쁨, 싱글톤 클래스에서 static 함수로 제공할 것

제어문

---

operators

- 특별한 케이스가 아니면 ++, - - 는 변수 앞에 쓴다.

`pageCount++;`	나쁨
`++pageCount;`	좋음
`[self downloadPage:downpageIdx++];`	이런경우 허용
`for (int i = nStartLoadPage; i <= nEndLoadPage; i++) {`	나쁨
`for (int i = nStartLoadPage; i <= nEndLoadPage; ++i) {`	좋음

---

if/else

- nil 검사는 느낌표(!)표만 검사한다. (권고사항)

```
if (button.enabled) {
    // Stuff
}
else if (otherButton.enabled) {
    // Other stuff
}
else {
    // More stuf
}
```

```
// Comment explaining the conditional
if (something) {
    // Do stuff
}

// Comment explaining the alternative
else {
    // Do other stuff
}
```

- 코드 한줄이 들어가도 중괄호({ })로 막는다.

`if ([self privateCheckIsMSPS20Format:indexPath.row pData:pData]) {`
`    cell.textLabel.text = [NSString stringWithFormat:`
`                                    @"%@ (msps20)", cell.textLabel.text];`
`}`
	좋음

`if (bStopLoop) {`
`    break;`
`}`
	나쁨

`if (bStopLoop) break;`	나쁨

`if (bStopLoop) {`
`    break;`
`}`
`else return;`
	나쁨

---

switch

```
switch (something.state) {
    case 0: {
        // Something
        break;
    }

    case 1: {
        // Something
        break;
    }

    case 2:
    case 3: {
        // Something
        break;
    }

    default: {
        // Something
        break;
    }
}
```

---

for

```
for (NSInteger i = 0; i < 10; ++i) {
    // Do something
}


for (NSString* key in dictionary) {
    // Do something
}
```

while

```
while (something < somethingElse) {
    // Do something
}
```

Objective-C

@property

- 소문자로 시작해서 camel 형태로 작명한다.
- 외부 클래스에서 필요로하는 변수만 property로 작성한다. (캡슐화를 위해서 필수이다.)
- NSString 으로 선언된 변수는 무조건 copy 특성을 준다.
- id 또는 NSObject 객체는 assign 특성을 준다.
- id 또는 NSObject 객체는 dealloc 에서 nil을 모두 대입해준다. (ARC가 자동으로 해주기는 하지만 컴파일러가 완벽하지 않다. 기본 매너로 지킬것을 요구한다.)
- id 선언된 객체는 protocol을 준수하도록 한다. (protocol을 선언해서 사용하자)
- property를 접근하는 함수를 따로 만들경우에는 get과 set으로 시작하고 property 이름이 첫글자를 대문자로 만든다.

```
@property (nonatomic, assign) id<Protocol> object;
@property (nonatomic, assign) NSObject<Protocol>* object;
@property (nonatomic, copy) NSString* string;
```

- property를 선언할때 set또는 get의 함수를 쓰지말고 대입연산자(=)를 이용해서 대입하거나 값을 가져올 것

`[_backView setBackgroundColor:[UIColor clearColor]];`	나쁨
`_backView.backgroundColor = [UIColor clearColor];`	좋음

코코아 delegate 패턴

- delegate 오브젝트들은 무조건 assign으로 받으며, dealloc 과 적당한 장소에서 nil을 대입해서 객체의 쓸데없는 접근을 막는다. retain은 ARC 환경에서도 리테인 싸이클을 유발하므로 절대로 쓰지 말아야 한다.
- _delegate 로 접근한다.
- 한클래스에서 delegate하는 오브젝트가 많으면 _delegateForSomething 형식으로 prefix 작명한다.
- @protocol을 이용해서 API 콜백함수를 무조건 작성한다. 즉, performSelector를 사용하지 말라는 것임.

백그라운드/포그라운드 스레드 설정

- GCD를 이용한 블록 코딩을 우선사용하며 performSelector 의 사용을 지양한다.
- 백그라운드 스레드를 사용하는 클래스는 헤더 파일에 변수 _dqueueBackgroundThread 를 선언해서 init 함수에서 초기화하고 dealloc 함수에서 release 해준다. 백그라운드 수행시마다 변수 _dqueueBackgroundThread를 이용해서 백그라운드 스레드를 dispatch 한다.

```objc
@interface MSPS20DownloadManager : NSObject <NSURLConnectionDelegate>
{
    // process typte
    dispatch_queue_t _dqueueBackgroundThread;
}
```

```objc
-(id)init
{
    self = [super init];
    if (self) {
        _dqueueBackgroundThread = dispatch_queue_create("com.media-story.MSPS20.MSPS20DownloadManager", NULL);
    }
    return self;
}
```

```objc
-(void)dealloc
{
    if (_dqueueBackgroundThread) {
        _dqueueBackgroundThread = nil;
    }
}
```

```objc
dispatch_async(_dqueueBackgroundThread, ^{
    [self downloadPage:[[[MSPS20CommandCenter getInstance] progressiveFileHandler] prodDic]];
});
```

- 포그라운드 스레드는 함수 dispatch_get_main_queue()를 사용한다.

```objc
dispatch_async(dispatch_get_main_queue(), ^{
    [self setLoadingMsg:@"config file download complete."];
    [self setLoadingProgress:0.5];
});
```

싱글톤 클래스 생성

- 싱글톤 클래스는 AppDelegate에 해당 클래스를 주렁주렁 매달지 말고 아래와 같은 코드로 생성할 것

```
+ (MSPS20DownloadManager*)getInstance
{
    static MSPS20DownloadManager *sharedInstance = nil;
    static dispatch_once_t onceToken;

    dispatch_once(&onceToken, ^{
        sharedInstance = [[self alloc] init];
    });
    return sharedInstance;
}
```

기타

주석

- 아래의 절차로 주석처를 하거나, 수동 주석처리를 한다.
  - “cmd+[”로 첫 들여쓰기를 없앤다.
  - “cmd+/”로 주석을 처리한다.
  - “cmd+]”로 주변과 들여쓰기를 맞춘다. 또는 “cmd+a”, “cotrol+a”로 문서 전체 들여쓰기를 맞춘다.

```
// 좋은 주석
-(BOOL)privateDoParseResults:(NSData*)data
{
    // parsing
    //NSError* error = nil;
    NSDictionary* infoDic = nil;
    //NSDictionary* infoDic = [MSPS20XMLReader dictionaryForXMLData:data
    //                                                      options:XMLReaderOptionsProcessNamespaces
    //                                                        error:&error];
…
}
```

```
// 나쁜 주석
-(BOOL)privateDoParseResults:(NSData*)data
{
    // parsing
//    NSError* error = nil;
    NSDictionary* infoDic = nil;
//    NSDictionary* infoDic = [MSPS20XMLReader dictionaryForXMLData:data
//                                                      options:XMLReaderOptionsProcessNamespaces
//
…
}
```

- 사용하지 않는 코드는 나중에 쓸지 모르더라도 삭제한다. 만약 보관하고 싶다면 git branch 해서 사본을 보관한다.
- 기능이 완료되지 않은 상황에서 다른 부분의 작업이 들어가야하는 상황이 발생하면 “// TODO:”로 주석을 삽입하고 설명을 넣는다. 이 방법은 xCode에서 TODO가 두드러지게 보여주는 기능이 있기 때문에 유용하다.
- 의미없이 주석을 이용해서 라인을 사용하는 것은 가급적 자제 한다. 예를 들면 “//”와 같은 주석은 넣지 않는다.

class 구현 파일 (.m/.mm) 템플릿/

- 클래스 prefix는 “MS”를 붙이며 camel 형태로 이름을 짓는다. 약자를 넣지 않으며 전치사를 포함해서 full 문장으로 구성한다.
- 클래스 생성은 Cmd+n으로 생성하고, 비슷한 함수나 클래스는 텍스트 자체를 복사하여 가져온다. 이 방법은 불필요한 코드가 같이 딸려서 코드에 복사되는 경우를 방지하며, 개발자에게 알고리즘 재확인 및 복사한 코드를 재확인하는 기회를 주기 때문에 필요하다.
- 클래스 생성 후 아래 코드의 템플릿을 복사 붙여넣기 해서 사용한다. 해당 템플릿에 함수 선언과, 구현을 채워넣는다는 생각으로 작성한다.
- 예제: 구현 파일

```
————————————————————————————————————————————————————
//
//  MSCodingStyleMain.m
//  CodingStyleGuideTest
//
//  Created by Earth on 2014. 7. 23..
//  Copyright (c) 2014년 MediaStory. All rights reserved.
//

#import "MSCodingStyleMain.h"

#pragma mark - enum Definition

/*****************************************************************************
 * enum Definition
 *****************************************************************************/


/*****************************************************************************
 * String Definition
 *****************************************************************************/
```

```
/**********************************************************************************
 * Constant Definition
 **********************************************************************************/


/**********************************************************************************
 * Function Definition
 **********************************************************************************/


/**********************************************************************************
 * Type Definition
 **********************************************************************************/

@interface MSCodingStyleMain()
// 여기에 private 변수 선언, , prefix는 "_"
@end

@interface MSCodingStyleMain(CreateMethods)
// 여기에 create/init 관련 함수 선언, prefix는 "create"
@end

@interface MSCodingStyleMain(PrivateMethods)
// 여기에 private 함수 선언, prefix는 "private"
@end

@interface MSCodingStyleMain(selectors)
// 여기에 @selector() 안에 들어가는 함수 선언, prefix는 "selector"
@end

@interface MSCodingStyleMain(IBActions)
// IBACtion methods
// 여기에 xib와 연결되는 IBAction 함수 선언, prefix는 "selector"
@end

@interface MSCodingStyleMain(ProcessMethod)
// processing methods
```

```
// 여기에 메인 기능을 수행하는 processing 함수 선언, prefix는 "process"
@end


/*****************************************************************************************
 * Implementation
 *****************************************************************************************/
@implementation MSCodingStyleMain

#pragma mark - class life cycle
// 이 안에 init, initWithNibName, initWithFrame, drawRect, viewDidLoad,
// didReceiveMemoryWarning, initWithCoder, dealloc 등 클래스 라이프 사이클과 관련된 함수 들을 이안에 넣는다.


#pragma mark - create methods
// CreateMethods 카테고리 안에 선언된 함수들을 이 안에 넣는다.


#pragma mark - process methods
// ProcessMethod 카테고리 안에 선언된 함수들을 이 안에 넣는다.


#pragma mark - private methods
// PrivateMethods 카테고리 안에 선언된 함수들을 이 안에 넣는다.


#pragma mark - selectors
// selectors 카테고리 안에 선언된 함수들을 이 안에 넣는다.


#pragma mark - IBActions
// IBActions 카테고리 안에 선언된 함수들을 이 안에 넣는다.


#pragma mark - <Protocol1로 선언된 Delegate의 full name>
// 해당 클래스가 사용하는 protocol1의 함수들을 이 안에 넣는다. 이 함수들은 카테고리로 따로 선언하지 않는다.


#pragma mark - <Protocol2로 선언된 Delegate의 full name>
```

```
// 해당 클래스가 사용하는 protocol2의 함수들을 이 안에 넣는다. 이 함수들은 카테고리로 따로 선언하지 않는다.
// 이런형태로 반복해서 프로토콜을 구현하는 함수를 넣는다.

@end
```

참고 싸이트

애플 코딩 가이드

- https://developer.apple.com/library/mac/documentation/Cocoa/Conceptual/CodingGuidelines/Articles/NamingMethods.html#//apple_ref/doc/uid/20001282-BCIGIJJF

구글 코딩 가이드

- http://google-styleguide.googlecode.com/svn/trunk/objcguide.xml

---

구글 코딩 가이드

- http://google-styleguide.googlecode.com/svn/trunk/objcguide.xml

---

뉴욕타임즈 코딩 가이드

- https://github.com/NYTimes/objective-c-style-guide

GitHub 코딩 가이드

- https://github.com/github/objective-c-style-guide

Sam Soffes 코딩 가이드

- https://gist.github.com/soffes/812796

CocoaDevCentral 코딩 가이드

- http://cocoadevcentral.com/articles/000082.php