Objective-C 快速入门

本文主要介绍 **Objective-C** 的基本语法，传统的面向对象语言的封装，继承以及多态在 **Objective-C** 上会是什么样子。适合于熟悉面向对象编程的读者。

Objective-C：加强版的 C?

Objective-C 是基于标准的 **ANSI C** 的一门面向对象语言。其语法和设计主要基于 Smalltalk，所以有些类似，而且支持标准的 **C 语言**语法。代码文件的类型有三种:[.h]头文件，[.m]C 代码，[.mm]C++代码。定义了一种新的调用头文件的方法#import，这样不会重复#include 头文件，只在没有调用过的时候调用一次，类似于 PHP 的 require_once。

最常用的数据类型：字符串 NSString

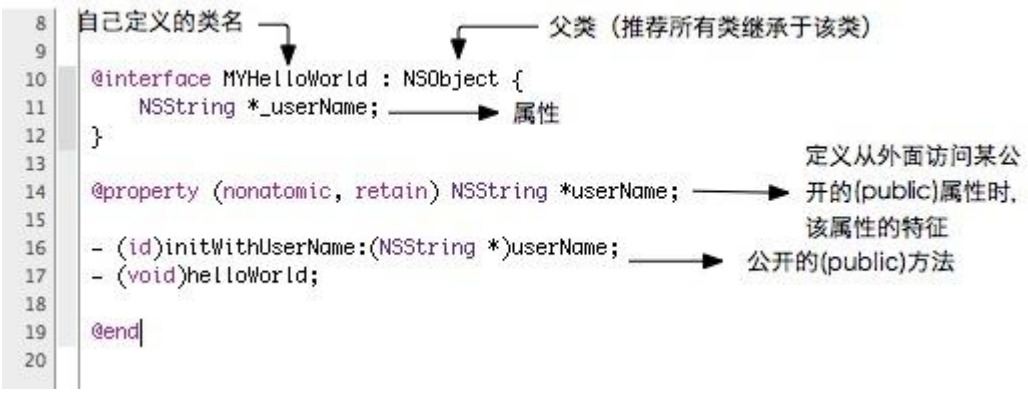
Objective-C 将字符串数据类型定义成为 Class，支持可变长度的字符串，支持 Unicode 等等一些实用的方法。通过使用@标志符，可以方便的由字符串常量定义 NSString 对象。下面给出一些定义 NSString 的代码实例。

```
NSString *string1 = @"const string";
NSString *string2 = [NSString stringWithFormat:@"%d %d %s", 1, 2, "hello string"];
```

类（Class）

作为面向对象语言最基本也是最重要的数据类型，**Objective-C** 当然不会不支持。**Objective-C** 定义类的特点在于先在.h 头文件里定义接口，然后在.m 代码文件里实现。

关于.h 头文件的具体的语法看下面的图更直接一点。



关于.m 代码文件的具体语法也请参考下面的图。

```
#import "MYHelloWorld.h"                                  调用头文件

@implementation MYHelloWorld                              申明实现该类

@synthesize userName = _userName;                         定义外部可访问的属性

- (id)initWithUserName:(NSString *)userName {            初始化该类的方法的实现
    self = [super init];                                  调用父类的初始化方法
    _userName = [userName copy];
    return self;                                          返回类的ID，类似于指针
}

- (void)helloWorld {                                      普通的方法的实现
    NSString *text = [NSString stringWithFormat:@"%@, Hello World!", _userName];
    printf([text UTF8String]);
}

- (void)dealloc {                                         内存释放
    [_userName release];
    [super dealloc];
}

@end
```

大家注意一下 initWithUserName()方法的返回值的数据类型为 id，是不是感觉这个一般的语言有点区别。其实，这种数据类型类似于 C++的指针。在 **Objective-C** 里面仍然叫做指针类型。**Objective-C** 的对象的定义，可以支持强类型的实例，以及弱类型的实例的定义。简单的说，强类型就是定义该实例的时候必须指明类的名字，弱类型就是没有指明类的名字。下面给大家看看实际的代码就会明白，定义这两者的时候的区别。

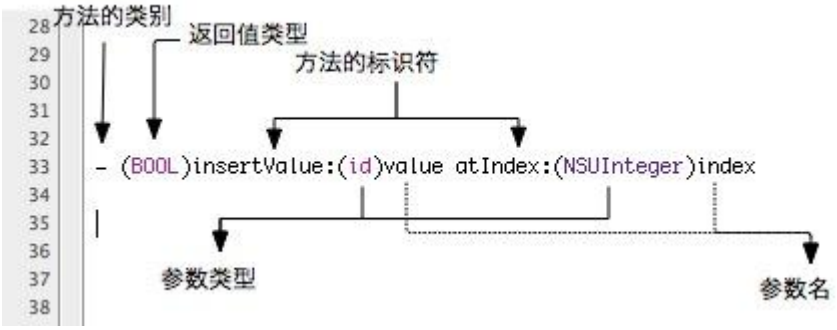
```
MYHelloWorld *myHelloWorld1;                // 强类型（Strong typing）
id                          myHelloWorld2;                // 弱类型（Weak typing）
```

顺别提一下，弱类型能给实现**设计模式**（**Design Patterns**）的时候带来很大的方便。

方法（Methods）

方法 Methods 又可称为函数。在 **Objective-C** 里的类可以定义两种方法。一种是**实例的方法**，一种是**类的方法**。**实例的方法**局限于某个类的实例，也就是必须定义这个类的实例之后，才能被调用执行的方法。**类的方法**不需要创建实例，直接通过类的名称就可以被调用执行的方法。

定义一个方法需要：方法名（一个或者多个关键字），返回值类型，参数类型和参数名。下面这的图详细的说明了如何定义一个**实例的方法**，其中负号[-]表示该方法为**实例的方法**，该方法的名称加上各个关键字包括冒号即为[insertValue:atIndex:]。



Objective-C 调用方法是通过**发送消息**给对应的实例对象。**发送消息**的方式其实是和一般的编程语言一样就是调用实例对象的方法，**Objective-C** 独特的地方就是方法的调用是通过一个方法名＋零个或多个标示符＋零个或多个参数，然而一般的编程语言只需要一个方法名＋零个或多个参数就可以了。**Objective-C** 里面之所以把调用方法称为发送消息，大概是因为所有的消息发送之后都是动态传递给实例对象的。并且，如果一个子类定义了一个和父类相同方法名＋标示符的方法之后，子类会先收到该消息，然后选择性的是否继续将该消息传递给父类。

发送消息是通过一对方括号[]来实现的。在括号的里面，实例对象在左边，消息以及参数等的定义在右边。例如：

```
[anObject insertValue:anObj atIndex:1];
```

为了避免生成多余的临时变量，Objecttive-C 容许直接使用消息的结果。如下例：

```
[[anObject getArray] insertValue:[anObject getValueToInsert]
atIndex:0];
```

类的方法，类似于静态方法(Static Function)，常用于作为工厂模式中用来生成新的实例。定义的时候和**实例的方法**有区别的地方就是开头的符号为**加号[+]**。调用的**类的方法**和调用(Static Function)基本相同，直接通过类名就可以，如下例：

```
NSMutableArray*     myArray = nil;       // nil 等同于通常的 NULL

// 创建一个动态数组，并且制定最初的大小
myArray = [NSMutableArray arrayWithCapacity:19];
```

属性

属性是一个可以取代方法的方便符号。在类中声明定义属性时，并不创建新的实例，就是一个能够方便快捷的用来访问实际已经存在的变量的方法。也就是说，属性其实并没有真正的存储任何数据。

其实属性是可以让你在编写代码时减少影响效率的冗余代码。简单的说属性要比 getter 和 setter 方法快。

属性还可以定义一些访问该属性时的限制或者如何获取该属性，例如指定 copy 用来复制该属性，readonly 用来指定该属性只读。

```
@property BOOL workFlag;
@property (copy) NSString* oString;    // 通过复制来使用该属性.
@property (readonly) UIView* oView;    // 定义一个类似于 getter 的属性.
```

调用某实例的属性有两种方法，一种是通过括号，和调用方法类似，另外一种是通过点符号[.]。第二方法比较方便，也符合大多数开发人员的习惯。

```
        // 第一种调用方法
        [myObject setFlag:YES];
        CGRect     viewFrame = [[myObject rootView] frame];
        // 第二种调用方法
        myObject.flag = YES;
        CGRect     viewFrame = myObject.rootView.frame;
```

协议和委托（Protocols and Delegates）

Objective-C 中的协议（Protocol）类似于常用的接口，协议（Protocols）中定义的方法，在类中实现。

```
@protocol MyFirstProtocol
        - (void)myFirstProtocolMethod;
@end
```

在 iPhone OS 中，协议（Protocol）通常用来实现委托对象（Delegate Object）。委托对象（Delegate Object）一般用来自己定义行为或者动作，也就是调用自己定义方法，但自己不实现该方法，委托其它的类来实现该方法。

UIApplication 类就是一个典型的例子。UIApplication 类中定义了一个应用程序应有的行为或者动作。而不是强制让你的 UIApplication 子类去接受当前应用程序的状态消息并做出相应处理。UIApplication 类通过调用特殊的方法，来传递这些消息给它的委托对象。这个委托对象通过实现名为 UIApplicationDelegate 的协议（Protocol），之后就可以接受到当前应用程序的状态消息并做出相应处理。比如内存不够的错误，应用程序被中断等重要消息。

下面给出一段关于 HelloWorld 的实例代码：

```
//main.m
#import
int main(int argc, char *argv[])
{
        NSAutoreleasePool * pool = [[NSAutoreleasePool alloc] init];
        int retVal = UIApplicationMain(argc, argv, nil, nil);
        [pool release];
        return retVal;
}
```

本文到这里就结束了，如果您有什么意见请在下面发表评论。本文只是针对有一定编程经验的开发人员，简单的介绍了 Objective-C 的主要特征。本站将会陆续推出各种 iPhone 开发的相关资讯，敬请大家关注。