Block (代码块)

概述

- 代码块对象简称为“代码块“，是对C语言中函数的扩展。运行在iOS4.0和OS X 10.6版本以上
- 代码块实际上是由C语言实现的，所以在各种以C作为基础的语言内都是有效的包括：Objective-C, C++ 以及Objective-C++
- 代码块本质上是和其他变量类似。不同的是，代码块存储的数据是一个函数体。使用代码块时，可以像调用其他标准函数一样，传入参数数，并得到返回值。
- 代码块不属于ANSI的C语言标准。关于代码块的提议已经提交给了C程序语言标准团体。

函数指针 VS 代码块

函数指针的语法格式

```
返回值类型 (*函数名)(参数列表);
eg: void (*myFunction) (void);
```

代码块的语法格式

```
返回值类型 (^代码块名)(参数列表) = ^返回值类型（参数列表）{表达式;};
eg: void (^myBlock)() = ^{printf("Hello, Block\n");};
```

说明:

返回值类型和参数列表都可以有条件省略
省略返回值类型时，若表达式中有return语句就使用该返回值的类型，若没有，则为void
省略返回值类型时，若表达式中又多个return语句，则所有return语句返回值类型必须相同
如果不适用参数，则参数列表也可以省略

代码块的使用

直接使用

```
void (^myBlock)() = ^{printf("Hello, Block\n");};
myBlock();
```

代码块内联

```
NSArray *array = @[@"One", @"Two", @"Three", @"Four"];
NSLog(@"Unsorted array: %@", array);
NSArray *sortedArray = [array
sortedArrayUsingComparator:^NSComparisonResult(id obj1, id obj2) {
        return [obj1 compare:obj2];
    }];
NSLog(@"Sorted array: %@", sortedArray);
```

使用typedef定义相同类型的代码块

```
typedef NSComparisonResult (^NSComparator)(id obj1, id obj2);
```

代码块的变量

- 全局变量和静态变量在代码块内部可以正常使用（可以修改其值）
- 局部变量在代码块内部无法修改其值，原因是代码块的局部变量在编译阶段被看做为常量
- 如果需要在代码块中修改局部变量，需要在局部变量前面加__block修饰符