Swift vs Objective C

多少人看过Swift语法？

>>>>>>读一段代码<<<<<<

一、Swift的背景和现状

二、Swift 和 OC的不同之处

三、Swift的新技巧

四、我们打算怎么用？

一、Swift的背景和现状

Swift的背景和现状

2014年9月11日发布swift 1.0

2015年2月12日发布swift 1.2

2015年6月9日发布swift 2.0

2015年9月23日发布swift 2.1

今年年底开源

二、Swift 和 OC的不同之处

Swift新增关键字

deinit、func、let、struct、subscript、typealias、var、fallthrough、as、dynamicType、is、get、inout、mutating、nonmutating、override、operator、infix、postfix、prefix、unowned、weak、willSet、associativity、set、didSet、left、none、precedence、right

1、Property

2、KVC & KVO & Observer

3、内存

4、Lazy Load

5、Class & Struct

6、Method Swizzling

Property

OC:

```
@property (nonatomic, strong) NSDate *date;
```

Swift:

```
var date: NSDate {
        get { return date }
        set { self.date = newValue }
    }
```

KVC & KVO & Observer

对于NSObject的子类，使用dynamic的方式：

>>>>>代码演示<<<<<

对于非NSObject的子类，使用泛型和闭包实现：

Observable-Swift

内存

OC:

+ (id) alloc;

-(void)dealloc;

Swift:

构造函数 (init系函数)

析构函数（deinit）

Lazy Load

>>>>>代码演示<<<<<

Lazy Load

func lazy<S : SequenceType>(s: S) -> LazySequence<S>

func lazy<S : CollectionType where S.Index : RandomAccessIndexType>(s: S) -> LazyRandomAccessCollection<S>

func lazy<S : CollectionType where S.Index : BidirectionalIndexType>(s: S) -> LazyBidirectionalCollection<S>

func lazy<S : CollectionType where S.Index : ForwardIndexType>(s: S) -> LazyForwardCollection<S>

Class & Struct

访问权限

继承性

值传递和引用传递

Method Swizzling

1、对于继承NSObject的子类，同样使用Runtime的method_exchangeImplementations实现IMP的交换，并且不能依赖load方法了，可以使用initialize代替

2、对于Swift的类，需要使用dynamic标记

三、Swift的新技巧

1、函数：多值返回、默认参数、外部参数、可变参数

2、命名空间

3、泛型

4、运算符重载

函数：多值返回、外部参数、默认参数、可变参数

>>>>>代码演示<<<<<

命名空间

嗯，可以摆脱NS、BX这类prefix了

其实Swift里面并不是真正意义上的namespace、准确一点叫做module

同一个module内的类，也不需要import

泛型

泛型是 Swift 强大特征中的其中一个，许多 Swift 标准库是通过泛型代码构建出来的

```
func swapTwoValues<T>(inout a: T, inout b: T) {
    let temporaryA = a
    a = b
    b = temporaryA
}
```

运算符重载

```
public func +(lhs: Int, rhs: Int) -> Int

public func -(lhs: Int, rhs: Int) -> Int

public func *(lhs: Int, rhs: Int) -> Int

public func /(lhs: Int, rhs: Int) -> Int
```

四、我们打算怎么用？

尝试独立的非核心模块

尝试独立的App

Thank You