1、Swift容易阅读，语法和文件结构简易化。

2、Swift更易于维护，文件分离后结构更清晰。

3、Swift更加安全，它是类型安全的语言。

4、Swift代码更少，简洁的语法，可以省去大量冗余代码

5、Swift速度更快，运算性能更高。

1、Swift的内存管理

Swift使用自动引用计数（ARC）来简化内存管理，与OC一致。

2、Swift的可选项类型（Optionals）介绍

Swift引入了可选项类型，用于处理变量值不存在的情况。Optionals类似于OC中指向nil的指针，但是适用于所有数据类型，而非仅仅局限于类，Optionals相比于OC中的nil指针，更加安全和简明，并且也是Swift诸多最强大功能的核心。

3、Swift中的 ！和  ？

这两个符号是用来标记这个变量的值是否可选，！表示可选变量必须保证转换能够成功，否则报错，但定义的变量可以直接使用；？表示可选变量即使转换不成功也不会报错，变量值为nil，如果转换成功，要使用该变量时，后面需要加！进行修饰。

4、Swift中范型的简单说明

范型是用来使代码能安全工作，swift中的范型可以在函数数据和普通数据类型中使用，例如类、结构体或枚举。范型可以解决代码复用的问题，

举个简单例子：这两个方法很类似，主要就一个参数类型的区别。

      func isIntEqual(x:Int,y:Int) -> Bool {  
         return x == y  
      }  
      func isStringEqual(x:String,y:String) -> Bool {  
        return x == y

      }

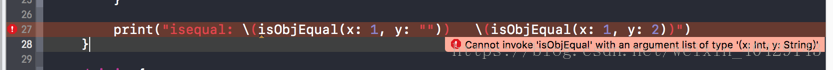
   我们可以利用范型合并一下：

func isObjEqual<T:Equatable>(x:T,y:T) -> Bool {

return x == y

}

这样，当我们使用时，我们如果将两个不同类型的数进行比较，编译器会马上提醒我们，从而快速避免这种情况。



5、Swift的访问权限变更

swift新增了两种访问权限，权限更细化。具体查看这里：

[访问权限 由大到小 依次为：open，public，internal（默认），fileprivate，private](http://www.cnblogs.com/yajunLi/p/6626509.html)

6、Swift Foundation框架

为了方便使用，Swift的基本类型都可以无缝转换到 Foundation 框架中的对应类型。

因为 Cocoa 框架所接受和返回的基本数据类型都是自身框架内的类型，也就是 Foundation 中所定义的像 NSString，NSNumber，NSArray 等这些东西。而脱离 Cocoa 框架进行 app 开发是不可能的事情。因此我们在使用 Swift 开发 app 时无法避免地需要在 Swift 类型和 Foundation 类型间进行转换。如果需要每次显式地书写转换的话，大概就没人会喜欢用 Swift 了。还好 Swift 与 Foundation 之间的类型转换是可以自动完成的，这使得通过 Swift 使用 Cocoa 时顺畅了很多。

而且这个转换不仅是自动的，而且是双向的，而且无论何时只要有可能，转换的结果会更倾向于使用 Swift 类型。也就是说，只要你不写明类型是需要 NS 开头的类型的时候，你都会得到一个 Swift 类型。

所以，Swift中的类型和OC的类型对应关系：

String - NSString

Int, Float, Double, Bool 以及其他与数字有关的类型 - NSNumber

Array - NSArray

Dictionary - NSDictionary

7、Swift便捷的函数式编程

Swift提供了Map、FlatMap、Filter、Reduce等函数方法，能够大大方便我们对对象处理。

举例最简单的场景：

Map：

var results = [1,3,5,7]

let results = values.map ({ (element) -> Int in

return element \* 2

})

//"[2, 6, 10, 14]"

Filter：

var values = [1,3,5,7,9]

let flattenResults = values.filter{ $0 % 3 == 0}

//[3, 9]

Reduce：

var values = [1,3,5]

let initialResult = 0

var reduceResult = values.reduce(initialResult, combine: { (tempResult, element) -> Int in

return tempResult + element

})

print(reduceResult)

//9

五、其他补充

========================= swift独有 ====================

1、范围运算符

a...b 表示 [a,b] 包括a和b 。 （如3...5  就是范围取3，4，5）

a..<b 表示 [a,b) 包括a，不包括b 。 （如3...5  就是范围取3，4）

常见的如for循环：for i in 0...9{}

2、独有的元组类型

元组(tuples)把多个值组合成一个复合值。元组内的值可以使任意类型,并不要求是相同类型。eg：

var value = (Int,String) = (x:15,y:"abc")

3、swift中使用let定义常量,var定义变量

使用常量,更加安全,不能够被修改,在需要对对象进行修改的时候 只能用var修饰.

4、if let 、 guard let 的用法

缩减代码量，安全处理数据逻辑。

========================= 细节使用区别 ====================

1、swift不分.h和.m文件 ，一个类只有.swift一个文件，所以整体的文件数量比起OC有一定减少。

2、swift句尾不需要分号 ，除非你想在一行中写三行代码就加分号隔开。

3、swift数据类型都会自动判断 ， 只区分变量var 和常量let

4、强制类型转换格式不同   OC强转：(int)a      Swift强转：Int(a)

5、关于BOOL类型更加严格 ，Swift不再是OC的非0就是真，而是true才是真false才是假

6、swift的 循环语句中必须加{} 就算只有一行代码也必须要加

7、swift的switch语句后面可以跟各种数据类型了 ，如Int、字符串都行，并且里面不用写break（OC好像不能字符串）

8、swift  if后的括号可以省略:  if a>b {}，而OC里 if后面必须写括号。

9、swift打印  用print("")    打印变量时可以  print("\(value)")，不用像OC那样记很多%@，d%等。

10、Swift3的【Any】可以代表任何类型的值，无论是类、枚举、结构体还是任何其他Swift类型，这个对应OC中的【id】类型。