Swift-烟花散尽俯首拾遗

大龄

1. 我们都是30岁以上
2. 我们都有近10年的IT从业经验
3. 我们都有5年以上的技术管理经验

痴迷

1. 我们痴迷技术
2. 我们崇拜乔布斯周鸿祎
3. 我们一刻不放弃技术
4. 我们为每一次技术革新而痴迷

迷茫

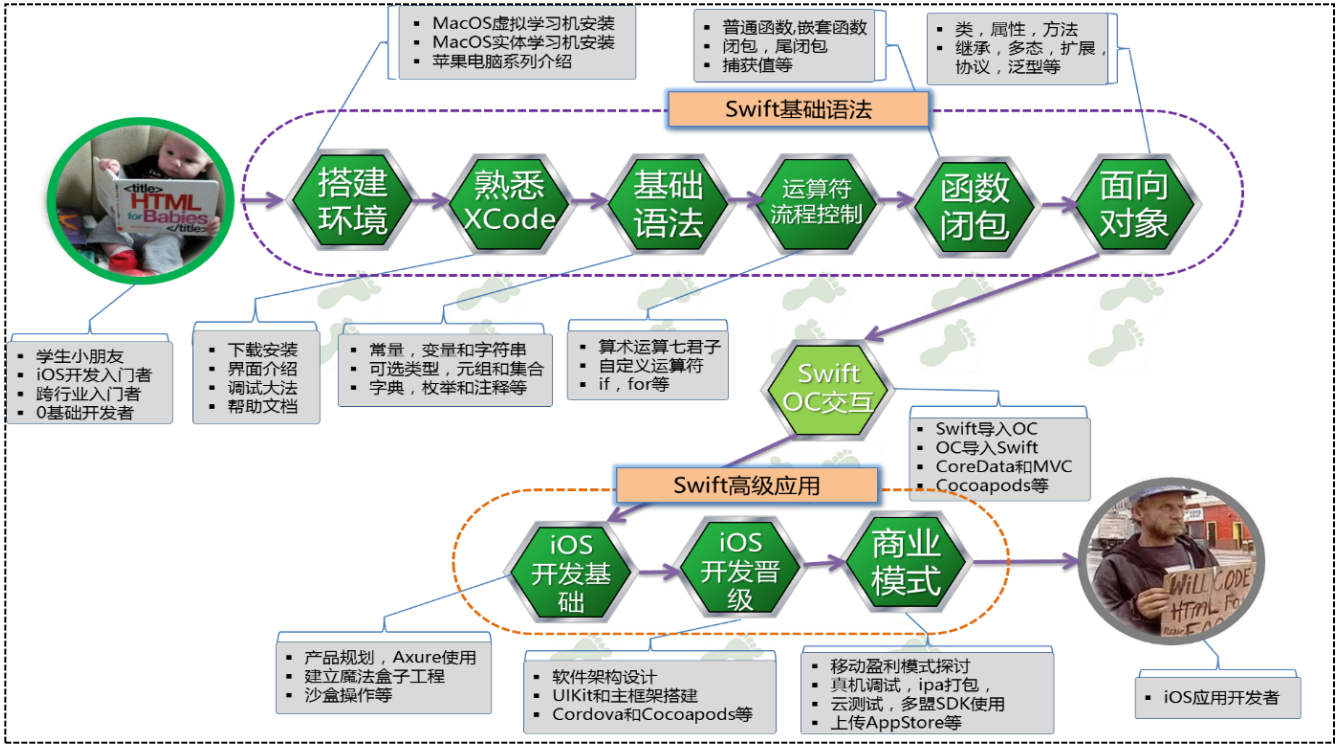
1. 我们担忧短暂的技术青春
2. 我们迷茫40岁后走向哪里
3. 我们迷茫于新型的技术
4. 我们迷茫生命在一串串代码里流失

价值

1. 我们用过各种语言
2. 我们经历过各种Crash
3. 我们有疯狂加班的洗礼
4. 我们有40万的码农兄弟做后盾

欲望

1. 我们追求IT行业的认可
2. 我们追求开源社区的持续贡献
3. 我们追求贡献技术分享的满足感
4. 我们追求未来更多的自由



价值驱动开发

勤练决定境界

看似容易实则难

因兴趣开始

因需求坚持

因生活执着

"Swift集合主流语言的特性！"
-Rust之父说

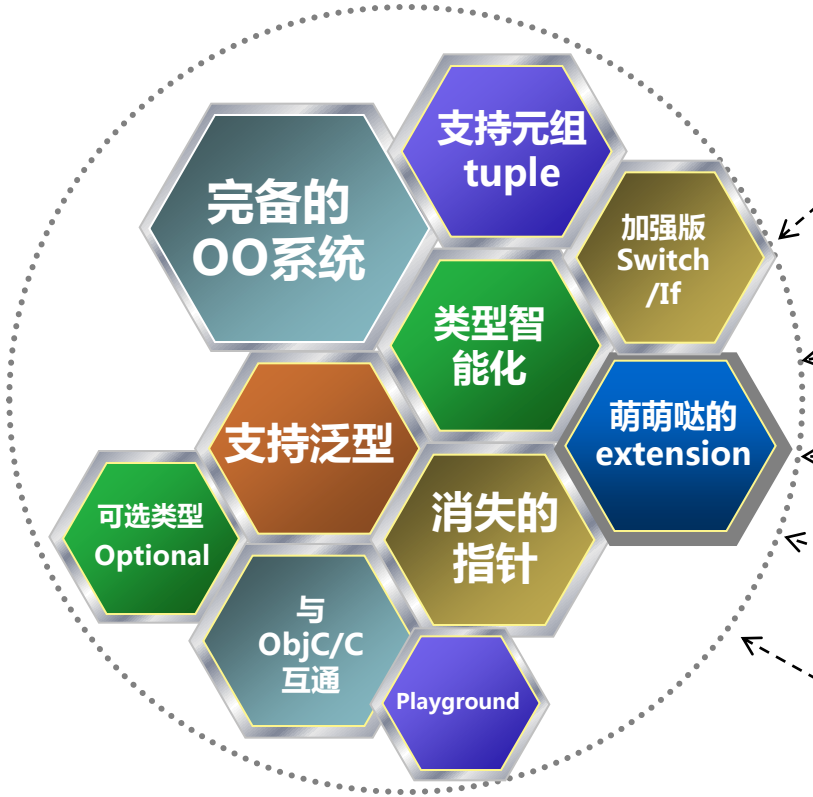
"Swift语言比ObjC顺眼多了！"
-知乎网友说

"简单容易上手，我这个**前端**狗都能懂！"
-知乎网友又说

"Swift的推出最大的受益者可能是各种培训厂商吧……大家懂的"
-知乎网友还说

"Scala的作者 —看java不爽；
Go的作者 — 看c不爽；
Swift的作者 —看OC不爽"
-知乎网友最后说

"出身名门，贵不可方物，必火！"
-老码丑哥说



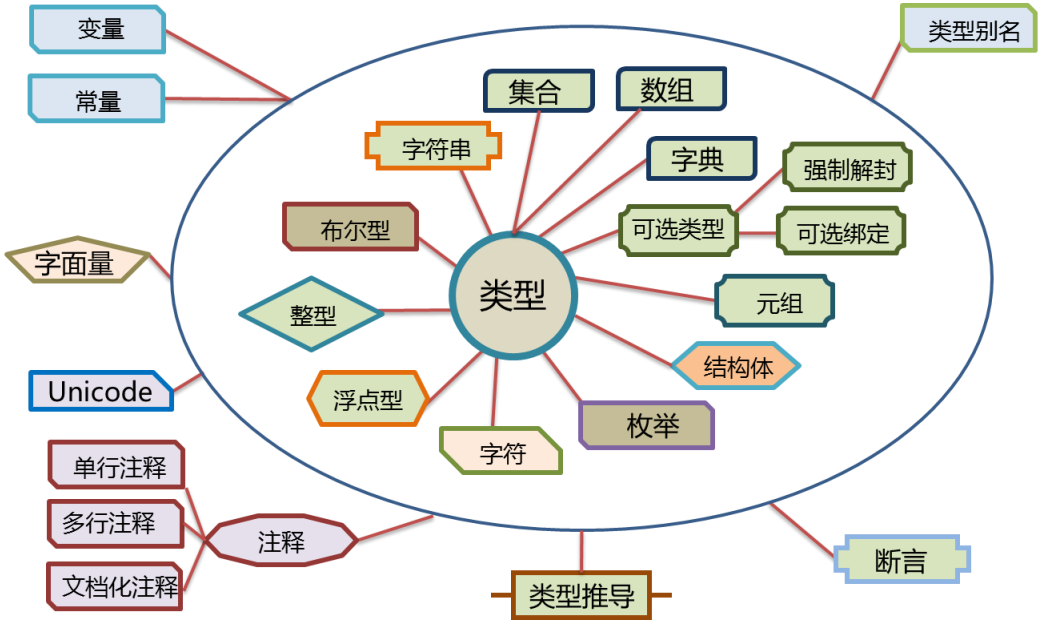
Chris Lattner

C#

GO

Ruby

Java



- 枚举类型很方便—原始值关联值相互转
- 类型推导很智能—忘却类型转换的痛
- 可选类型很温馨—？能搞定一切
- 元组类型很标配—脚本的事也能干
- 字符串型很博爱—内插特性海纳百川
- 注释也能玩高端—Doxgen已哭晕

注释也能玩高端

mai...wift › helloOldCoder(_:age:)

```swift
import Foundation

//TODO: 产品经理让你增加功能
    /**
    这是一个Swift的函数，将在第五章深入讲解
    看不懂也不用忧伤！这个函数的功能如下：

    -函数名
    -函数参数和返回值

        + 只有一个返回值
        * 并且返回值是true

    1. Swift的函数必须以func开头
    2. Swift的参数需要以:参数名:类型 的方式定义
    3. Swift的返回值则是函数参数结束后以:->类型 的方式定义
*/
func helloOldCoder( name: String, age: Int )->Bool {
    print( "老码是一群快乐的码农！" )
    return true;
}
```

Quick Help

Declaration `func helloOldCoder(name: String, age: Int) -> Bool`

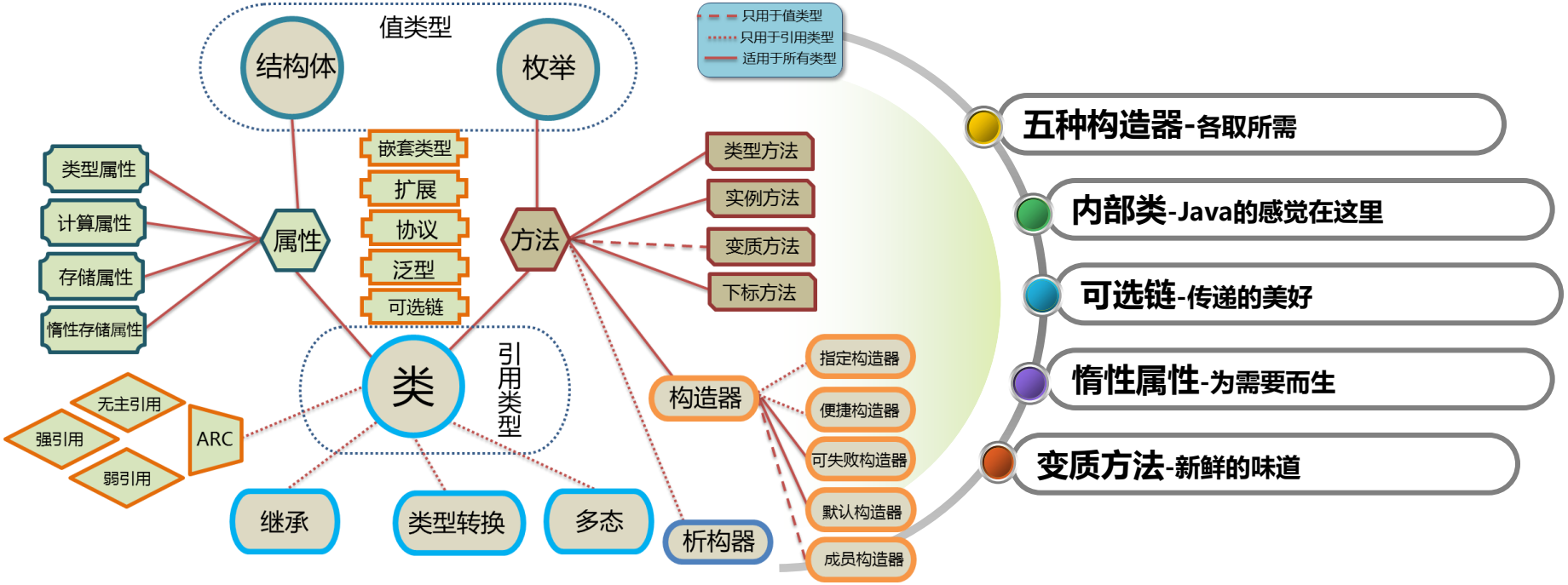
Description 这是一个Swift的函数，将在第五章深入讲解 看不懂也不用忧伤！这个函数的功能如下：

-函数名 -函数参数和返回值

```
+ 只有一个返回值
* 并且返回值是true
```

1. Swift的函数必须以func开头
2. Swift的参数需要以:参数名:类型 的方式定义
3. Swift的返回值则是函数参数结束后以:->类型 的方式定义

Declared In main.swift


值类型
结构体
枚举
只用于值类型
只用于引用类型
适用于所有类型
类型属性
计算属性
存储属性
惰性存储属性
属性
嵌套类型
扩展
协议
泛型
可选链
方法
类型方法
实例方法
变质方法
下标方法
类
引用类型
无主引用
强引用
弱引用
ARC
构造器
指定构造器
便捷构造器
可失败构造器
默认构造器
成员构造器
继承
类型转换
多态
析构器
五种构造器-各取所需
内部类-Java的感觉在这里
可选链-传递的美好
惰性属性-为需要而生
变质方法-新鲜的味道

五种构造器

```
// 定义蓝翔收费标准的枚举类型-注意"可失败构造器"在枚举类型中的使用
enum LanXiangFee{
    case XiJianChui
    case WaJueJi

    // 这是可失败构造器，表示在钱不够时返回nil以表示实例构造失败
    init?( money:Int){
        if money >= 30000 && money < 50000 {
            self = .XiJianChui
        }else if money >= 50000{
            self = .WaJueJi
        }else{
            print( "拿着\(money)就想上蓝翔，你对得起唐国强老师吹下的牛吗！")
            // 在五大类构造器中只有可失败构造器可以返回值，且返回值只能为nil
            return nil
        }}
}
// 老码打算改行去学洗剪吹，爸爸妈妈只给了250RMB(使用结构体的成员造器生成了一个Student的实例oldCoder)
var oldCoder = Student(name: "老码", field: "洗剪吹", money: 250)
// 路上遇到一个蓝翔学挖掘机的学员路人甲(使用结构体的默认生成了一个Student实例student1)
var student1 = Student()
// 找到了蓝翔技校(使用类的便捷构造器生成lanXiangTechnicalSchool实例)
var lanXiangTechnicalSchool = TechnicalSchool()
// 找到了老罗技校(使用类的指定构造器生成laoLuoTechnicalSchool实例)
var laoLuoTechnicalSchool = TechnicalSchool(name: "北京老罗技校", location: "北京",
    topField: "锤子挖掘机",advertisement: "学好挖掘机，还得靠三倍情怀！")
var feeForOldCoder = LanXiangFee( money: oldCoder.money)
if feeForOldCoder == nil {
    print("蓝翔拒绝了你，请找老罗技校，亲！")
    laoLuoTechnicalSchool.receive(oldCoder)
}else{
    lanXiangTechnicalSchool.receive(oldCoder)
}
// 因为3个身份证的问题，蓝翔技校可能被关闭(使用可失败构造器触发实例构造失败)
var lanXiangTechnicalSchoolNow  = TechnicalSchool(has3IDCard: true)
if lanXiangTechnicalSchoolNow == nil {
    print( "无法生成lanXiangTechnicalSchoolNow实例" )
}
// 谣言已过，蓝翔技校开张(使用可失败构造器保证实例构造成功)
lanXiangTechnicalSchoolNow  = TechnicalSchool(isRumor:  true)
if lanXiangTechnicalSchoolNow != nil {
    print( "成功生成了lanXiangTechnicalSchoolNow实例" )
}
```

```
"拿着250就想上蓝翔，你对得起唐国强老师吹下的牛吗！\n"

Student

Student

TechnicalSchool

TechnicalSchool

nil

"蓝翔拒绝了你，请找老罗技校，亲！\n"
TechnicalSchool

nil

"无法生成lanXiangTechnicalSchoolNow实例\n"

TechnicalSchool

"成功生成了lanXiangTechnicalSchoolNow实例\n"
```

内部类

```swift
import Foundation

// 花中的世界
class FlowerWorld {
    // 估计花中世界也有自己的上帝存在
    class var God : String {
        return "花仙子"
    }

    // 花中世界也有居民
    class Resident {
        // 居民也有孩子，嵌套类
        class Children {
            var name:String = ""
        }

        var name:String
        var gender: String
        var children: Children?
        var type:ResidentType

        init( name: String,gender:String, type:ResidentType) {
            self.name = name
            self.gender = gender
            self.type = type
        }
    }
    // 有居民必然有地位划分
    enum ResidentType:Int {
        case Worker = 1
        case Farmer
        case Officer
    }

    // 花的世界也有空气
    struct Air {
        var name:String
        var amount:String
    }

    // 以下是花中世界的方法和属性
    var air = Air( name: "氧气", amount: "无限")
    var residents = [Resident]()

    // 人类就是花中世界的造物主，推一下，让它动起来
    func run() {
        let residents1 = Resident( name: "亚当", gender: "男",
            type: .Farmer)
        let residents2 = Resident( name: "夏娃", gender: "女",
            type: .Farmer)
        residents.append( residents1 )
        residents.append( residents2 )
        let child = Resident.Children()
        child.name = "花世界公民的祖先"
        residents1.children = child
        residents2.children = child
    }

    // 如果我们可以和花中世界对话，它们估计会说
    func say() {
        print( "我们是花中世界，我们的上帝是:\(FlowerWorld.God) \n我们呼吸\( air.
            name ) 存量 \(air.amount ) \n这里有 \(residents.count)位先民\n他们
            的名字是 \(residents[0].name )和\(residents[1].name)\n他们孕育了一
            个孩子,名字叫 \(residents[0].children!.name )")
    }

}

// 人类培育一朵鲜花，也就是培育一个世界
var flowerWorld = FlowerWorld()
// 花的世界开始演化
flowerWorld.run()
// 和花中世界的人对话
flowerWorld.say()
// 我们依然可以定义一个Children类类型，虽然它和FlowerWorld.Resident类中的Children
    类型同名，但是依然可以使用
class Children {
    var name:String = ""
}
```

惰性属性

```swift
//
//  main.swift
//  Code6-2-3-2-1
//
//  Created by OldCoder on 15/5/25.
//  Copyright (c) 2015年 OldCoder. All rights reserved.
//

import Foundation

class Buddha {
    var name: String
    init( name: String) {
        self.name = name
    }
}


class People {
    var dream: String?
    var reality: String = ""
    var lover: People?

    lazy var buddha = Buddha(name: "释迦牟尼") // 懒惰存储属性

    func pray()      // 向佛祈祷，寻求帮助
    {
        print( "佛祖的法号:\(buddha.name)" )
    }
}

let youngCoder = People()        // 小码农华丽出场
youngCoder.pray()              // 作业完不成，只能向佛祈祷
```

```
Code6-2-3-2-1 〉Thread 1 〉0 main
(lldb) print youngCoder
(Code6_2_3_2_1.People) $R0 = 0x0000000100703890 {
  dream = nil
  reality = ""
  lover = nil
  buddha.storage = nil
}
(lldb)
```

```
Code6-2-3-2-1 〉Thread 1 〉0 main
佛祖的法号:释迦牟尼
(lldb) print youngCoder
(Code6_2_3_2_1.People) $R0 = 0x0000000101600260 {
  dream = nil
  reality = ""
  lover = nil
  buddha.storage = (name = "释迦牟尼") {
    name = "释迦牟尼"
  }
}
(lldb)
```

变质方法

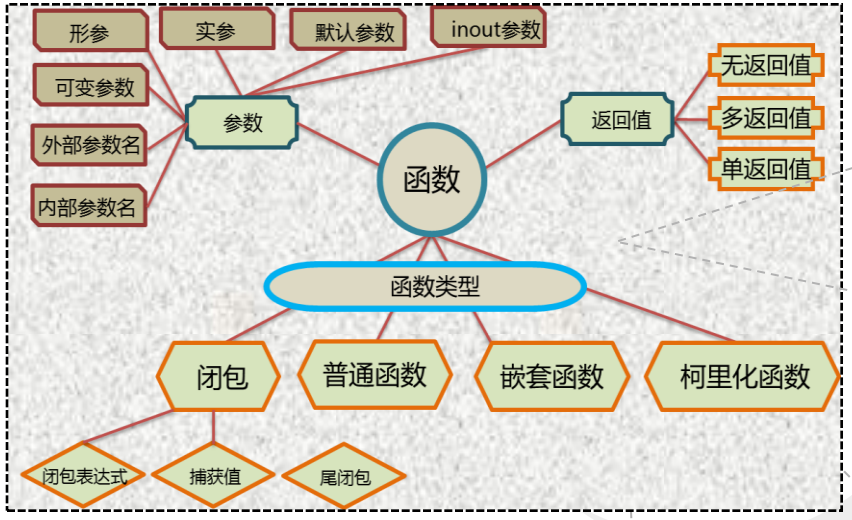
```
1  import Foundation
2
3  struct Warcraft{
4      var scene: String
5      var money: Int
6
7      mutating func jumpToSense( scene: String )
8      {
9          // 直接生成一个新的Warcraft实例，而不是单纯地修改scene和money属性
10         self = Warcraft( scene: scene, money: 100)
11     }
12 }
13
14 // 新的游戏，新的场景，且金币为200
15 var myWarcraft = Warcraft(scene: "无间地狱", money: 200)
16 // 疯狂练级，赚了1000金币
17 myWarcraft.money = 1000
18 print("当前场景: \( myWarcraft.scene ) 当前金钱:\(myWarcraft.money)")
19 // 水平太差，死掉，恢复到魔域之火场景，金币丢失
20 myWarcraft.jumpToSense("魔域之火")
21 print("当前场景: \( myWarcraft.scene ) 当前金钱:\(myWarcraft.money)")
22
```

```
Warcraft

Warcraft

Warcraft
"当前场景: 无间地狱 当前金钱:1000\n"

Warcraft
"当前场景: 魔域之火 当前金钱:100\n"
```



闭包
高端难用

多返回
元组的魅力

嵌套函数
jQuery的味道

柯里化
一半一半
再一半

闭包

```
1  import Foundation
2
3  func makeIncrementor(forIncrement amount: Int) -> () -> Int {
4      var runningTotal = 0
5      func incrementor() -> Int {
6          runningTotal += amount
7          return runningTotal
8      }
9      return incrementor
10 }
11
12 let incrementByTen = makeIncrementor(forIncrement: 10)
13 incrementByTen()  // 返回值是10
14 incrementByTen()  // 返回值是20
15 incrementByTen()  // 返回值是30
16
17 let incrementBySeven = makeIncrementor(forIncrement: 7)
18 incrementBySeven()  // 返回值是7
19 incrementByTen()    // 返回值是40
20
```

(2 times)

(5 times)

10
20
30
7
40

(5 times)

(2 times)

() -> Int

() -> Int
10
20
30

() -> Int
7
40

■ 嵌套函数

```
11 func chooseBankingBusiness (bankingBusiness: String) -> (String)->Bool {
12     // drawMoney成为嵌套函数
13     func drawMoney(cardNumber: String)-> Bool {
14         // 这里实现具体的取款业务
15     }
16     // saveMoney成为嵌套函数
17     func saveMoney(cardNumber: String) -> Bool {
18         // 这里实现存款的业务
19     }
20     var bankingBusinessFunc = drawMoney
21         if bankingBusiness == "取款" {
22             bankingBusinessFunc = drawMoney
23         }
24         else if bankingBusiness == "存款" {
25             bankingBusinessFunc = saveMoney
26         }
27     return bankingBusinessFunc
28 }
29
30
```

柯里化函数

```
1  //: Playground - noun: a place where people can play
2
3  import Foundation
4
5  // 定义普通形式的salary函数
6  func salary( name:String, base:Int, ot: Int)->Bool{
7      print("\(name) 收到工资:\(base) 加班费:\(ot)")
8      return true
9  }
10
11 // 定义柯里化形式的salary函数
12 func salary(name:String)(base:Int)(ot:Int)->Bool{
13     print("\(name) 收到工资:\(base) 加班费:\(ot)")
14     return true
15 }
16
17 // 使用类型别名重命名salay函数的两种定义形式
18 typealias PaySalaryTypeCurried = Int->Int->Bool
19
20 // 使用salary的柯里化形式->注意柯里化函数的类型
21 func success( type:String, salary: Int->Int->Bool){
22     if type == "砖工"{
23         salary(250)(250)
24     }else{
25         salary(100)(100)
26     }
27 }
28
29 // 失败回调函数
30 func fail( reason:String)
31 {
32     print( "真跳了！")
33 }
34
35 // 老码和工头讨薪的函数，讨薪的成本在于遇到了谁，如果是良心企业就成功
36 func begSalaryByForeman( who:String, success:(String, PaySalaryTypeCurried)->(), fail:(String)->()){
37     if who == "良心企业"{
38         print("遇到良心企业，讨薪成功，发工资了:")
39         success("砖工", salary("老码"))
40         success("泥工", salary("老罗") )
41     }else{
42         fail("遇到无良企业，讨薪失败，无奈跳楼！")
43     }
44 }
45
46 begSalaryByForeman("良心企业", success: success, fail: fail)
```

```
(2 times)
(2 times)

true

true

"遇到良心企业，讨薪成功，发工资了:\n"
```

Alamofire: Swift版本的HTTP网络库，可代替OC版的AFNetworking

CORDOVA: 开发Web App或者Hyber App的流行框架

CocoaPods: Swift开发中必用的类库管理工具

RxSwift: Swift响应式开发框架，

BeeFramework: 支持HTML/CSS渲染的OC框架。

SwiftyJson: 简单易用且方便的Json库

NSHipster: 收录国外开发者对于ObjC，Swift研究学习的文章

Reveal: 界面调试利器

Swift开源

- Swift开源-跨平台的顾虑消失
- 适应多变的业务系统
- Swift易用的特性引来各平台的粉丝
- 开发的生态圈由工匠向工人转变

类库完善

- 活跃的社区
- 琳琅满目的工具集
- 丰富的问题解决系统
- 苹果的态度
- Swift语言的稳定

公司转型

- 人才成本出现倾斜
- 大公司导向性作用
- 公司人才战略调整
- Swift商业化成熟
- 技术演进作用

Competence储备

- 大量的新生代开发者加入
- 更多的培训机构介入
- 更多的公司参与
- ObjC人才涨势放缓
- 商业价值凸显

谢谢！

@未来眼之老码团队
http://weibo.com/u/5241713117

@老码团队邮箱
oldcoderteam@163.com

@老码团队
18602803782

@Swift官方文档
http://numbbbbb.gitbooks.io/-the-swift-programming-language-/