uml.org.cn 文章 文库 视频 讲座 课程 咨询 认证

随时听讲座
每天看新文
追随技术信仰

火龙果讲堂：

- 一线专家
- 案例回顾
- 经验分享

Swift 编程实践，最佳

宜信 孟祥月

安排

- Swift实践
- 案例讲解：PhotoPicker
  - 与自动布局的结合
- 交流时间

目录

- 简单介绍一下我学习Swift的过程
- 我的Swift编程实践
- 拥抱Swift

我学习Swift的过程

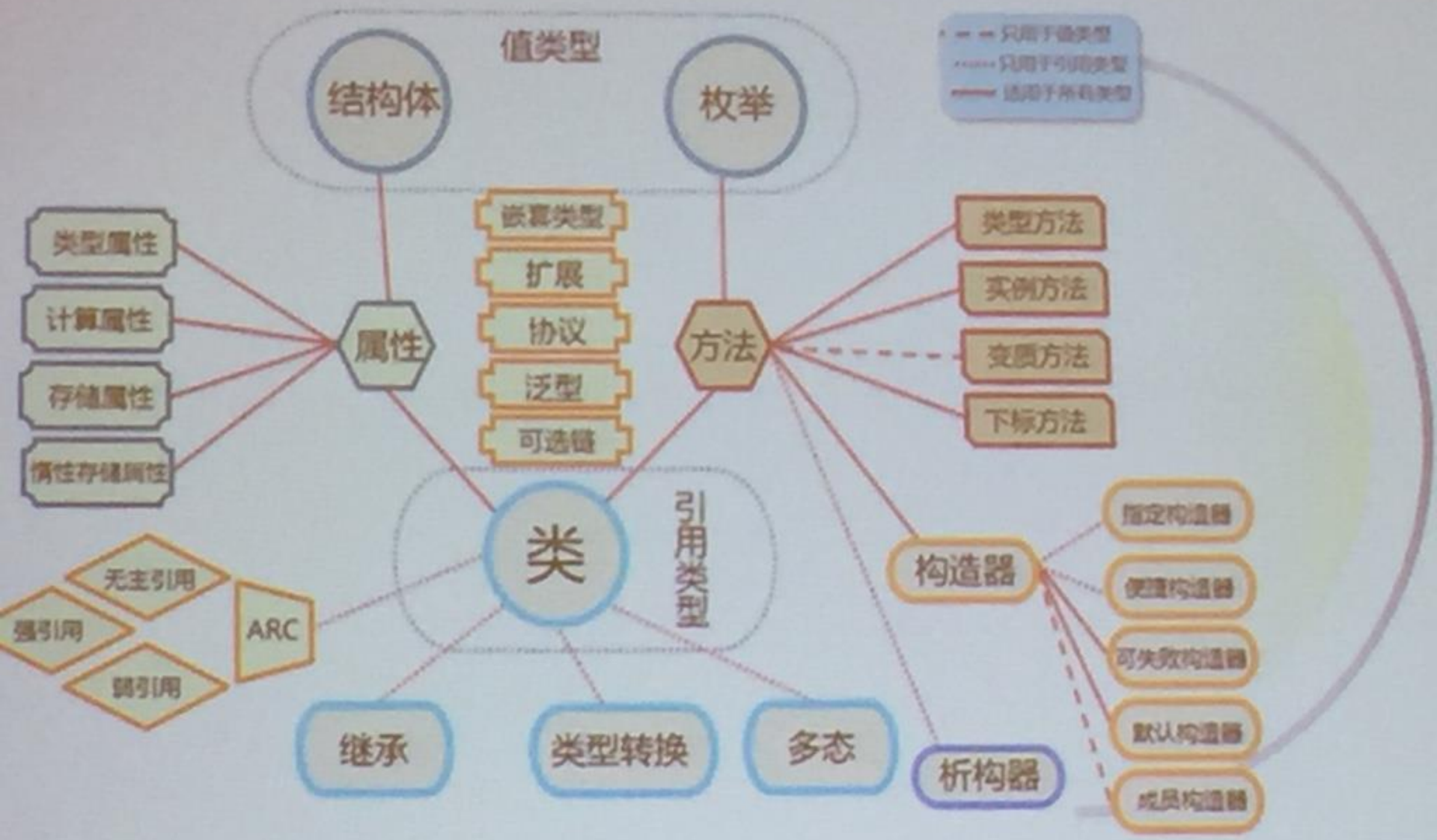
- WWDC 2014 发布Swift
- WWDC 2015 发布Swift 2.0 并宣布年底开源
- The-Swift-2.0-Programming-Language-playground（https://github.com/mengxiangyue/The-Swift-2.0-Programming-Language-playground）

我的Swift编程实践

Swift特点

- Swift. A modern programming language that is safe, fast, and interactive.
- 无缝调用Objective-C、C API
- 提高编程的效率

变量
常量
字面量
Unicode
单行注释
多行注释
文档化注释
注释
类型推导
断言
类型别名
强制解封
可选绑定
可选类型
元组
结构体
枚举
字符
浮点型
整型
布尔型
字符串
集合
数组
字典
类型


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
变异方法
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

简洁的语法

- 分号
- 类型推断
- if 括号
- switch break
- API

```
import UIKit

let ary = [3, 5, 1, 16, 7]                                  [3, 5, 1, 16, 7]
//let sortAry = ary.sort({ (a: Int, b: Int) -> Bool in
//    return a < b
//})
let sortAry = ary.sort(<)                                   [1, 3, 5, 7, 16]
```

基本类型全部是值类型


```
public struct Int
public struct Float
public struct  Double
public struct Array<Element>
public struct Dictionary<Key : Hashable, Value>
```

```
var arrayA = [1, 2]
var arrayB = arrayA
arrayB.append(3)

arrayA
arrayB
```

```
[1, 2]
[1, 2]
[1, 2, 3]

[1, 2]
[1, 2, 3]
```

```
var arrayA: NSMutableArray = [1, 2]
var arrayB: NSMutableArray = arrayA
arrayB.addObject(3)

arrayA
arrayB
```

```
[1, 2]
[1, 2]
[1, 2, 3]

[1, 2, 3]
[1, 2, 3]
```

强大的Enum

- 可以是Int之外的其他类型
- 可以关联某些值

```
enum ProductType: String {
    case Bicycle
    case Car
    case Bus
}

let product = ProductType(rawValue: "Car")
```

Car

```
enum ProductType {
    case Bicycle
    case Car(seatCount: Int)
    case Bus(seatCount: Int, wheelCount: Int)
}

let car = ProductType.Car(seatCount: 4)

switch car {
case .Car(let seatCount):
    print("the car has \(seatCount) seats")
default:
    break
}

if case .Car(let seatCount) = car {
    print("the car has \(seatCount) seats")
}
```

```
Car(4)

"the car has 4 seats\n"

"the car has 4 seats\n"
```

元组、Switch

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

```
let scoreArray = [("A", 90), ("B", 75), ("C", 65), ("D", 55)]
scoreArray.forEach { (score: (String, Int)) -> () in
    switch score {
    case (let name, let number) where number >= 60:
        print("\(name) 及格了")
    case (let name, let number) where number < 60:
        print("\(name) 不及格")
    default:
        break
    }
}
/*
A 及格了
B 及格了
C 及格了
D 不及格
*/
```

Optional

- 使用可选类型（optionals）来处理值可能缺失的情况。可选类型表示：有值，等于 x ；或者没有值

```
var serverResponseCode: Int? = 404
// serverResponseCode 包含一个可选的 Int 值 404
serverResponseCode = nil
// serverResponseCode 现在不包含值
```

函数

- 一等公民
- 参数和返回值
- 函数类型
  - (参数类型列表) -> 返回值类型
    - (Int, Int) -> Int

简单的单例


```
class Manager {
    static let sharedInstance = Manager()
    private init() {

    }
}
```

牛逼的下标


```
class Task: Hashable {
    var taskId: Int
    init(taskId: Int) {
        self.taskId = taskId
    }
    var hashValue: Int {
        return taskId
    }
}

func ==(left: Task, right: Task) -> Bool {
    return left.taskId == right.taskId
}

class Result {
}

class TaskQueue {
    var queue = [Task: Result]()
    subscript(task: Task) -> Result{
        get {
            return queue[task]!
        }
        set {
            queue[task] = Result()
        }
    }
}
```

灵活的extension

- 协议扩展 类似虚函数
- 使用extension组织代码

```
protocol Runable {
    func run()
}

extension Runable {
    func run() {
        print("Runable protocol default implement")
    }
}

class People: Runable {
}

let people = People()
people.run() // Runable protocol default implement
```

```
class MyViewController: UIViewController {

}

extension MyViewController: UITableViewDataSource {
    func tableView(tableView: UITableView, numberOfRowsInSection section: Int) -> Int {
        return 2
    }
    func tableView(tableView: UITableView, cellForRowAtIndexPath indexPath: NSIndexPath) ->
        UITableViewCell {
        return UITableViewCell()
    }
}
```

运算符重载


```
func +(left: CGSize, right: CGSize) -> CGSize {
    return CGSize(width: left.width + right.width, height: left.height + right.height)
}
```

详细介绍：http://blog.csdn.net/mengxiangyue/article/details/43437797

Swift脚本

- Carthage
- Epoch
  - https://github.com/Zewo/Epoch
- 2015年底开源，开源之后，会有更多想象

REPL

- 交互式解释器

下面的截图运行与Mac终端：

```
bogon:~ xiangyue$ xcrun swift
Welcome to Apple Swift version 2.1 (700.1.10
ance.
  1> var x = 2
x: Int = 2
  2> let y = x + 3
y: Int = 5
  3> x + y
$R0: Int = 7
  4>
```

拥抱Swift

你是否应该选择 Swift？

- Swift还在高速变化
- 编译器不是十分稳定（代码提示，增量编译等）
- 与OC相比更少的第三方库
- 编程习惯的一些转变

Swift Now

- Apple 全力推行
- 语言层面更加丰富、现代
- 完全兼容 Cocoa、Cocoa Touch
- 直接支持系统底层的 C api
- 最低支持 iOS 7
- 更加稳定的Xcode
- 越来越多的开发者和第三方库

案例 PhotoPicker

项目地址：https://github.com/mengxiangyue/PhotoPicker

听众参与

- 问题
  - swift在未来会取代OC么
  - swift编程相比OC 简单了 还是更难了?
  - iOS项目工作经验在面试中的比重有多大?
- 留给听众的问题
  - 反射 Mirror方面的知识
  - 面向协议编程

交流时间