Why Scala

吴雪峰@Thoughtworks

2014年6月8日

benewu@gmail.com

议程

- 为什么要静态编译和强类型
- 为什么要动态执行和弱类型
- 为什么要面向对象
- 为什么要函数式编程

- 什么是Scala

- 现在能用Scala吗?

为什么要静态编译和强类型

- 编译时检查，类型安全，更早知道问题
- 工具提示
- 性能好
- C++, Java/C#

为什么要动态执行和弱类型

- 轻量,灵活
- 运行时可改变程序行为
- Python/Ruby, JavaScript

为什么要面向对象

“所有人都在面向对象”

–Johnny Appleseed

面向对象三大特征 和 要解决的三个问题

封装

可维护

继承

可重用

多态

可扩展

为什么要函数式编程

“λ”

–Johnny Appleseed

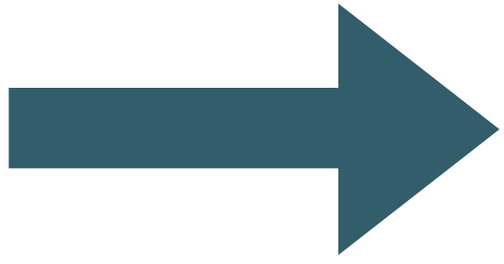
λ演算的函数可以接受函数当作输入（参数）和输出（返回值）

函数式编程特性和软件开发要解决的问题

无副作用

引用透明

高阶函数



可维护

可重用

可扩展

“那为什么函数式编程没有流行起来?”

–Johnny Appleseed

图灵完备

- 在可计算性理论里，如果一系列操作数据的规则（如指令集、编程语言、细胞自动机）可以用来模拟单带图灵机，那么它是图灵完备的
- 可计算的都可计算

历史真想

艾伦·图灵

老师

阿隆佐·邱奇

可计算性 - 青出于蓝

- 可计算性理论（Computability theory）作为计算理论的一个分支，研究在不同的计算模型下哪些算法问题能够被解决

- 阿隆佐·邱奇: λ演算 - 计算资源无限，递归

- 艾伦 图灵: 单带图灵机 - 机械实现

邱奇图灵论题

那为什么函数式编程没有流行

- **物理限制**
- 计算资源有限
- 内存有限

CPU和内存资源级数增长

FP v.s. OO

- 可重用单位: 类 v.s. 方法
- 可维护: 把信息封装起来 v.s. 信息不会发生变化
- 可扩展: 类组合 v.s. 方法组合

在一起

动态执行

JavaScript

Ruby

Python

Swift

Java 8

Clojure

C#

面向对象

函数式

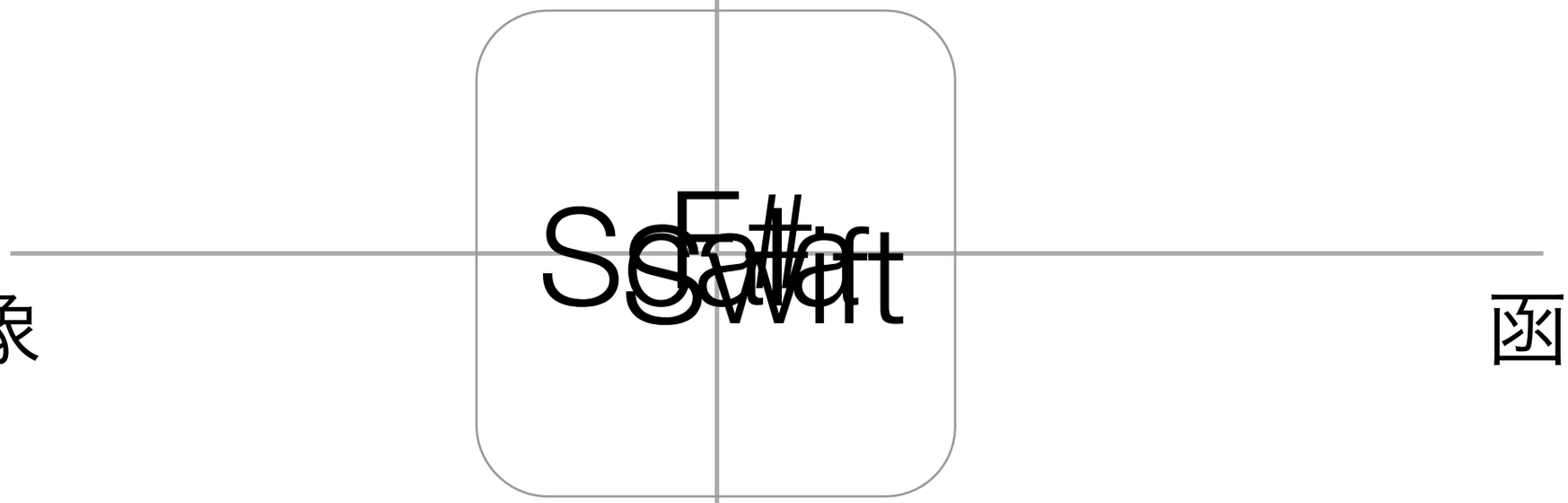
Object-C

Java

静态编译

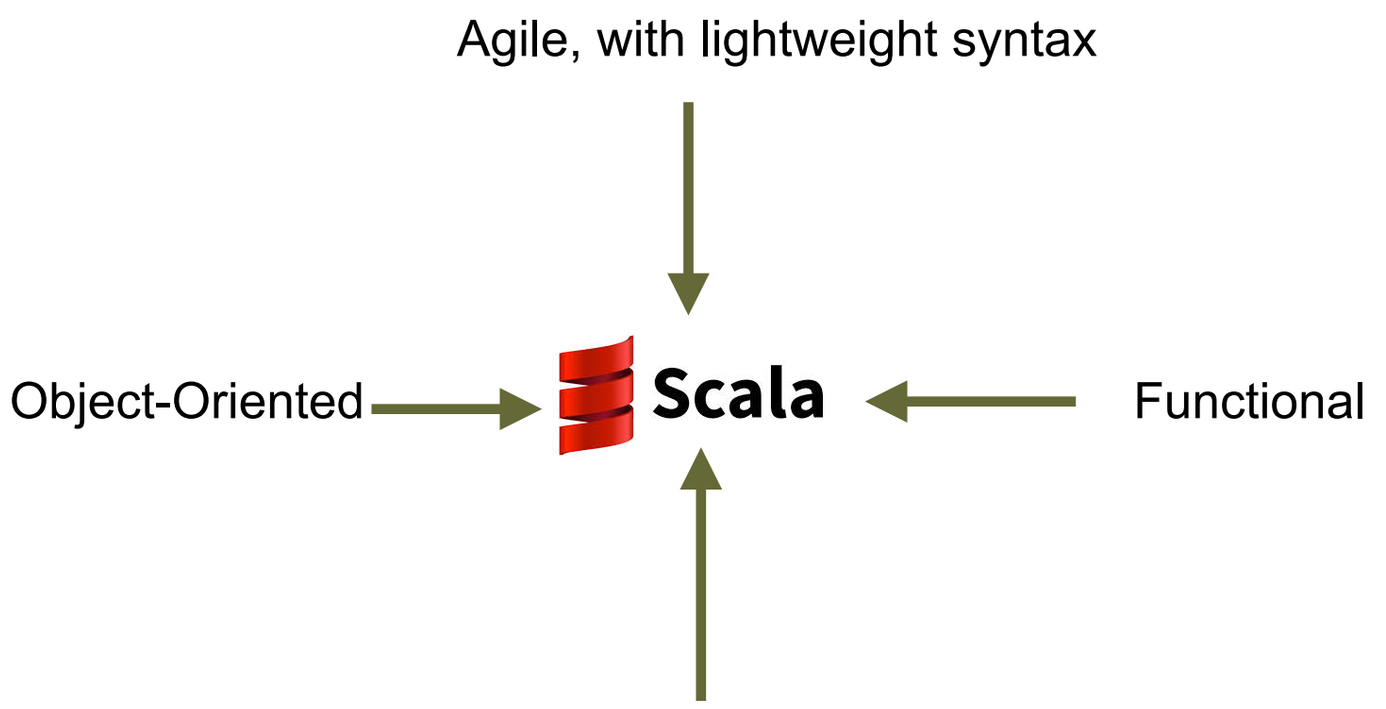
动态执行

面向对象



函数式

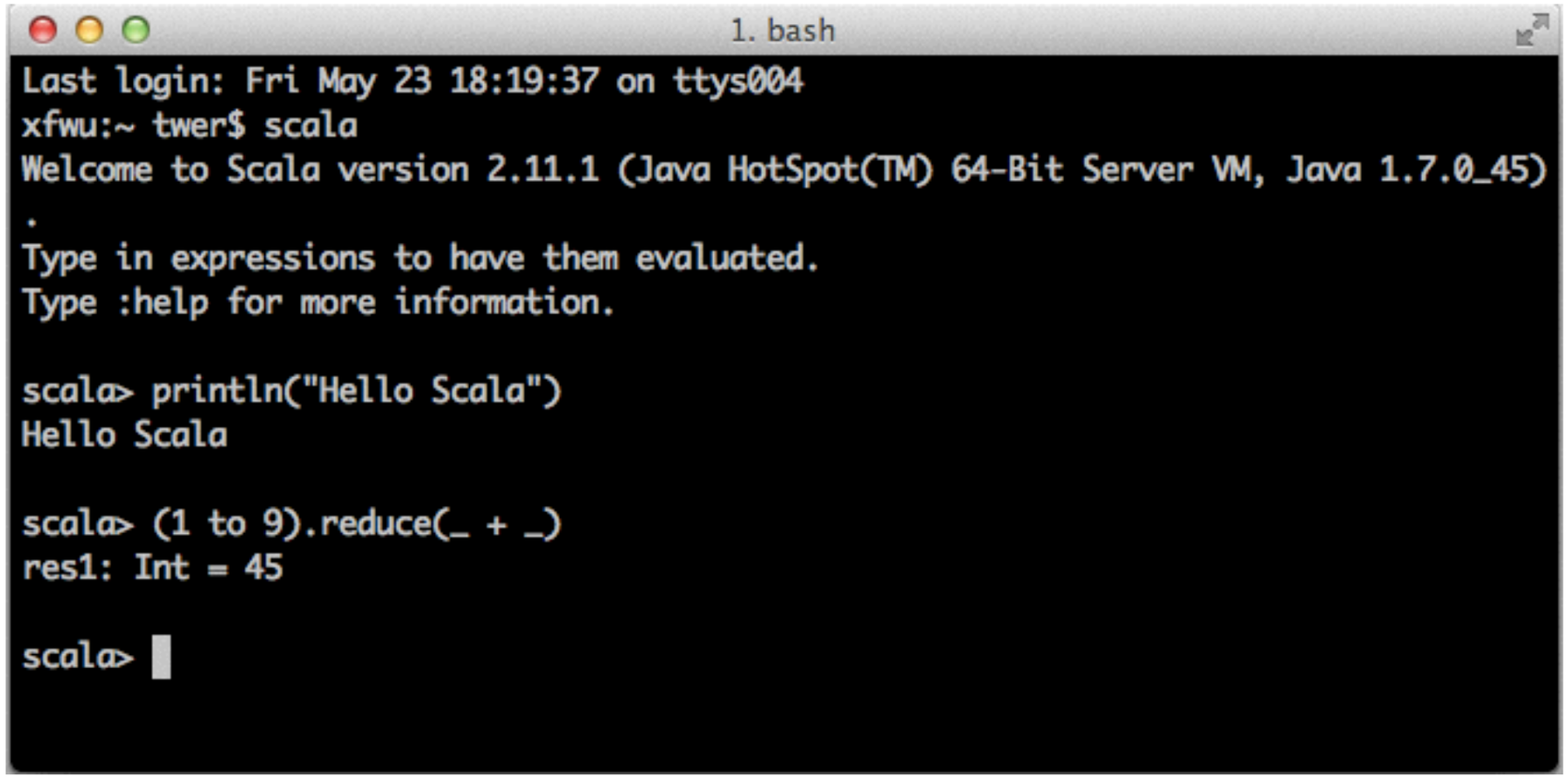
静态编译



Scala 是个混合体

Scala能做好吗

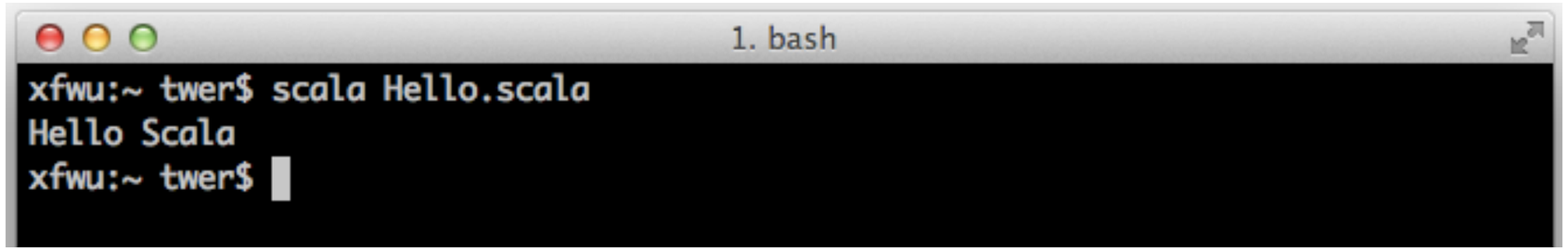
动态性


1. bash
Last login: Fri May 23 18:19:37 on ttys004
xfwu:~ twer$ scala
Welcome to Scala version 2.11.1 (Java HotSpot(TM) 64-Bit Server VM, Java 1.7.0_45)
.
Type in expressions to have them evaluated.
Type :help for more information.
scala> println("Hello Scala")
Hello Scala
scala> (1 to 9).reduce(_ + _)
res1: Int = 45
scala>

```scala
object Hello {

  def main(args: Array[String]): Unit = {
    println("Hello Scala")
  }

}
```

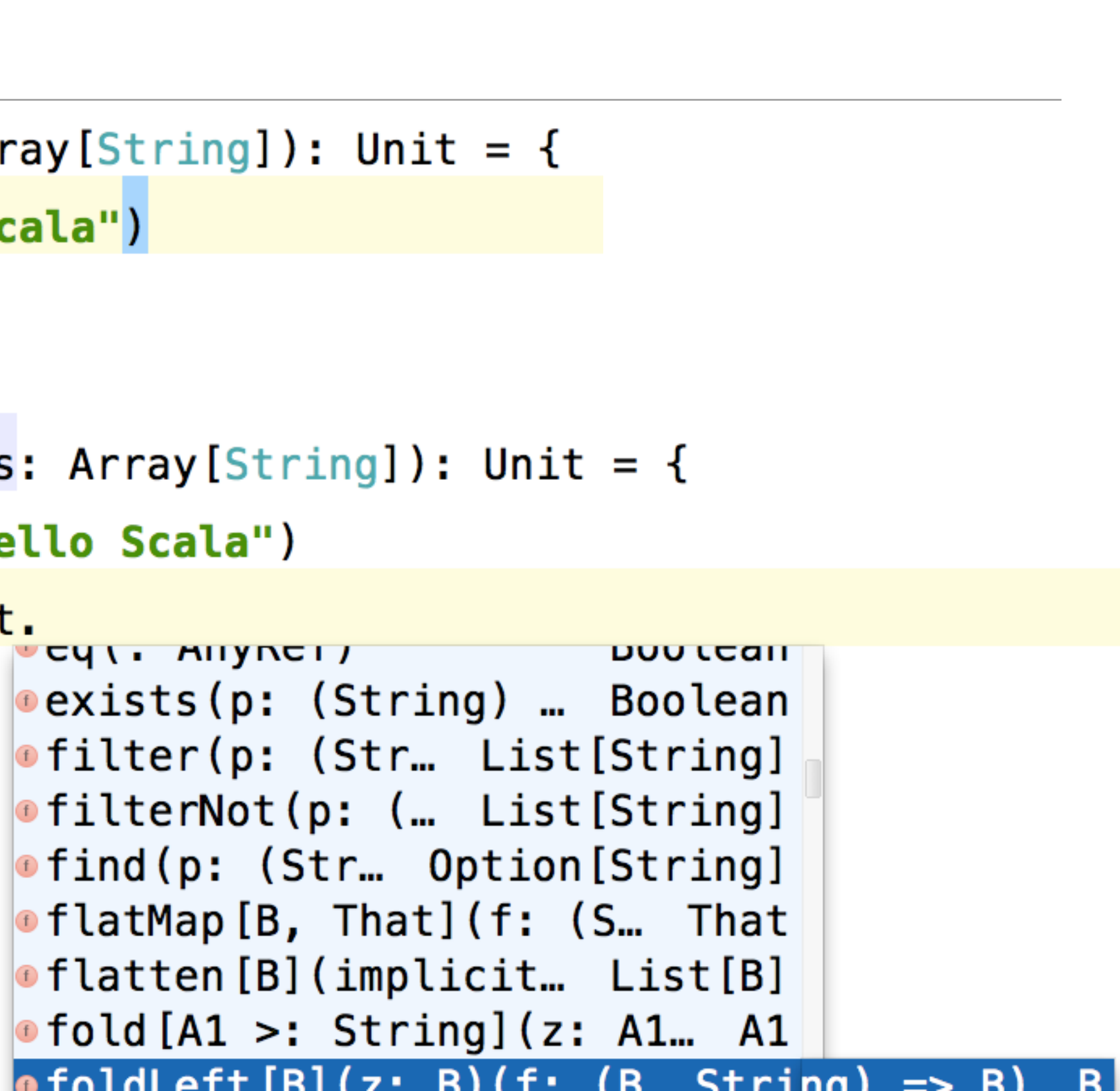


静态编译

```
def main(args: Array[String]): Unit = {
  printl("Hello Scala")
}  object Hello {

      def main(args: Array[String]): Unit = {
  println("Hello Scala")
  args.toList.
  }

}
```



强类型系统

```scala
object Hello {

  def main(args: Array[String]): Unit = {
    println(echo(Some("Shanghai")))
    println(echo(None))
    println(echo("Beijing"))
  }

  def echo(name: Option[String]): String = {
    s"""Hello ${name.getOrElse("Scala")}"""
  }

}
```

Type mismatch, expected: Option[String], actual: String

面向对象

封装　　继承　　多态

```
case class Person(name: String, age: Int)
```

```
val xuefeng = Person("Xuefeng", 18)
println(xuefeng.name)
```

```
trait Fruit {
   val price: Double
   val weight: Double
   def amount = price * weight
 }
 case class Apple(weight: Double, price: Double) extends Fruit
```

```
val hongfushi = Apple(1.0,15.00)
 println(hongfushi.amount)
```

```
case class Watermelon(weight: Double, price: Double,rate:Double) extends Fruit {
   override def amount = price * weight * rate
 }
```

```
val nanhui8424 = Watermelon(1.0,15.00,0.9)
      println(nanhui8424.amount)
```

函数式

函数是一等公民

可以独立定义

```
def add10(x: Int) = x + 10
```

可以作为函数参数

```
val add10Result = (1 to 10).map(x => add10(x))
println(add10Result)
```

可以作为函数返回值

```
def add(x: Int): (Int => Int) = y =>  x + y
val add10 = add(10)
add10(9)
```

议程

- 为什么要静态编译和强类型
- 为什么要动态执行和弱类型
- 为什么要面向对象
- 为什么要函数式编程

- 什么是Scala
- 现在能用Scala吗?

How many FP
people see OOP

How many OOP
people see FP

And that’s where we are ☺

常见抱怨

- 学习曲线太陡峭
- 类型系统不严谨
- 编译太慢，工具不成熟
- 很多语法糖和魔法符号
- 语法太灵活

10 Years of Scala

Subject: **ANNOUNCEMENT: The Scala Programming Language**
Newsgroups: **gmane.comp.lang.scala**
Date: 2004-01-20 17:25:18 GMT (10 years, 15 weeks, 6 days, 3 hours and 12 minutes ago)

We'd like to announce availability of the first implementation of the Scala programming language. Scala smoothly integrates object-oriented and functional programming. It is designed to express common programming patterns in a concise, elegant, and type-safe way. Scala introduces several innovative language constructs. For instance:

- Abstract types and mixin composition unify ideas from object and module systems.

- Pattern matching over class hierarchies unifies functional and object-oriented data access. It greatly simplifies the processing of XML trees.

- A flexible syntax and type system enables the construction of advanced libraries and new domain specific languages.

At the same time, Scala is compatible with Java. Java libraries and frameworks can be used without glue code or additional declarations.

The current implementation of Scala runs on Java VM. It requires JDK 1.4 and can run on Windows, MacOS, Linux, Solaris, and most other operating systems. A .net version of Scala is currently under development.

For further information and downloads, please visit:

现状

- ~100,000 Scala Dev v.s. ~10,000,000 Java Dev
- 成功案例: twitter, linkedin,沃尔玛,Sony,REA
- 用不用Scala, 仍然很有争议

学习资源

- zh.scala-tour.com
- http://twitter.github.io/scala_school/zh_cn/index.html
- scalacn@googlegroup.com
- https://class.coursera.org/progfun-003