Atitit 编程语言的类型系统

目录

[1.2. 动态类型语言（Dynamically Typed Language）： 1](#_Toc29885)

[1.3. 静态类型语言（Statically Typed Language）： 1](#_Toc10288)

[1.4. 强类型定义语言（Explicit type conversion，强制数据类型定义语言，类型安全的语言）： 2](#_Toc1608)

[1.5. 弱类型定义语言（Implicit type conversion，类型不安全的语言）： 2](#_Toc8408)

[2. 静态类型可以分为两种： 2](#_Toc6153)

[2.1. explicitly typed显式类型 2](#_Toc28040)

[2.2. implicity typed隐式类型 3](#_Toc19660)

[2.3. 宣告型态（manifest type）语言， 3](#_Toc7402)

[2.4. type-inferred语言 3](#_Toc31550)

[3. 常见语言分布图 3](#_Toc27369)

#### **2. 四种语言分类及其区别**

## 动态类型语言（Dynamically Typed Language）：

运行期间才做数据类型检查的语言，即动态类型语言编程时，永远不用给任何变量指定数据类型。该语言会在第一次赋值给变量时，在内部将数据类型记录下来。

例如：ECMAScript(JavaScript)、Ruby、Python、VBScript、php

Python和Ruby就是典型动态类型语言，其他各种脚本语言如VBScript也多少属于动态类型语言

优点：方便阅读，不需要写非常多的类型相关的代码；

缺点：不方便调试，命名不规范时会造成读不懂，不利于理解等

## 静态类型语言（Statically Typed Language）：

编译期间做检查数据类型的语言，即写程序时要声明所有变量的数据类型，是固定的。使用数据之前，必须先声明数据类型（int ,float,double等）。相当于使用之前，首先要为它们分配好内存空间。

例如：C/C++是静态类型语言的典型代表，其他的静态类型语言还有C#、JAVA等

优点：结构非常规范，便于调试，方便类型安全

缺点：为此需要写更多类型相关代码，不便于阅读、不清晰明了

## 强类型定义语言（Explicit type conversion，强制数据类型定义语言，类型安全的语言）：

一旦变量被指定某个数据类型，如果不经强制转换，即永远是此数据类型。

举例：若定义了一个整型变量a，若不进行显示转换，不能将a当作字符串类型处理

强类型语言是指需要进行变量/对象类型声明的语言，一般情况下需要编译执行。例如C/C++/Java/C#

## 弱类型定义语言（Implicit type conversion，类型不安全的语言）：

数据类型可以被忽略的语言。它与强类型定义语言相反, 一个变量可以赋不同数据类型的值。

举例：在VBScript中，可以将字符串 '12' 和整数 3 进行连接得到字符串 '123'，  然后可以把它看成整数 123，而不需要显示转换

例如PHP/ASP/Ruby/Python/Perl/ABAP/SQL/JavaScript/Unix Shell等

注意：强类型定义语言在速度上可能略逊色于弱类型定义语言，但是强类型定义语言带来的严谨性能够有效的避免许多错误。

作者：陈满iOS  
链接：https://www.jianshu.com/p/336f19772046  
來源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。

# 静态类型可以分为两种：

## explicitly typed显式类型

如果类型是语言语法的一部分，在是**explicitly typed显式类型；**

## implicity typed隐式类型

如果类型通过编译时推导，是**implicity typed隐式类型**, 比如ML和Haskell

## 宣告型态（manifest type）语言，

包含宣告型态（manifest type）语言，即每一个变量和函数的型态都清楚地宣告

## type-inferred语言

type-inferred语言（例如MUMPS，ML） 类型推断

# 常见语言分布图

