

彻底掌握Kotlin

变量常量与类型

A

条件语句

B

函数

C

android

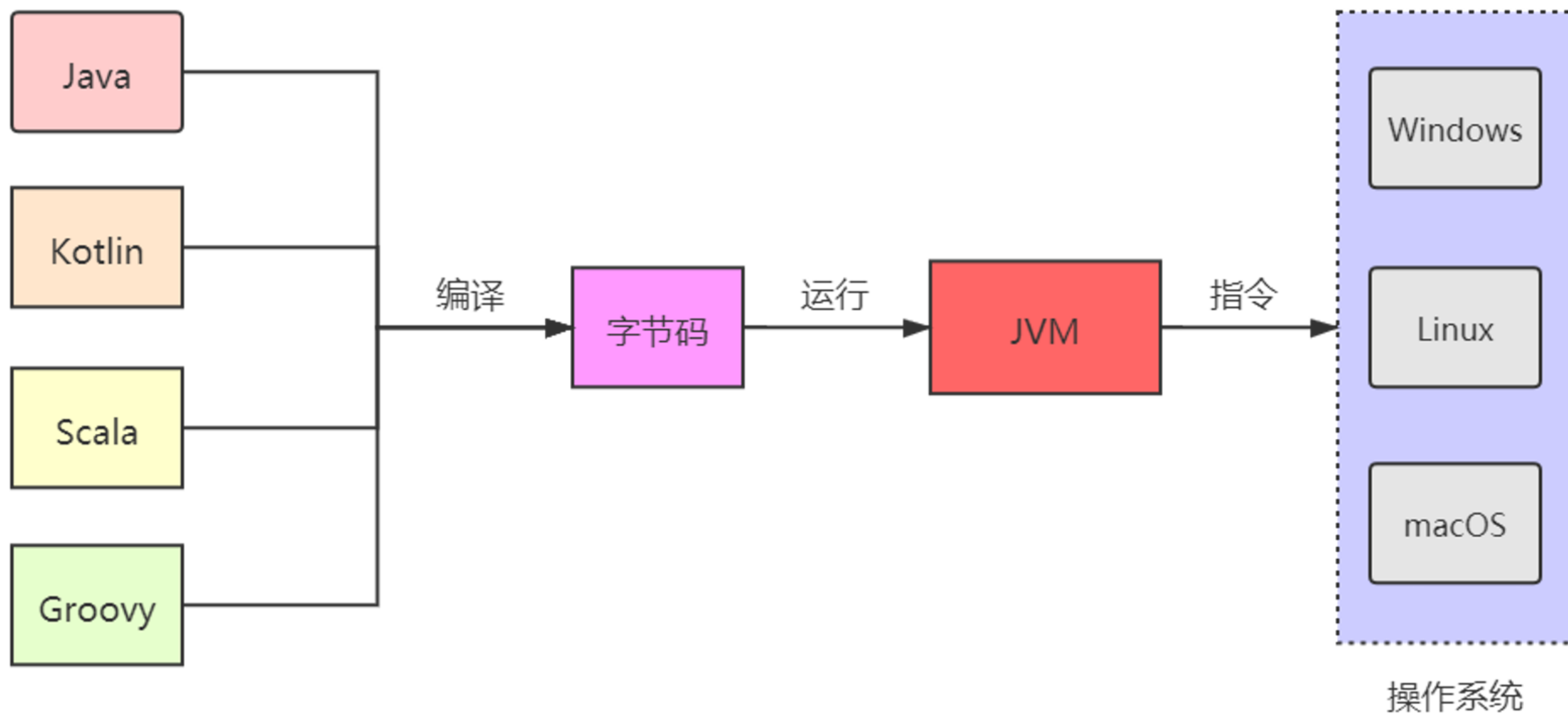


Kotlin初探

Kotlin的诞生

- 2011年，JetBrains宣布开发Kotlin编程语言，这门新语言可以用来编写在Java虚拟机上运行的代码，是Java和Scala语言之外的又一选择。2017年，Google在赢得与Oracle的诉讼一年后，**Google宣布Kotlin正式获得官方支持**，可用于Android应用开发。Kotlin的应用范围迅速扩展，它从一门前途光明的编程语言摇身一变，成了这个世界上最重要的移动操作系统的钦定开发语言。kotlin语法简洁，具备现代高级语言特性，并且能和Java遗留代码无缝互操作。因为具备这些优势，今天，越来越多的公司使用它开发。

Kotlin与JVM



为什么要学Kotlin

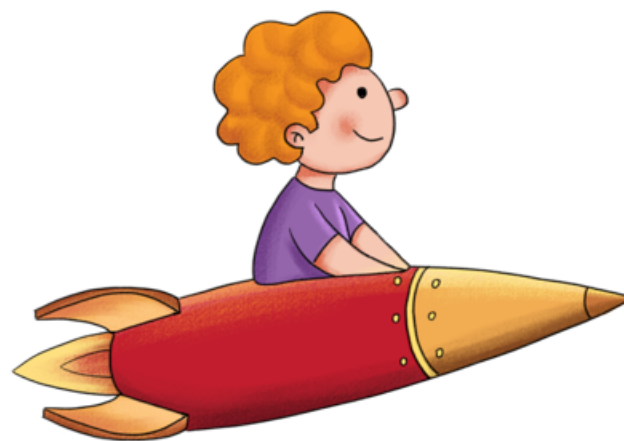
- Java语言比较稳健，久经考验。多年来，它一直是最常用的一种编程语言，造就了庞大的生产代码库。自从1995年Java问世以来，对于优秀的编程语言应满足什么条件，人们已通过实践积攒了很多经验教训。然而，**Java却裹足不前**，开发者喜欢的很多现代语言高级特性，它都没有，或者迟迟加入。
- Kotlin从这些经验教训中受益良多，而Java中的某些早期设计却愈显陈旧。脱胎于旧语言，**Kotlin解决了他们的很多痛点**，进化成了一门优秀的语言。相比Java，Kotlin进步巨大，带来了更可靠的开发体验。

Kotlin的跨平台特性

- Kotlin不仅支持编写代码在虚拟机上运行，而且还是一门跨平台的通用型语言，我们可以用Kotlin开发各种类型的原生应用，如Android、macOS、Windows、JavaScript应用。
- Kotlin能脱离虚拟机层，直接编译成可以在Windows、Linux和macOS平台上运行的原生二进制代码。

学习方式

- 与Java语言对比，提高学习效率
- 知识点尽量覆盖全面，不留认知缺陷
- 课程不断更新迭代



android

变量常量与类型

声明变量

赋值运算符

```
var maximumAge: Int = 5
```

变量定义关键字 变量名 类型定义 赋值

The diagram illustrates the components of the variable declaration `var maximumAge: Int = 5`. Brackets are used to group parts of the code: a bracket under `var` is labeled '变量定义关键字' (Variable definition keyword); a bracket under `maximumAge` is labeled '变量名' (Variable name); a bracket under `: Int` is labeled '类型定义' (Type definition); a bracket under `=` is labeled '赋值运算符' (Assignment operator) above it; and a bracket under `5` is labeled '赋值' (Assignment) below it.

Kotlin内置数据类型

类型	描述	示例
String	字符串	“Hello World”
Char	单字符	‘A’
Boolean	true/false	true false
Int	整数	5
Double	小数	3.14
List	元素集合	1,8,10 “Jack”,“rose”,“Jack”
Set	无重复元素的集合	“Jack”,“Jason”,“Jacky”
Map	键值对集合	“small” to 5, “medium” to 8, “large” to 9

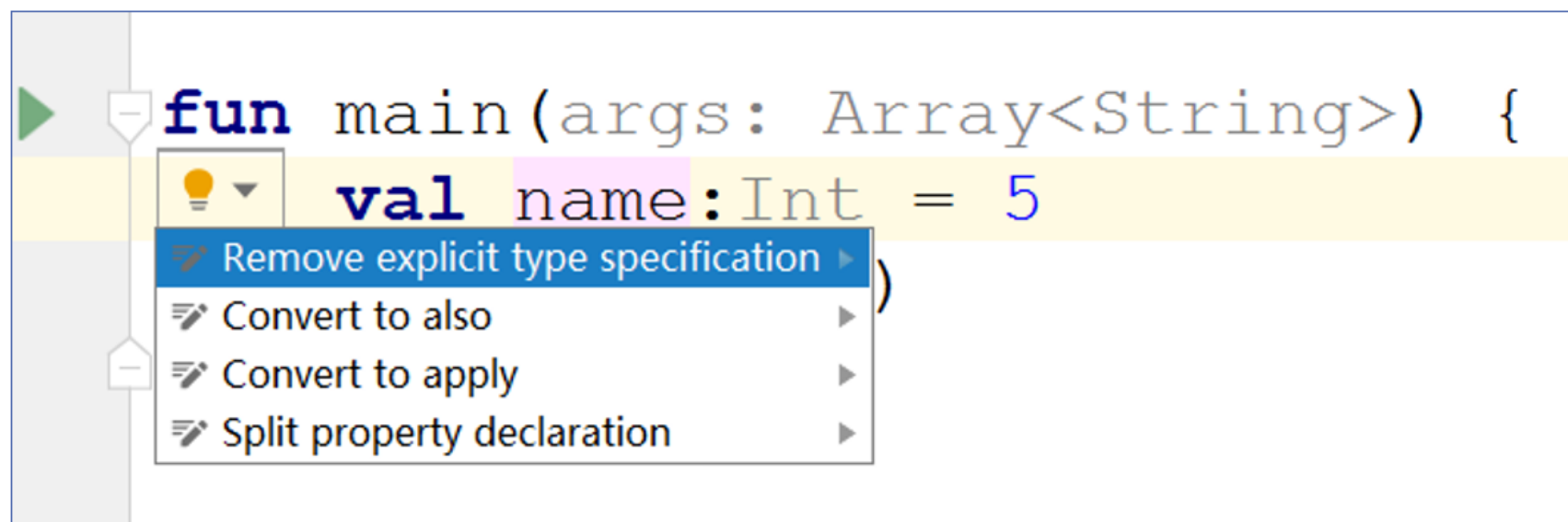
只读变量

- 要声明可修改变量，使用var关键字。
- 要声明只读变量，使用val关键字。

```
2  ▶  fun main(args: Array<String>) {  
3      val name:String = "jack"  
4      var age:Int = 10;  
5      age += 1;  
6      println(name)  
7  }
```

类型推断

- 类型推断：对于已声明并赋值的变量，它允许你省略类型定义。



编译时常量

- 只读变量并非绝对只读。
- **编译时常量只能在函数之外定义**，因为编译时常量必须在编译时赋值，而函数都是在运行时才调用，函数内的变量也是在运行时赋值，编译时常量要在这些变量赋值前就已存在。
- 编译时常量只能是常见的基本数据类型：String、Int、Double、Float、Long、Short、Byte、Char、Boolean。

查看Kotlin字节码

➤ 查看Kotlin编译之后的字节码，有助于我们深入理解Kotlin语言。

➤ 两种方式

- Shift键两次，输入Show kotlin
- Tools->Kotlin->Show Kotlin Bytecode

```
Kotlin Bytecode
Decompile [x] Inline [x] Optimization [x] Assertions [x] IR [ ] JVM 8 target [ ]

4 public final class HelloKt {
5
6
7 // access flags 0x19
8 public final static Ljava/lang/String; MAX = ""
9 @Lorg/jetbrains/annotations/NotNull;() // invis
10
11 // access flags 0x19
12 public final static main([Ljava/lang/String;)V
13 // annotable parameter count: 1 (visible)
14 // annotable parameter count: 1 (invisible)
15 @Lorg/jetbrains/annotations/NotNull;() // invi
16 L0
17 ALOAD 0
18 LDC "args"
```


Kotlin的引用类型与基本数据类型

- Java有两种数据类型：引用类型与基本数据类型。
- Kotlin只提供引用类型这一种数据类型，出于更高性能的需要，Kotlin编译器会在Java字节码中**改用基本数据类型**。



android

条件语句

表达式

➤ if/else if表达式

➤ range表达式

- in A..B, in关键字用来检查某个值是否在指定范围之内。

➤ when表达式

- 允许你编写条件式，在某个条件满足时，执行对应的代码
- 只要代码包含else if分支，都建议改用when表达式

```
15      val school = "0学"
16      val level:Any = when(school) {
17          "学前班" -> "幼儿"
18          "小学" -> "少儿"
19          "中学" -> "青少年"
20          "大学" -> "成年"
21          else -> {
22              println("未知")
23          }
24      }
25      println(level)
26  }
```

string模板

- 模板支持在字符串的引号内放入变量值
- 还支持字符串里计算表达式的值并插入结果，添加在`${}`中的任何表达式，都会作为字符串的一部分求值。

```
1  ▶ fun main() {  
2      val origin = "Jack"  
3      val dest = "Rose"  
4      println("$origin love $dest")  
5  
6      val flag = false  
7      println("Answer is: ${if(flag) "我可以" else "对比起"}")  
8  }
```

ANDROID



函数

函数头

可见性修饰符 函数名 返回类型

private fun doSomething (age: Int, flag: Boolean) : String

函数声明关键字 函数参数

The diagram illustrates the components of a Kotlin function signature: `private fun doSomething (age: Int, flag: Boolean) : String`. Annotations with arrows point to specific parts: '可见性修饰符' (visibility modifier) points to `private`; '函数名' (function name) points to `doSomething`; '返回类型' (return type) points to `: String`; '函数声明关键字' (function declaration keyword) points to `fun`; and '函数参数' (function parameters) points to the parameter list `(age: Int, flag: Boolean)`, which is enclosed in a dashed box.

函数参数

➤ 默认值参

- 如果不打算传入值参，可以预先给参数指定默认值

➤ 具名函数参数

- 如果使用命名值参，就可以不用管值参的顺序

```
7  fun fix(name: String, age: Int = 2) {  
8      println(name + age)  
9  }  
10  
11  fun main() {  
12      fix(age=4, name="jack")  
13  }
```


Unit函数

- 不是所有函数都有返回值，Kotlin中**没有返回值的函数叫Unit函数**，也就是说他们的返回类型是Unit。在Kotlin之前，函数不返回任何东西用**void**描述，意思是“没有返回类型，不会带来什么，忽略它”，也就是说如果函数不返回任何东西，就忽略类型。但是，void这种解决方案无法解释现代语言的一个重要特征，泛型。

Nothing类型

- TODO函数的任务就是抛出异常，就是永远别指望它运行成功，返回Nothing类型。

```
public inline fun TODO(reason: String): Nothing = throw NotImplementedError()
```

```
1  fun main() {  
2      var i = 1;  
3      if(i > 3){  
4          println(i)  
5      }else{  
6          TODO(reason: "argument is too small.")  
7      }  
8      i++  
9      println(i)  
10 }
```

反引号中的函数名

- Kotlin可以使用空格和特殊字符对函数命名，不过函数名要用一对反引号括起来。
- 为了支持Kotlin和Java互操作，而Kotlin和Java各自却有着不同的保留关键字，不能作为函数名，使用反引号括住函数名就能避免任何冲突。

```
1 fun `**~special function with weird name**`() {  
2     println("I am weird.")  
3 }  
4  
5 fun main() {  
6     `**~special function with weird name`()  
7 }
```

```
5 fun main() {  
6     MyJava.`is`()  
7 }
```