**什么是Kotlin**？ Kotlin，如前面所说，

它是JetBrains开发的基于JVM的语言。JetBrains因为创造了 一个强大的Java开发IDE被大家所熟知。Android Studio，官方的Android IDE，就 是基于Intellij，作为一个该平台的插件。 Kotlin是使用Java开发者的思维被创建的，Intellij作为它主要的开发IDE。

**对于 Android开发者，有两个有趣的特点**：

对Java开发者来说，Kotlin是非常**直觉化**的，并且非常容易学习。语言的大部 分内容都是与我们知道的非常相似，不同的地方，它的基础概念也能迅速地掌 握它。 它与我们日常生活使用的IDE无需配置就能完全整合。***Android Studio能够非常 完美地理解、编译运行Kotlin代码***。而且对这门语言的支持来正是自于开发了 这个IDE的公司本身，所以我们Android开发者是一等公民。 但是这仅仅是开发语言和开发工具之间的整合。相比Java 7的优势到底是什么呢？ 它**更加易表现**：这是它最重要的优点之一。你可以编写少得多的代码。

**它更加安全**：

Kotlin是空安全的，也就是说在我们**编译时期就处理了各种null的 情况**，避免了执行时异常。如果一个对象可以是null，则我们需要明确地指定 它，然后在使用它之前检查它是否是null。你可以节约很多调试空指针异常的 时间，解决掉null引发的bug。

**它是函数式的**：Kotlin是基于面向对象的语言。但是就如其他很多现代的语言 那样，它使用了很多函数式编程的概念，比如，使用lambda表达式来更方便地 解决问题。

其中一个很棒的特性就是Collections的处理方式。

## 函数式编程

函数式编程的精髓在于函数本身。在函数式编程中函数是第一等公民，与其他数据类型一样，处于平等地位，可以赋值给其他变量，也可以作为参数，传入另一个函数，或者作为别的函数的返回值。

函数式编程好的实践在于对运算过程的高度抽象和没有”副作用”（既保持函数的独立性），函数式编程三板斧是函数式编程的典型范式，在编程中被大量使用，即使人们不关注函数式编程，在使用函数式编程语言的时候，也会不自觉的使用这些函数。

函数式编程是一种思维方式，函数式编程鼓励放弃对状态的维持（是命令式编程的基础），将所有的操作都交给运行时去执行。当然为了保证程序运行的效率，这需要提供一些辅助性的手段（缓存、缓求值等）。

view.setOnClickListener(object : OnClickListener { override fun onClick(v: View) { toast("Click") } }

，Kotlin允许Java库的一些优化，**Interface中包含单个函数可以被替代 为一个函数**。

所以可以定义为这样fun setOnClickListener(listener: (View) -> Unit)

它可以**扩展函数**：这意味着我们可以扩展类的更多的特性，甚至我们没有权限 去访问这个类中的代码。

它是**高度互操作性**的：你可以继续使用所有的你用Java写的代码和库，因为两 个语言之间的互操作性是完美的。甚至可以在一个项目中使用Kotlin和Java两 种语言混合编程。

Anko <http://www.jianshu.com/p/7cf5b42eb25f>

[Anko](https://github.com/JetBrains/anko) 是一个使开发Android应用更简单更快捷的库,Anko使你的代码简洁易懂, 使开发者不用再在意Android SDK对Java版本的限制(目前还不支持Java8 =。= ).

Anko主要是针对 Java 和 Kotlin developers.

来一个列子看一下。为了不太墨迹，一些不必要的xml声明此处略去。

<RelativeLayout>

<TextView

android:id="@+id/sample\_text\_view"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_alignParentTop="true"

android:text="Sample text view"

android:textSize="25sp" />

<Button

android:id="@+id/sample\_button"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_below="@+id/sample\_text\_view"

android:text="Sample button" />

</RelativeLayout>

//使用anko定义布局

relativeLayout {

val textView = textView("Sample text view") {

textSize = 25f

}.lparams {

width = matchParent

alignParentTop()

}

button("Sample button").lparams {

width = matchParent

below(textView)

}

}

**Kotlin中没有new关键字**

**变量和属性** 在Kotlin中，一切都是对象。没有像Java中那样的原始基本类型。这个是非常有帮 助的，因为我们可以使用一致的方式来处理所有的可用的类型。

**基本类型** 像integer，float或者boolean等类型仍然存在，但是它们全部都会作为对象 存在的。数字类型中不会自动转型

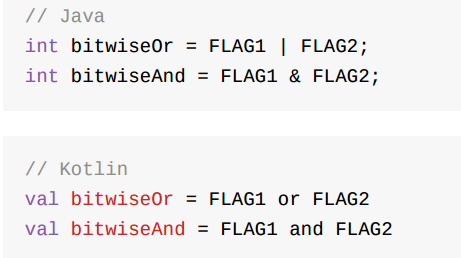
举个例子，你不能给 Double 变量分配一 个 Int 。

必须要做一个明确的类型转换，可以使用众多的函数之一： val i:Int=7 val d: Double = i.toDouble()

字符（Char）不能直接作为一个数字来处理。

在需要时我们需要把他们转换为 一个数字： val c:Char='c' val i: Int = c.toInt() 位运算也有一点不同。

在Android中，我们经常在 flags 中使用“或”，所以我 使用"and"和"or来举例：



用***try.kotlinlang.org***在不需要一个真正的项目和不需要部署到机器的前提下来测试一 些简单的代码范例

 a report written by a developer at Square



**变量**可以很简单地定义成可变( var )和不可变（ val ）的变量。这个与Java中使 用的 final 很相似。

一个不可变对象意味着它在实例化之后就不能再去改变它的状态了。如果你需要一 个这个对象修改之后的版本，那就会再创建一个新的对象。这个让编程更加具有健 壮性和预估性。在Java中，大部分的对象是可变的，那就意味着任何可以访问它这 个对象的代码都可以去修改它，从而影响整个程序的其它地方。 不可变对象也可以说是线程安全的，因为它们无法去改变，也不需要去定义访问控 制，因为所有线程访问到的对象都是同一个。 所以在Kotlin中，如果我们想使用不可变性，我们编码时思考的方式需要有一些改 变。一个重要的概念是：尽可能地使用 val 。除了个别情况（特别是在Android 中，有很多类我们是不会去直接调用构造函数的），大多数时候是可以的。

命名参数 在调用函数时可以参数可以命名。这对于那种有大量参数的函数是很方便的.

注意,命名参数语法不能够被用于调用Java函数中,因为Java的字节码不能确保方法 参数命名的不变性

使用 Ant 获得 Ant 任务 Kotlin 提供了 Ant 三个任务:

kotlinc : Kotlin 面向 JVM 的编译器

kotlin2js: 面向 javaScript 的编译器

withKotlin: 使用标准 javac Ant 任务时编译

Kotlin 文件的任务 这些任务定义学在 kotlin-ant.jar 标准库中，位与 kotlin compiler 的 lib 文件夹下

# <http://www.tuicool.com/articles/v6fiMzN> Kotlin 函数式编程三板斧