
HZ BOOKS
华章科技

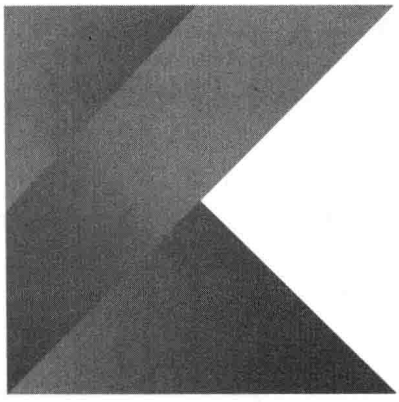
水滴技术团队出品，开源项目Qui三核心贡献者**章建良**亲自执笔

围绕Kotlin设计理念，深入阐述Kotlin设计哲学、基础语法、语言特性、设计模式、函数式编程、异步开发和工程实战等核心内容

# Kotlin 核心编程

水滴技术团队 —— 著

机械工业出版社
China Machine Press



DIVE INTO
KOTLIN

# Kotlin
# 核心编程

水滴技术团队◎著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Kotlin 核心编程 / 水滴技术团队著 . —北京：机械工业出版社，2019.4

ISBN 978-7-111-62431-8

I. K… II. 水… III. JAVA 语言 - 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2019）第 062053 号

# Kotlin 核心编程

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）

责任编辑：孙海亮　　责任校对：殷　虹

印　　刷：北京瑞德印刷有限公司　　版　　次：2019 年 4 月第 1 版第 1 次印刷

开　　本：186mm × 240mm　1/16　　印　　张：23.25

书　　号：ISBN 978-7-111-62431-8　　定　　价：89.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

客服热线：（010）88379426　88361066　　投稿热线：（010）88379604

购书热线：（010）68326294　　读者信箱：hzit@hzbook.com

# Preface 前　言

## 为什么要写这本书

2017 年 5 月，Hadi Hariri（JetBrains 的首席布道师）在座无虚席的 Google I/O 大会上介绍 Kotlin 时，先开了一个玩笑："大概 4 年半之前，我们曾在一个容纳 900 人的会场做过同样的事情，但结果只来了 7 个人。"

他说的是事实，自从 Google 宣布 Kotlin 成为 Android 官方编程语言之后，Kotlin 这门默默无闻的语言一下子成为技术圈中的"明星"。随后，关于 Kotlin 的开源项目和学习资料也如雨后春笋般出现。

同一时刻，在我们位于杭州的办公室里，水滴的同事也在进行着一个用 Kotlin 研发的 Android 项目。作为一个采用 Scala 全栈开发的"非主流"技术团队，我们对 Kotlin 有天然的好感。一方面，它在某些地方非常像 Scala。相比 Java，它们都拥有更简洁的语法，以及更多的函数式特性（如高阶函数、更强的类型推导、不同程度上的模式匹配等）。另一方面，Kotlin 还有比 Scala 更快的编译速度，同时兼容 Java 6，这使得我们可以用它完美替代 Java 以更好地进行 Android 开发工作。

那么 Kotlin 到底是怎样一门编程语言呢？我们试图通过这本书来回答这个问题。

与其他 Kotlin 的书籍不同，本书在工具属性上会显得稍弱。如果你想快速索引 Kotlin 某个具体语法的使用，推荐你去阅读 Kotlin 的官方文档或者《Kotlin 极简教程》。但假使你有一颗好奇的心，渴望窥探 Kotlin 这门语言的设计哲学，那么本书可以提供一个浅薄的参考视角。本书会围绕 Kotlin 的设计理念，介绍其核心的语言特性，探索它在设计模式、函数式编程、并发等方面的具体应用。

越来越多的公司和团队开始加入 Kotlin 的阵营。除了 Android 之外，依靠 Kotlin Native 等项目，Kotlin 也开始在其他领域施展拳脚。在 Android 官方支持 Kotlin 之后的数月，Google 又推出了 Android 的 Kotlin 扩展库，在很大程度上提升了 Android 开发的体验。Spring 5 正式发布时，也将 Kotlin 作为其主打的新特性之一，使 Kotlin 再一次受到了很多 Web 开发者的关注。这一切都预示着这门语言将有无比广阔的前景。

值得注意的是，除了蓬勃发展的生态之外，Kotlin 语言本身也在不断迭代。截至本书完稿时，Kotlin 又发布了一些有趣的新特性（如 inline class），我们对 Kotlin 的未来充满了期待。

## 读者对象

- Kotlin 爱好者
- 想进阶的 Java 程序员
- 对函数式编程感兴趣的读者
- Android 开发者
- 开设 Java 相关课程的大专院校的学生

## 本书主要内容

本书分为 4 部分：

第 1 部分为热身篇——Kotlin 基础。介绍 Kotlin 设计哲学、生态及基础语法。

第 2 部分为下水篇——Kotlin 核心。涉及 Kotlin 的语言特性，包括面向对象、代数数据类型、模式匹配、类型系统、Lambda、集合、多态、扩展、元编程等方面的知识。其中“代数数据类型和模式匹配”“多态和扩展”在同类书籍中没有过多深入，但笔者认为它们是 Kotlin 语言中相当重要的特性和应用，故本书中进行了详细介绍探索。

第 3 部分为潜入篇——Kotlin 探索。该部分之所以命名为“探索”，是希望进一步探索 Kotlin 的设计模式和编程范式，内容包含设计模式、函数式编程、异步和并发编程。其中“函数式编程”为超越 Kotlin 本身的内容，但可以为读者提供深入理解 Kotlin 语言特性的示范。

第 4 部分为遨游篇——Kotlin 实战。着重演示 Kotlin 在 Android 和 Web 平台中的应用，包含基于 Kotlin 的 Android 架构、开发响应式 Web 应用。

## 勘误和支持

由于作者的水平有限，编写时间仓促，书中难免会出现一些错误或者不准确的地方，恳请读者批评指正。如果你遇到任何问题，或有宝贵意见，欢迎发送邮件至邮箱 scala.cool@gmail.com，期待能够得到你们的真挚反馈。

此外，本书所有章节相关的源代码可以通过 https://github.com/DiveIntoKotlin 进行查看和下载。

# 致谢

首先感谢程凯先生，他第一时间跟进并审阅了本书大部分稿件，反馈了许多相当宝贵的改进意见。他所在的小余科技是由傅盛投资的创业公司，有着一支非常优秀的 Android 研发团队。

感谢淘宝飞猪旅行的陈静波先生、依图科技的邵子奇先生、麦多多的创始人古井好月（花名）、工行交易银行中心的季乔卡先生、图讯科技的邹苏启先生、浙江工业大学的蒋博文同学，他们对本书进行了审阅，对内容的纠错和调整提供了很大的帮助。

感谢机械工业出版社华章公司的杨福川老师、孙海亮老师，他们参与了本书的创作，为书籍的策划及写作细节提供了专业的指导，使得本书可充分面向读者、面向市场。

最后，要感谢水滴团队所有参与本书创作的小伙伴们，是大家认真的态度和强大的执行能力促成了本书的出版，他们分别是章建良、单鑫鑫、泮关森、肖宇、汤俞龙、袁国浩。

谨以此书献给所有关注我们 ScalaCool 团队博客的朋友们，以及众多热爱 Kotlin 的开发者！

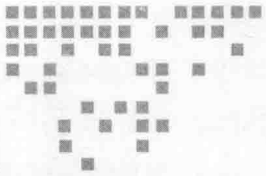
水滴技术团队

# 目　录 *Contents*

## 下水篇 Kotlin 核心

热身篇

# Kotlin 基础



Chapter 1 第 1 章

# 认识 Kotlin

在 Java 之后，JVM 平台上出现了其他的编程语言，Scala 和 Kotlin 可以算是其中的佼佼者。Scala 已成为大数据领域的明星，而 Kotlin 在 2017 年 Google I/O 大会之后，也成为安卓平台上潜力巨大的官方支持语言。它们都因被冠以“更好的 Java”而为人称道，然而它们采用的却是两种不同的设计理念。

本章我们通过对比 Java、Scala、Kotlin 这 3 种编程语言各自的发展路线，来认识 Kotlin 的设计哲学。

## 1.1 Java 的发展

不得不说，Java 是当今最成功的编程语言之一。自 1996 年问世，Java 就始终占据着编程语言生态中很大的份额。它的优势主要体现在：

- **多平台与强大的社区支持**。无论是用于 Web 开发还是用于移动设备，Java 都是主流的编程语言之一。
- **尊重标准**。它有着严格的语言规范及向后兼容性，因此非常适合开发团队之间的协作，即使组织变动，新人同样可以在相同的规范下快速参与项目开发。

然而，随着计算平台的快速发展，平台和业务本身对编程语言提出了更大的挑战。Java 的发展也受到环境变化所带来的影响。一方面，多核时代与大数据的到来，使得古老的函数式编程又重新变得“时髦”，Scala、Clojure 这种多范式的编程语言开始受到越来越多开发人员的关注和喜爱；另一方面，Java 的严格规范也常常引发抱怨。

因此，Java 必须开始改变。

### 1.1.1　Java 8 的探索

如果说 Java 5 引入泛型是 Java 发展历史上重大的进步，那么 Java 8 的发布也同样意义深远，它是 Java 对其未来发展的一次崭新探索。Java 8 引入了很多全新的语言特性，如：

- 高阶函数和 Lambda。首次突破了只有类作为“头等公民”的设计，支持将函数作为参数来进行传递，同时结合 Lambda 语法，改变了现有的程序设计模式。
- Stream API。流的引入简化了日常开发中的集合操作，赋予了 Java 更强大的业务表达能力，并增强了代码的可读性。
- Optional 类。它为消除 null 引用所带来的 NullPointerException 问题，在类型层面提供了一种解决思路。

这一次的发布在 Java 社区引起了不同寻常的反响，因为 Java 程序员开始感受到另外一种编程范式所带来的全新体验，也就是所谓的函数式编程。拥抱函数式也为 Java 的发展指出了一个很好的方向。

### 1.1.2　Java 未来的样子

2016 年 11 月，在欧洲最大的 Java 峰会上，Oracle 的 Java 语言架构师 Brian Goetz 分享了关于 Java 这门语言未来发展的演讲。本次会议最大的收获就是探索了未来 Java 可能支持的语言特性，它们包含：

- 数据类。
- 值类。
- 泛型特化。
- 更强大的类型推导。
- 模式匹配。

以上的语言特性对于初尝函数式编程甜头的 Java 开发者而言，是十分值得期待的。它们可以进一步解放 Java，让开发工作变得更加高效和灵活。比如，一旦 Java 支持了数据类，我们就可以用非常简短的语法来表示一个常见的数据对象类，如下所示：

```
public class User(String firstName, String lastName, DateTime birthday)
```

而若用如今的 JavaBean，则意味着好几倍的代码量，这一切都让人迫不及待。与此同时，或许早有 Java 程序员开始了 JVM 平台上另一种语言的研究。这门语言已支持了所有这些新的特性，并在设计之初就集成了面向对象和函数式两大特征，它就是 Scala。

## 1.2　Scala 的百宝箱

Scala 是洛桑联邦理工大学的马丁（Martin Odersky）教授创造的一门语言。他也参与了 Java 语言的发展研究工作，在 Java 5 中引入的泛型就是他的杰作。事实上，在 Java 刚发布

的时候，马丁教授就开始了 Java 的改良工作——他在 JVM 平台探索函数式编程，并发布了一个名为 Pizza 的语言，那时就支持了泛型、高阶函数和模式匹配。

然而，在随后的探索过程中，他渐渐发现 Java 是一门具有硬性约束的语言，在某些时候不能采用最优的方式来实施设计方案。因此，马丁教授和他的研究伙伴决定重新创造一门语言，既在学术上合理，同时也具备实用价值。这就是开发 Scala 的初衷。

### 1.2.1 学术和工业的平衡

Scala 是一门非常强大的编程语言，正如它名字（Scalable，可拓展）本身一样，用 Scala 编程就像拥有了哆啦 A 梦的口袋，里面装满了各种编程语言特性，如面向对象、函数式、宏。

Scala 不仅在面向对象方面进行了诸多的改良，而且彻底拥抱了函数式。因此 Scala 也吸引了函数式编程社区很多厉害的程序员，他们将函数式编程的思想注入 Scala 社区，如此将使用 Scala 进行函数式编程提高到了新的高度。

由于 Scala 设计者学院派的背景，以及 Scala 某些看似“不同寻常”的语法，使它在发展早期（甚至现在）经常被描述为“过于学院派”，以至于马丁教授在某次 Scala 大会的演讲时，自嘲“Scala 真正的作用是将人引向了 Haskell”。

然而，真实的 Scala 却是在不断地探索学术和实用价值两方面的平衡。不可否认的是：

- Scala 已经成为大数据领域的热门语言，明星项目 Spark 就是用 Scala 开发的，还有很多其他知名的项目，如 Akka、Kafka 等。
- 越来越多的商业公司，如 Twitter、PayPal、Salesforce 都在大量使用这门语言。

另外，Scala 也确实是一门有着较陡的学习曲线的语言，因为它强大且灵活，正如马丁教授所言，Scala 相信程序员的聪明才智，开发人员可以用它来灵活选择语言特性。但学术和工业的平衡始终是一个难题，与 Java 严格标准相比，Scala 的多重选择也常常因复杂而被人吐槽。

### 1.2.2 复合但不复杂

那么，Scala 真的复杂吗？我们不知听了多少次类似这样的抱怨。在搞明白这个问题之前，我们需要先弄清楚到底什么是“复杂”。在英文中，复杂一词可以联想到两个单词：complex 和 complicated。实际上它们的含义截然不同，更准确地说，complex 更好的翻译是“具有复合性”。

Nicolas Perony 曾在 Ted 上发表过一次关于“复合性理论”的演讲。

---

什么是复合性？复合并不是复杂。一件复杂的事物是由很多小部分组成的，每一部分都各不相同，而且每一部分都在这个体系中有其自身的确切作用。与之相反，一个复合的系统是由很多类似的部分所组成的，而且（就是因为）它们之间的相互影响形成了一种宏观上一致的行为。复合系统含有很多互动的元素，它们根据简单的、个体的规则行动，如此导致新特征的出现。

---

马丁教授曾发表过一篇名为《简单还是复杂》的文章，表达过类似的观点。如果对搭积木这件事情进行思考，摩比世界提供了固定的方案，而乐高则提供了无穷的选择。然而，前者的零件种类和数量都比后者要多得的。类似的道理，编程语言可以依靠功能累加来构建所谓的语法，同样也可以通过简单完备的理论来发展语言特性。在马丁教授看来，Scala 显然属于后者，它并不复杂，而且非常简单。

### 1.2.3　简单却不容易

事实上，函数式编程最明显的特征就是具备复合性。函数式开发做得最多的事情就是对需要处理的事物进行组合。如果说面向对象是归纳法，侧重于对事物特征的提取及概括，那么函数式中的组合思想则更像是演绎法，近似于数学中的推导。

“简单”的哲学也带来了相应的代价：

- 这是一种更加抽象的编程范式，诸如高阶类型、Typeclass 等高级的函数式特性虽然提供了无比强大的抽象能力，但学习成本更高。
- 它建立了另一种与采用 Java 面向对象编程截然不同的思维模式。这种思维方式上的巨大差异显然是一个极高的门槛，同时也是造成 Scala 令人望而却步的原因之一。

Scala 在选择彻底拥抱函数式的同时，也意味着它不是一门容易的语言，它无法成为一门像 Java 那样主流的编程语言。事实上，即使很多人采用 Scala 来进行开发，也还是采用类似 Java 的思维模式来编程。换句话说，Scala 依旧是被当作更好的 Java 来使用的，但这确实是当今主流编程界最大的诉求。

在这种背景下，Kotlin 作为一门 JVM 平台上新兴的编程语言，悄悄打开了一扇同样广阔的大门。

## 1.3　Kotlin——改良的 Java

2010 年，JetBrains 产生创造 Kotlin 的想法。关于大名鼎鼎的 JetBrains，想必在业内是人尽皆知，知名的 IntelliJ IDEA 就是他们的产品之一。拥有为各种语言构建开发工具经验的 JetBrains，自然是对编程语言设计领域最熟悉的一群人。当时，一方面他们看到了 C# 在 .NET 平台上大放异彩；另一方面，Java 相比新语言在某种程度上的滞后，让他们意识到改良 Java 这门主流语言的必要性。

JetBrains 团队设计 Kotlin 所要面临的第一个问题，就是必须兼容他们所拥有的数百万行 Java 代码库，这也代表了 Kotlin 基于整个 Java 社区所承载的使命之一，即需要与现有的 Java 代码完全兼容。这个背景也决定了 Kotlin 的核心目标——为 Java 程序员提供一门更好的编程语言。