Pearson

Broadview
www.broadview.com.cn

Dart 编程语言

[美] Gilad Bracha 著
戴虹 译

The Dart Programming Language

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

Dart 编程语言

[美] Gilad Bracha 著
戴虬 译

The Dart Programming Language

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书总计 9 章，前 6 章对 Dart 的对象、库、函数、类型、表达式与语句等基础知识进行了详细介绍；第 7、8 章对反射和 isolate 等进阶内容进行了深入讲解；第 9 章为总结。本书与众不同之处是，除了介绍语言特性，更着重于讲解语言背后的原理和思想。

本书内容较为深入，不太适合初学者，读者至少要具备基本的编程知识，最好是接触过其他编程语言且对 Dart 有基本了解。

版权贸易合同登记号　图字：01-2016-0601

图书在版编目（CIP）数据

Dart 编程语言 /（美）吉拉德·布拉查（Gilad Bracha）著；戴虬译. —北京：电子工业出版社，2017.6
书名原文：The Dart Programming Language
ISBN 978-7-121-31511-4

Ⅰ. ①D… Ⅱ. ①吉… ②戴… Ⅲ. ①程序语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2017)第 105153 号

策划编辑：张春雨　付　睿
责任编辑：徐津平
印　　刷：三河市华成印务有限公司
装　　订：三河市华成印务有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×980　1/16　印张：13.25　字数：255 千字
版　　次：2017 年 6 月第 1 版
印　　次：2017 年 6 月第 1 次印刷
定　　价：69.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819，faq@phei.com.cn。

感谢我的母亲，Shoshana，

她教导我做事要力争精益求精。

译者序

Dart 是一门由 Google 开发并被批准为 ECMA 标准（ECMA-408）的通用编程语言。它可以用于构建 Web 端、服务端和移动端应用程序。它是开源的，使用宽松的免费软件许可证（修改版 BSD 许可证）。Dart 是完全面向对象的，使用类和单继承，可编译为 JavaScript，语法风格是类 C 的。它支持接口、mixin、抽象类、泛型和可选类型。

Dart 目前在业界的认识度不高，社区规模也比较小，主要原因是 Dart 自身还在不断地发展、完善中，而 Google 对外还没有真正地对 Dart 做过推广。幸好，近两年 Dart 的发展开始步入正轨。Dart 是 2016 年 Google 内部使用量增长最快的编程语言，多个重要项目如 AdSense、AdWords 等 Web 应用，正在用 Dart 重写。Google 正在研发中的跨平台移动端开发工具 Flutter，也正在使用 Dart。Dart 未来的发展值得我们期待。

本书虽然名为《Dart 编程语言》，且内容也涉及了对象、类型、表达式与语句等基础知识，但它总体还是着重于讲解语言背后的原理和思想。这意味着本书并不适合真正的初学者，读者至少要具备基本的编程知识，最好对 Dart 有基本的了解且使用过一门编程语言，例如 JavaScript、Java 或 C#等。

作为一门新生的可选类型编程语言，Dart 目前的关注者比较少，可参考的资料也非常匮乏，而本书内容具有一定深度，加上笔者水平有限，翻译时出现错误或偏差在所难免，欢迎读者朋友指正。

感谢出版社编辑的校对，特别感谢付睿编辑，让我得到了翻译（学习）本书的机会。翻译工作耗费了我大量的业余时间。也感谢家人的支持与理解。

推荐序

在 2006 年的早春时节，我在博客上写了一篇简短的文章《Gilad 是正确的》，主要内容是表述作为一名静态类型主义者，我认同 Gilad 关于可选和分层类型系统的想法，静态类型不能改变程序的运行时行为，不阻止非法程序的编译或执行，是面向数百万开发人员的编程语言必然要做出的设计权衡。当时我正在忙于学习 Visual Basic 语言，它通过 Option Strict Off 语句已经可以支持可选类型，但该特性受到静态类型支持者的猛烈抨击。类型系统通常是高度非线性的，当它成长至某一节点之后，其复杂性将呈爆炸性增长，带给开发者的价值却很少，还使语言实现者的生活变得暗淡无光。可选和分层类型系统通过允许强制静态类型与动态类型和平共存来实现一种更加缓和的处理方式。现在，近十年后，Gilad 开创性的愿景已经化名为渐进类型并逐步成为主流。过去几年中所诞生的许多编程语言，如 Hack、TypeScript、Flow、Racket 和 Dart，都选择了渐进类型。即使是学术界，也接受了这个想法，并用包含如“三人行”和“责备”等轻蔑字眼的标题，来编写与之相关的论文。

Dart 另一个务实且没有被语言纯粹主义者所接受的事实是，Dart 的类型系统被故意设计为非严格的。用正常的语言来描述，这意味着 Dart 类型检查器在编译时不会标记某些类型错误，反而依靠运行时检查来确保类型安全。Dart 中类型非严格的主要来源是协变泛型。为了解释其中的分歧，让我们先看看一台自动售货机，我们只能从中取饮料。如果自助餐厅需要售卖汽水的自动售货机，则我们可以合法地安装售卖根汁汽水的自动售货机，因为根汁汽水是一种汽水（但是在需要售卖根汁汽水的自动售货机的地方，安装售卖汽水的自动售货机则是非法的）。在编程语言中，我们说自动售货机是协变的。接下来让我们看看只能用来装填垃圾的垃圾桶。如果自助餐厅需要一个垃圾桶来回收垃圾，则我们可以合法安装一个普通的垃圾桶，因为可回收垃圾是垃圾的一种（但对于需要普通垃圾桶的地方，安装可回收垃圾桶是非法的）。在编程语言中，我们说垃圾桶是逆变的。你不是唯一对协

变和逆变感到困惑的人，你会欣赏 Dart 的决定，使所有的泛型类型协变。这种选择的后果是，如果你需要普通垃圾桶，则你可以合法安装一个用于可回收垃圾的垃圾桶，但该垃圾桶会拒绝回收人们向其中投入的所有不可回收的垃圾。虽然理论上非严格，但对于大多数开发人员来说，不安全的协变实际上是很自然的，我赞赏 Dart 设计者在这里做出的选择。作为曾经挣扎于选择 **super** 或 **extends** 的人可以证明，选择支持静态类型安全的泛型语言，其付出的代价是建立在减少用户之上的。

Dart 语言设计者做出的其他务实选择，使 Dart 的编码过程变得流畅。例如，Dart 没有接口、抽象基类或“普通”类。相反，Dart 只有作为接口的类，你可以实现它们，或者通过继承它们来作为基类使用，或者通过 mixin 来重用它们的实现。Dart 中的每个类型都是一个对象，所以基础类型（如数字）和常规对象类型是没有区别的。虽然 Dart 中的所有内容都是对象，但定义顶级函数和变量也是可以的，所以顶层类中不再需要可怕的 public static void 咒语了。Dart 允许用户自定义算术运算符，但不支持基于类型的方法重载，这大大简化了语言本身。其他支持基于类型重载的编程语言，它们的语言规范花费了大量不必要的篇幅来描述此特征的语义。null 感知运算符（null 也是一个普通的对象）和级联为点操作符赋予了更多的能力，也使 API 的作者不费吹灰之力就能编写出可让用户流畅使用的 API。

因为所有类型都是可选的，所以 Dart 本质上是一种动态语言，虽然如此，与其他大多数动态语言相比，你很少有机会产生疑问。有 null 但没有 undefined，因此只有==但没有===。只有 true 是 true，所以也不需要使用 (foo && foo.bar()) 来检查 null。Dart 有常见的整数和浮点数字类型，但是+和==不会产生令人惊讶的运行时类型转换，类型转换可能会是伟大的考试问题或有趣的会议演示，但都会带来令人沮丧的错误。

在我看来，虽然我明显存在偏见，但 Dart 成为我最喜欢的编程语言的原因是，它是我知道的唯一支持以下四种模式的编程语言。

	One	Many
sync	{... **return** *e*; ... }	**sync*** {... **yield** *e*; ... **for**() ... }
async	**async** {... **await** *e* ... }	**async*** {... **await** *e* ... **yield** *e*; ... } **async*** {... **await for**() ... }

也就是说，Dart 通过使用 sync*块中的生成器和 for 循环，对同步数据流（Iterable <T>）的生成和使用提供良好的支持，使用 async 块中的 await 表达式，生成和使用 future（Future <T>），最后且最重要的是，使用 async*块中的异步生成器和 for 循环生成及使用异步数据流（Stream <T>）。内置对异步编程的支持在任何现代编程语言中都是必不可少的，虽然数据存在于内存中，但让它们在网络上传输，那是非常“遥远”的，在如此高延迟下进行

同步访问的代价极高。类似于 JavaScript，但又不同于其他支持生成器的语言，Dart 拥有所谓的委派生成器，避免流的生成在嵌套和递归中呈现二次方规模的爆发性增长。

尽管有以上诸多优点，Dart 在大体上是一门被设计得比较无趣的语言。由于支持 getter、setter、lambda、enum、reified generic、模块、一个精心设计的标准库和一个简单的包管理器，如果你用过 Java 或 C#，则使用 Dart 会让你感觉非常舒适，就像一双合脚的鞋子；如果你用过 JavaScript，那么你会觉得 Dart 像新鲜的空气。本书将帮助你了解 Dart 的所有特性，了解为什么需要这些特性及它们是如何实现的，Gilad 对细节的详细讲解和其独特的方式，可让你在短时间内熟悉 Dart。

Erik Meijer
加利福尼亚州 帕罗奥图
2015 年 10 月

前言

本书与其他 Dart 书籍有什么不同？其他 Dart 书籍都注重于实操，本书着重于讲解原理和思想。

Dart 的实操非常重要，但它们可能会逐年变化，并可能一直变化下去。相比之下，Dart 背后的原理应该很少会随时间而改变。如果你对激发 Dart 语言设计的想法及它们是如何实现的，还有如何权衡现实世界对 Dart 的需求等感兴趣，那么你应该阅读本书。

Dart 的主要思想之一是可选类型。我在几十年前就开始从事可选类型的工作，今天我们看到使用可选类型系统的语言在飞速增加，我对此感到非常欣慰。虽然 Dart 和它的任何竞争对手都不像我想要的那样实现可选类型，但可选类型成为主流的事实才是最重要的。

更重要的思想是，Dart 是一种面向对象的语言，这不等同于传统意义上的类、继承或其他大多数开发者所理解的概念，它的深层含义可以表述为，一个对象最重要的是它可观察的行为。同样，Dart 对这个想法并没有完美地实现，但比大多数主流语言要好。

本书中涉及的另一个关键思想是反射。大部分关于编程的书籍都没有很好地讨论反射。因此，我非常渴望在本书中探讨反射。然而，Dart 的反射历史却意外地曲折。

一方面，很多 Dart 用户都渴望使用反射，有时只是为了用而用，但并不一定适用。另一方面，某些作为 Dart 编译目标的平台存在严重限制，使得对反射的支持异常昂贵，尤其是需要考虑代码的体积。这种双向压力使 Dart 的反射陷入尴尬的境地。

我们从一开始就知道代码体积和反射对其影响的敏感性。这个问题及其解决方案在 2011 年 11 月的第 1 个 Dart 反射设计文档中被讨论。然而，直到实现解决方案且开发者可以轻松使用时，已经花费了四年。

我希望本书有效地传达上述及其他语言设计相关的想法，但它们的好坏由读者来判断。我们可能可以创造一门更加纯粹的语言，并做得更好，但另一方面，我们也不清楚是否可以得到大家的认可。也许有一天我会进行那样的尝试。

本书的写作历时很久。直到我能完整讲述一个关于反射的合理故事，我才完成本书。拖延本书写作的另一个原因是，本书的主题发展得如此之快，以至于它总是面临过时的风险。这种风险还没有过去，但在某个时刻我们需要说“适可而止”。

Dart 并没有完美实现驱动其设计的思想。没有人比它的设计者更了解这个事实。然而，它是一门真正的语言，已经被用来书写数百万行的关键任务代码。它以某种方式推动了编程文化的发展，最引人注目的是在可选类型领域。正如有人在丹麦说的：“它本可能更糟”。

致谢

Dart 编程语言是大团队努力的结果。Dart 的开发工作让我收获许多快乐，我需要跟许多同事合作，其中有很多人更可以称为朋友，跟他们合作是一件非常愉快的事情。我非常荣幸地在本书及 Dart 语言规范中列举部分曾经和我合作过的同事。

Dart 语言是由 Lars Bak 和 Kasper Lund 构思和设计的，所以没有他们，本书就不可能存在。Lars 是我长期的朋友和同事，从一开始就管理 Dart 项目。除此之外，Lars 让我参与 Dart 的设计工作，为此，我特别感谢他。Lars 和 Kasper 不仅是非常有天赋的系统设计师和实现者，而且很好相处！

特别感谢 Erik Meijer，和他共同开发 Dart 的异步功能是一种快乐。Erik 是一名真正的编程语言专家，且属于非常少见的那种。

如果本书第 8 章的完成需要感谢 Erik，那么第 7 章的完成就需要感谢 Ryan Macnak、Peter Ahé、Erik Ernst 的帮助了，他们分别实现了 Dart VM 中 mirrors、dart2js 中的 mirrors 及 reflectable 库。

本书的写作，不仅得到了 Lars 的支持，也得到了我的经理 Ivan Posva 的支持。在过去的四年里，我与 Ivan 和 VM 团队的其他成员共享办公空间，他们是：Zachary Anderson、Daniel Andersson、Siva Annamalai、Régis Crelier、Matthias Hausner、Ryan Macnak、John McCutchan、Srdjan Mitrovic 和 Todd Turnidge。感谢他们的愉快陪伴。

我经常拜访位于丹麦奥胡斯的 Dart 总部，其过程充满乐趣（即使实际旅途并不有趣）。Linda Lykke Rasmussen 的行政支持是无价的。

我对 Dart 语言规范的编写工作是本书的直接基础来源。这项工作得益于许多人的建议，但只有 Lasse Nielsen 对细节的不寻常的觉察，使他捕捉到了许多微妙的问题。

我也参与了 Dart 的标准化工作，Anders Sandholm 帮我承担了其中的大部分工作，我对他表示感谢。我还要感谢 Dart 标准委员会 ECMA TC52 的其他参与者。

如果没有其他 Dart 团队成员一直以来的工作，Dart 不可能有今天。人很多，这里不便一一列出，但他们都为 Dart 的今天付出了努力。

和我长期合作的编辑 Greg Doench 一直非常耐心，我很高兴能与他合作。

和往常一样，我的妻子 Weihong 和我的儿子 Teva，让这一切都是值得的。

Gilad Bracha
加利福尼亚州 洛思阿图斯
2015 年 11 月

目录