




[BT](#)

[如何利用碎片时间提升技术认知与能力？ 点击获取答案](#)

- [投稿](#)
- [活动大本营](#)
- [InfoQ手机客户端](#)
- [关于我们](#)
- [合作伙伴](#)

- 欢迎关注我们的：
- 
- 
- 

InfoQ – 促进软件开发领域知识与创新的传播

[tree](#)

欢迎， tree ！

- [账号配置](#)
- [我的阅读清单](#)
- [偏好设置](#)
- [您关注的主题](#)
- [您关注的同行](#)
- [编辑之选](#)

[注销](#)

1



- [En](#)
- [中文](#)
- [日本](#)
- [Fr](#)
- [Br](#)

966,690 二月 独立访问用户

- [语言 & 开发](#)
 - [Java](#)
 - [Clojure](#)
 - [Scala](#)
 - [.Net](#)
 - [移动](#)
 - [Android](#)
 - [iOS](#)
 - [HTML 5](#)
 - [JavaScript](#)
 - [函数式编程](#)
 - [Web API](#)

特别专题 语言 & 开发

高可用低延时的PayPal风控数据平台



作为全球领先的互联网支付公司，PayPal支持用户可以安全地在 200 多个国家和地区间进行在线交易。每天有超过2000万笔交易在PayPal平台发生。PayPal风险控制平台需要实时高效地对每一笔交易进行风险评估。随着业务的持续发展，风控所需的数据规模持续增长，数据类型纷繁多样，这就对风控数据平台的高可用和低延迟提出了更大的挑战。PayPal风控数据平台团队基于对业务的分析，深入研究各种...

[浏览所有 语言 & 开发](#)

- [架构 & 设计](#)
 - [架构](#)
 - [企业架构](#)
 - [性能和可伸缩性](#)
 - [设计](#)
 - [案例分析](#)
 - [设计模式](#)
 - [安全](#)

特别专题 架构 & 设计

[Netflix最新视频优化实践：用更少的带宽打造完美画质](#)



[本文介绍了Netflix视频算法团队通过改进的视频编码方式和算法改善视频质量，降低视频码率，并将其全面应用到生产环境中，借此提高用户观影体验的做法和实践。](#)

[浏览所有 架构 & 设计](#)

- [数据科学](#)
 - [大数据](#)
 - [NoSQL](#)
 - [数据库](#)

特别专题 数据科学

[从批处理ETL到流式处理：一个来自Netflix的案例](#)



[在2017年的纽约QCon大会上，Shriya Arora呈现了“Personalizing Netflix with Streaming Datasets”的演讲，分享了Netflix的一个数据作业迁移案例，他们使用Flink替代了原先基于批处理的ETL。](#)

[浏览所有 数据科学](#)

- [文化 & 方法](#)
 - [Agile](#)
 - [领导能力](#)
 - [团队协作](#)
 - [测试](#)
 - [用户体验](#)
 - [Scrum](#)
 - [精益](#)

特别专题 文化 & 方法

[从Instagram到Reddit，浅谈西方工程师文化和管理](#)



[Instagram为什么会这么成功？技术和管理在这里发挥了什么样的作用？他们的局限又是什么？当Instagram被一次次纳入斯坦福商学院研究案例的时候，这里面有没有不为人知的危机时刻？Reddit是一个什么样的公司？管理层的好坏如何决定了公司的命脉？本次演讲主要从工程师管理的角度，谈谈产品、文化和团队建设在西方文化下的一些个人见解、心得收获。](#)

[浏览所有 文化 & 方法](#)

- [DevOps](#)
 - [持续交付](#)
 - [自动化操作](#)
 - [云计算](#)

特别专题 DevOps

如何调试分布式系统：与微服务调试工具“Squash”创始人Idit Levine的对话



近期，InfoQ与solo.io的CEO Idit Levine进行了一次座谈。Idit新创立了一种称为“Squash”的开源微服务调试器。在此次座谈中，她介绍了观察和调试分布式系统和应用中的挑战。

[浏览所有 DevOps](#)

InfoQ手机客户端

[架构](#)

[移动](#)

[运维](#)

[云计算](#)

[AI前线](#)

[大数据](#)

[前端](#)

[QCon合集](#)

[ArchSummit](#)

[百度](#)

全部话题

您目前处于：[InfoQ首页](#) [文章](#) 为什么Flutter会选择 Dart ?

为什么Flutter会选择 Dart ?



喜欢

/ 作者 [Wm Leler](#)，译者 [王纯超](#) 发布于 2018年3月14日. 估计阅读时间: 26 分钟 / [QCon北京站9折倒计时](#)，人工智能、区块链、大数据、架构等领域海内外先驱实践！ [1 讨论](#)

分享到：[微博](#) [微信](#) [Facebook](#) [Twitter](#) [有道云笔记](#) [邮件分享](#)

- ["稍后阅读"](#)
- ["我的阅读清单"](#)



亲爱的读者：我们最近添加了一些个人信息定制功能，您只需选择感兴趣的技术主题，即可获取重要资讯的[邮件和网页通知](#)。

许多语言学家认为，一个人说的自然语言会[影响他们的思维方式](#)。这个理论适用于[计算机语言](#)吗？使用不同编程语言编程的程序员针对问题想出的解决方案经常完全不同。举一个极端的例子，为了程序结构更加清晰，计算机科学家[取消了goto语句](#)（这与小说[《1984》](#)中的极权主义领导者从自然语言中删除异端词语以消除[思维犯罪](#)不太一样，但道理就是这样）。

这与[Flutter](#)和[Dart](#)有什么关系？确实有关系。早期的Flutter团队评估了十多种语言，并选择了Dart，因为它符合他们构建用户界面的方式。

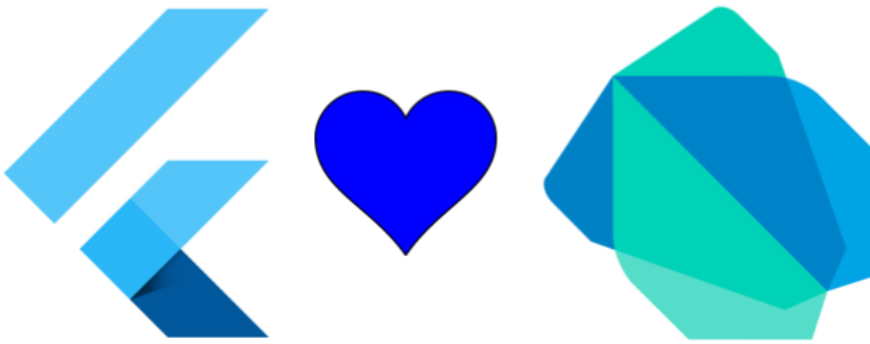
Dart是开发人员喜欢Flutter的一大原因。如以下[推文](#)：

[@flutterio](#) got me to look at [@dart](#), and I'm glad I took it for a spin. [#Dart](#) is an awesome language, and [#flutterio](#) takes it even further, to mobile devices <3

以下是使Dart成为Flutter不可或缺的一部分的特性：

- Dart是AOT（Ahead Of Time）编译的，编译成快速、可预测的本地代码，使Flutter几乎都可以使用Dart编写。这不仅使Flutter变得更快，而且几乎所有的东西（包括所有的小部件）都可以定制。
- Dart也可以JIT（Just In Time）编译，开发周期异常快，工作流颠覆常规（包括Flutter流行的亚秒级有状态热重载）。
- Dart可以更轻松地创建以60fps运行的流畅动画和转场。Dart可以在没有锁的情况下进行对象分配和垃圾回收。就像JavaScript一样，Dart避免了抢占式调度和共享内存（因而不需要锁）。由于Flutter应用程序被编译为本地代码，因此它们不需要在领域之间建立缓慢的桥梁（例如，JavaScript到本地代码）。它的启动速度也快得多。
- Dart使Flutter不需要单独的声明式布局语言，如JSX或XML，或单独的可视化界面构建器，因为Dart的声明式编程布局易于阅读和可视化。所有的布局使用一种语言，聚集在一处，Flutter很容易提供高级工具，使布局更简单。
- 开发人员发现Dart特别容易学习，因为它具有静态和动态语言用户都熟悉的特性。

并非所有这些功能都是Dart独有的，但它们的组合却恰到好处，使Dart在实现Flutter方面独一无二。因此，没有Dart，很难想象Flutter像现在这样强大。



本文接下来将深入探讨使Dart成为实现Flutter的最佳语言的许多特性（包括其标准库）。

编译和执行

[如果你已经了解静态语言与动态语言、AOT与JIT编译以及虚拟机等主题，可以跳过本节。]

历史上，计算机语言分为两组：[静态语言](#)（例如，Fortran和C，其中变量类型是在编译时静态指定的）和[动态语言](#)（例如，Smalltalk和JavaScript，其中变量的类型可以在运行时改变）。静态语言通常编译成目标机器的本地机器代码（或汇编代码）程序，该程序在运行时直接由硬件执行。动态语言由解释器执行，不产生机器语言代码。

当然，事情后来变得复杂得多。[虚拟机](#)（VM）的概念开始流行，它其实只是一个高级的解释器，用软件模拟硬件设备。虚拟机使语言移植到新的硬件平台更容易。因此，VM的输入语言常常是[中间语言](#)。例如，一种编程语言（如[Java](#)）被编译成中间语言（[字节码](#)），然后在VM（[JVM](#)）中执行。

另外，现在有[即时（JIT）编译器](#)。JIT编译器在程序执行期间运行，即时编译代码。原先在程序创建期间（运行时之前）执行的编译器现在称为[AOT编译器](#)。

一般来说，只有静态语言才适合AOT编译为本地机器代码，因为机器语言通常需要知道数据的类型，而动态语言中的类型事先并不确定。因此，动态语言通常被解释或JIT编译。

在开发过程中AOT编译，开发周期（从更改程序到能够执行程序以查看更改结果的时间）总是很慢。但是AOT编译产生的程序可以更可预测地执行，并且运行时不需要停下来分析和编译。AOT编译的程序也更快地开始执行（因为它们已经被编译）。

相反，JIT编译提供了更快的开发周期，但可能导致执行速度较慢或时快时慢。特别是，JIT编译器启动较慢，因为当程序开始运行时，JIT编译器必须在代码执行之前进行分析和编译。研究表明，[如果开始执行需要超过几秒钟，许多人将放弃应用](#)。

以上就是背景知识。将AOT和JIT编译的优点结合起来不是很棒吗？请继续阅读。

编译与执行Dart

在创造Dart之前，Dart团队成员在高级编译器和虚拟机上做了开创性的工作，包括动态语言（如JavaScript的[V8引擎](#)和Smalltalk的[Strongtalk](#)）以及静态语言（如用于Java的[Hotspot编译器](#)）。他们利用这些经验使Dart在编译和执行方面非常灵活。

Dart是同时非常适合AOT编译和JIT编译的少数语言之一（也许是唯一的“主流”语言）。支持这两种编译方式为Dart和（特别是）Flutter提供了显著的优势。

JIT编译在开发过程中使用，编译器速度特别快。然后，当一个应用程序准备发布时，它被AOT编译。因此，借助先进的工具和编译器，Dart具有两全其美的优势：极快的开发周期、快速的执行速度和极短启动时间。

Dart在编译和执行方面的灵活性并不止于此。例如，Dart可以[编译成JavaScript](#)，所以浏览器可以执行。这允许在移动应用和网络应用之间重复使用代码。开发人员报告他们的移动和网络应用程序之间的[代码重用率高达70%](#)。通过将Dart编译为本地代码，或者编译为JavaScript并将其与[node.js](#)一起使用，Dart也可以在服务器上使用。

最后，[Dart还提供了一个独立的虚拟机](#)（本质上就像解释器一样），虚拟机使用Dart语言本身作为其中间语言。

Dart可以进行高效的AOT编译或JIT编译、解释或转译成其他语言。Dart编译和执行不仅非常灵活，而且速度**特别快**。

下一节将介绍Dart编译速度的颠覆性的例子。

有状态热重载

Flutter最受欢迎的功能之一是其极速**热重载**。在开发过程中，Flutter使用JIT编译器，通常可以在一秒之内重新加载并继续执行代码。只要有可能，应用程序状态在重新加载时保留下来，以便应用程序可以从停止的地方继续。

除非自己亲身体验过，否则很难理解在开发过程中快速（且可靠）的热重载的重要性。开发人员报告称，它改变了他们创建应用的方式，将其描述为[像将应用绘制成生活](#)一样。

[以下是一位移动应用程序开发人员对Flutter热重载的评价](#)：

我想测试热重载，所以我改变了颜色，保存修改，结果.....就喜欢上它了♥！

这个功能真的很棒。我曾认为Visual Studio中**编辑和继续**（Edit & Continue）很好用，但这简直令人惊叹。有了这个功能，我认为移动开发者的生产力可以提高两倍。

这对我来说真的是翻天覆地的变化。当我部署代码并花费很长时间时，我分心了，做了其他事情，当我回到模拟器/设备时，我就忘了想测试的内容。有什么比花5分钟将控件移动2px更令人沮丧？有了Flutter，这不再存在。

Flutter的热重载也使得尝试新想法或尝试替代方案变得更加容易，从而为创意提供了巨大的推动力。

到目前为止，我们讨论了Dart给开发人员带来的好处。下一节将介绍Dart如何使创建满足用户需求的顺畅的应用程序更加轻松。

避免卡顿

应用程序**速度快**很不错，但**流畅**则更加了不起。即使是一个超快的动画，如果它不稳定，也会看起来很糟糕。但是，防止[卡顿](#)可能很困难，因为因素太多。Dart有许多功能可以避免许多常见的导致卡顿的因素。

当然，像任何语言一样，Flutter也可能写出来卡顿的应用程序；Dart通过提高可预测性，帮助开发人员更好地控制应用程序的流畅性，从而更轻松地提供最佳的用户体验。

[效果怎样呢？](#)

以60fps运行，使用Flutter创建的用户界面的性能远远优于使用其他跨平台开发框架创建的用户界面。

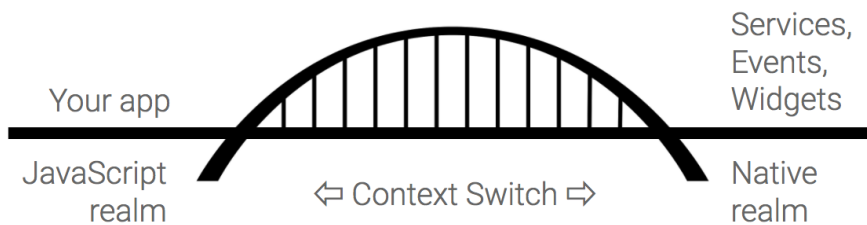
不仅仅比跨平台的应用程序好，而且和最好的原生应用程序一样好：

UI像黄油一样顺滑.....我从来没有见过这样流畅的Android应用程序。

AOT编译和“桥”

我们讨论过一个有助于保持顺畅的特性，那就是Dart能AOT编译为本地机器码。预编译的AOT代码比JIT更具可预测性，因为在运行时不需要暂停执行JIT分析或编译。

然而，AOT编译代码还有一个更大的优势，那就是避免了“JavaScript桥梁”。当动态语言（如JavaScript）需要与平台上的本地代码互操作时，[它们必须通过桥进行通信](#)，这会导致[上下文切换](#)，从而必须保存特别多的状态（可能会存储到辅助存储）。这些[上下文切换](#)具有双重打击，因为它们不仅会减慢速度，还会导致严重的卡顿。



注意：即使编译后的代码也可能需要一个接口来与平台代码进行交互，并且这也可以称为桥，但它通常比动态语言所需的桥快几个数量级。另外，由于Dart允许将小部件等内容移至应用程序中，因此减少了桥接的需求。

抢占式调度、时间分片和共享资源

大多数支持多个并发执行线程的计算机语言（包括Java、Kotlin、Objective-C和Swift）都使用[抢占式](#)来切换线程。每个线程都被分配一个时间分片来执行，如果超过了分配的时间，线程将被上下文切换抢占。但是，如果在线程间共享的资源（如内存）正在更新时发生抢占，则会导致[竞态条件](#)。

竞态条件具有双重不利，因为它可能会导致严重的错误，包括应用程序崩溃并导致数据丢失，而且由于它取决于[独立线程的时序](#)，所以它特别难以找到并修复。在调试器中运行应用程序时，竞态条件常常消失不见。

解决竞态条件的典型方法是使用[锁](#)来保护共享资源，阻止其他线程执行，但锁本身可能导致卡顿，甚至[更严重的问题](#)（包括[死锁](#)和[饥饿](#)）。

Dart采取了不同的方法来解决这个问题。Dart中的线程称为isolate，不共享内存，从而避免了大多数锁。isolate通过在通道上传递消息来通信，这与[Erlang](#)中的actor或JavaScript中的Web Worker相似。

Dart与JavaScript一样，是[单线程](#)的，这意味着它根本不允许抢占。相反，线程显式让出（使用[async/await](#)、[Future](#)和[Stream](#)）CPU。这使开发人员能够更好地控制执行。单线程有助于开发人员确保关键功能（包括动画和转场）完成而无需抢占。这通常不仅是用户界面的一大优势，而且还是客户端——服务器代码的一大优势。

当然，如果开发人员忘记了让出CPU的控制权，这可能会延迟其他代码的执行。然而我们发现，忘记让出CPU通常比忘记加锁更容易找到和修复（因为竞态条件很难找到）。

对象分配和垃圾回收

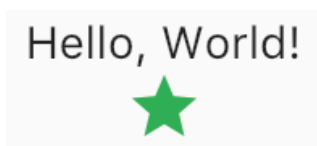
另一个严重导致卡顿的原因是垃圾回收。事实上，这只是访问共享资源（内存）的一种特殊情况，在很多语言中都需要使用锁。但在回收可用内存时，[锁会阻止整个应用程序运行](#)。但是，Dart几乎可以在没有锁的情况下执行垃圾回收。

Dart使用先进的[分代垃圾回收和对象分配方案](#)，该方案对于分配许多短暂的对象（对于Flutter这样的反应式用户界面来说非常完美，Flutter为每帧重建不可变视图树）都特别快速。Dart可以用一个指针凹凸分配一个对象（不需要锁）。这也会带来流畅的滚动和动画效果，而不会出现卡顿。

统一的布局

Dart的另一个好处是，Flutter不会从程序中拆分出额外的模板或布局语言，如JSX或XML，也不需要单独的可视布局工具。以下是一个简单的Flutter视图，用Dart编写：

```
new Center(child:
  new Column(children: [
    new Text('Hello, World!'),
    new Icon(Icons.star, color: Colors.green),
  ])
)
```



Dart编写的视图及其效果

注意，可视化这段代码产生的效果是多么容易（即使你没有使用Dart的经验）。

[Dart 2](#)即将发布，这将变得更加简单，因为`new`和`const`关键字变得可选，所以静态布局看起来像是用声明式布局语言编写的：

```
Center(child:
  Column(children: [
    Text('Hello, World!'),
    Icon(Icons.star, color: Colors.green),
  ])
)
```

然而，我知道你可能在想什么——缺乏专门的布局语言怎么会被称为优势呢？但它确实是颠覆性的。以下是一名开发人员在一篇题为“[为什么原生应用程序开发人员应认真看待Flutter](#)”的文章中写的内容。

在Flutter里，界面布局直接通过Dart编码来定义，不需要使用XML或模板语言，也不需要使用可视化设计器之类的工具。

说到这里，大家可能会一脸茫然，就像我当初的反应一样。使用可视化工具不是更容易吗？如果把所有的逻辑都写到代码里不是会让事情变复杂吗？

结果不然。天啊，它简直让我大开眼界。

首先是上面提到的热重载。

这比Android的Instant Run和任何类似解决方案不知道要领先多少年。对于大型的应用同样适用。如此快的速度，正是Dart的优势所在。

实际上，可视化编辑器就变得多余了。我一点都不怀恋XCode的自动重布局。

Dart创建的布局简洁且易于理解，而“超快”的热重载可立即看到结果。这包括布局的非静态部分。

结果，在Flutter中进行布局要比在Android/XCode中快得多。一旦你掌握了它（我花了几个星期），由于很少发生上下文切换，因此会节省大量的开销。不必切换到设计模式，选择鼠标并开始点击，然后想是否有些东西必须通过编程来完成，如何实现等等。因为一切都是程序化的。而且这些API设计得非常好。它很直观，并且比自动布局XML更强大。

例如，下面是一个简单的列表布局，在每个项目之间添加一个分隔线（水平线），以编程方式定义：

```
return new ListView.builder(itemBuilder: (context, i) {
  if (i.isOdd) return new Divider();
  // rest of function
});
```

在Flutter中，无论是静态布局还是编程布局，所有布局都存在于同一个位置。[新的Dart工具](#)，包括Flutter Inspector和大纲视图（利用所有的布局定义都在代码里）使复杂而美观的布局更加容易。

Dart是专有语言吗？

不，Dart（如Flutter）是完全开源的，具备清楚的许可证，同时也是[ECMA标准](#)的。Dart在Google内外很受欢迎。在谷歌内部，它是增长最快的语言之一，并被Adwords、Flutter、[Fuchsia](#)和其他产品使用；在谷歌外部，Dart代码库有超过100个外部提交者。

Dart开放性的更好指标是Google之外的社区的发展。例如，我们看到来自第三方的关于Dart（包括Flutter和AngularDart）的文章和视频源源不断，我在本文中引用了其中的一些内容。

除了Dart本身的外部提交者之外，[公共Dart包仓库](#)中还有超过3000个包，其中包括Firebase、Redux、RxDart、国际化、加密、数据库、路由、集合等方面的库。

Dart程序员难找吗？

如果没有很多程序员知道Dart，找到合格的程序员会困难吗？显然不是。Dart是一门难以置信的易学语言。事实上，已经了解Java、JavaScript、Kotlin、C# 或Swift等语言的程序员几乎可以立即开始使用Dart进行编程。

一个程序员在名为“[为什么Flutter 2018年将起飞](#)”的文章中写到：

[Dart](#)是用于开发Flutter应用程序的语言，很易学。谷歌在创建简单、有文档记录的语言方面拥有丰富的经验，如Go。到目前为止，对我来说，Dart让我想起了Ruby，很高兴能够学习它。它不仅适用于移动开发，也[适用于Web](#)

开发。

另一篇关于Flutter和Dart的文章，题为“[为什么是Flutter而不是其他框架？](#)”

Flutter使用由Google创建的Dart语言，老实说，我不喜欢C#或JAVA这样的强类型语言，但我不知道Dart编写代码的方式有什么与众不同。但我觉得写起来很舒服。也许是因为它非常简单易学，而且非常直观。

Dart通过广泛的用户体验研究和测试，专门设计得熟悉并易于学习。例如，在2017年上半年，Flutter团队与八位开发人员一起进行了[用户体验研究](#)。我们给他们简短地介绍了Flutter，然后给他们一个小时左右，创建了一个简单的视图。所有参与者都能够立即开始编程，即使他们以前从未使用过Dart。他们专注于写响应式视图，而不是语言。[Dart直接就能上手用了](#)。

最后，一位参与者（在任务中进展得特别快）没有提及任何有关该语言的内容，所以我们问他是否知道他正在使用哪种语言。他说不知道。语言不成问题；他在几分钟内就能用Dart编程。

学习新系统的难点通常不是学习语言，而是学习编写好代码的所有库、框架、工具、模式和最佳实践。Dart库和工具格外出色，并且文档详尽。[有一篇文章宣称](#)：“意外之喜是，他们还极其爱护代码库，并且他们拥有我见过的最好的文档。”花费在学习Dart上的时间很容易通过学习其他东西节省的时间弥补。

作为直接证据，Google内部的一个大型项目希望将其移动应用程序移植到iOS。他们即将聘请一些iOS程序员，但转而决定尝试Flutter。他们监测了让开发者上手Flutter需要多长时间。结果表明，程序员可以学会Dart和Flutter，并在三周内达到高效率。相比之下，他们之前观察到仅仅让程序员上手Android（更不用说他们必须聘用和培训iOS开发人员）需要五个星期。

最后，一家将三种平台（iOS、Android和Web）上的大型企业应用程序都迁移到Dart的公司，有一篇文章“[我们为什么选择Flutter以及它如何改变我们的公司](#)”。他们的结论：

招人变得容易多了。无论他们是来自Web、iOS还是Android，我们现在都希望接受最佳人选。

现在我们拥有3倍的工作效率，因为我们所有的团队都集中在一个代码库上。

知识共享达到前所未有的高度。

使用Dart和Flutter使他们的生产力提高到三倍。考虑到他们以前在做什么，这应该不会令人感到意外。与许多公司一样，它们利用不同的语言、工具和程序员为每个平台（Web、iOS和Android）构建独立的应用程序。切换到Dart意味着他们不再需要雇佣三种不同的程序员。而且他们很容易将现有的程序员转移到使用Dart。

他们和其他人发现，一旦程序员开始使用Flutter，他们就会[爱上Dart](#)。他们喜欢Dart的简洁和缺乏仪式。他们喜欢级联、命名参数、async/await和Stream等[语言特性](#)。而最重要的是，他们喜欢Dart带来的Flutter功能（如热重载），以及Dart帮助他们构建的美丽、高性能的应用程序。

Dart 2

在本文发表时，[Dart 2](#)正在发布。Dart 2专注于[改善构建客户端应用程序的体验](#)，包括加快开发人员速度、改进开发人员工具和类型安全。例如，Dart 2具有[坚实的类型系统](#)和[类型推理](#)。

Dart 2还使new和const关键字可选。这意味着可以在不使用任何关键字的情况下描述Flutter视图，从而减少混乱并且易于阅读。例如：

```
Widget build(BuildContext context) =>
  Container(
    height: 56.0,
    padding: EdgeInsets.symmetric(horizontal: 8.0),
    decoration: BoxDecoration(color: Colors.blue[500]),
    child: Row(
      ...
    ),
  );
```

Dart 2自动计算出所有的构造函数，并且“padding:”的值是一个常量。

秘诀在于专注

Dart 2的改进集中在优化客户端开发。但Dart仍然是构建服务器端、桌面、嵌入式系统和其他程序的绝佳语言。

专注是一件好事。几乎所有持久受欢迎的语言都受益于非常专注。例如：

- C是编写操作系统和编译器的系统编程语言。

- Java是为嵌入式系统设计的语言。
- JavaScript是网页浏览器的脚本语言。
- 即使是饱受非议的PHP也成功了，因为它专注于编写个人主页（它的名字来源）。

另一方面，许多语言已经明确地尝试过（并且失败了）成为完全是通用的，例如PL/1和Ada等等。最常见的问题是，如果没有重点，这些语言就成了众所周知的厨房洗碗槽。

许多使Dart成为好的客户端语言的特性也使其成为更好的服务器端语言。例如，Dart避免了抢占式多任务处理，这一点与服务端上的Node具有相同的优点，但是数据类型更好更安全。

编写用于嵌入式系统的软件也是一样的。Dart能够可靠地处理多个并发输入是关键。

最后，Dart在客户端上的成功将不可避免地引起用户对服务器上使用的更多兴趣——就像JavaScript和Node一样。为什么强迫人们使用两种不同的语言来构建客户端——服务器软件呢？

结论

这对于Dart来说是一个激动人心的时刻。使用Dart的人喜欢它，而Dart 2中的新特性使其成为你工具库中更有价值的补充。如果你还没有使用过Dart，我希望这篇文章为你提供了有关Dart的新特性的有价值的信息，并且你会试一试Dart和Flutter。

查看英文原文：<https://hackernoon.com/why-flutter-uses-dart-dd635a054ebf>

感谢[覃云](#)对本文的审校。

此内容所在的主题为 [语言 & 开发](#)

[+ 关注 话题](#)

[语言 & 开发](#) [架构 & 设计](#) [Flutter](#) [Dart](#)

相关主题:

相关内容

[为什么原生应用开发者需要关注Flutter](#)

[为什么说Flutter是革命性的？](#)

[高可用低延时的PayPal风控数据平台](#)

[QUIC在微博中的落地思考](#)

[微软人工智能又一里程碑：微软中-英机器翻译水平可“与人类媲美”](#)

相关厂商内容

[详解 gRPC 背后的运行机制与原理](#)

[逻辑思维Go语言微服务改造完整过程](#)

[Netflix的未来IT架构模型：Serverless](#)

[阿里巴巴数据处理引擎Blink核心设计](#)

[谷歌研究院出品：TensorFlow在深度学习中的应用](#)

相关赞助商



告诉我们您的想法

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

☐ 当有人回复此评论时请E-mail通知我

社区评论 [Watch Thread](#)
[Dart的GC适合于嵌入式与服务器?](#) by 孙 奇辉 Posted 2018年3月14日 08:37

Dart的GC适合于嵌入式与服务器? 2018年3月14日 08:37 by "[孙 奇辉](#)"

嵌入式与服务器很少重启, 70*24*365后, Dart的gc会否产生大量碎片呢?

- [喜欢](#)
- [回复](#)
 - [回到顶部](#)

[关闭](#)

by

发布于

- [查看](#)
- [回复](#)
- [回到顶部](#)

[关闭](#)

主题

您的回复

引用原消息

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

☐ 当有人回复此评论时请E-mail通知我

[关闭](#)

主题

您的回复

允许的HTML标签: a,b,br,blockquote,i,li,pre,u,ul,p

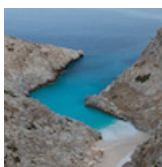
☐ 当有人回复此评论时请E-mail通知我

[关闭](#)

赞助商链接

相关内容

- [为什么说Flutter是革命性的?](#) 2017年10月13日



- [高可用低延时的PayPal风控数据平台](#) 2018年3月17日
- [QUIC在微博中的落地思考](#) 2018年3月16日
- [微软人工智能又一里程碑：微软中-英机器翻译水平可“与人类媲美”](#) 2018年3月16日
- [为孩子和开发团队解密机器学习](#) 2018年3月16日
- [可观测性对测试的影响：QCon伦敦大会上对Amy Phillips的访谈](#) 2018年3月16日
- [如何完善自我](#) 2018年3月16日
- [前端每周清单第 53 期：Go 与 WebAssembly, React Suspense 演练, CSS 技巧](#) 2018年3月16日



- [Netflix最新视频优化实践：用更少的带宽打造完美画质](#) 2018年3月16日



- [如何调试分布式系统：与微服务调试工具“Squash”创始人Idit Levine的对话](#) 2018年3月16日 



- [ICASSP Poster论文：阿里提出深层前馈序列记忆神经网络，语音识别性能提升20%](#) 2018年3月16日

赞助商内容



• [AICon 人工智能专家团带你直击AI工业落地](#)

想了解深度学习在CTR预估、图像处理中的应用？想系统学习知识图谱技术？从模型到实战，60讲深度学习应用实践课，AICon人工智能专家团带你进阶深度学习！

Geekbang InfoQ

ArchSummit
全球架构师峰会

2018·深圳站

深圳·华侨城洲际酒店

会议：7月6-7日

培训：7月8-9日



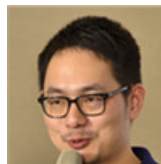
• [All in AI下的海内外架构推演](#)

Apple、Google、Facebook、Microsoft等国内外上百余项架构案例详解，尽在ArchSummit全球架构师峰会

赞助商 **ArchSummit**

相关内容

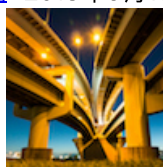
- [从Instagram到Reddit, 浅谈西方工程师文化和管理](#) 2018年3月16日
- [Propel: 由Node.js之父创建的JavaScript科学计算库](#) 2018年3月15日
- [迅雷发布2017年报: 下季度将推共享计算企业级新产品](#) 2018年3月15日



- [化茧成蝶: Go在FreeWheel服务化中的实践](#) 2018年3月15日
- [数据驱动的持续改进思维](#) 2018年3月15日
- [想搞定微服务, 先搞定 RPC 框架?](#) 2018年3月15日
- [推荐你身边的女神加入 TGO 鲲鹏会吧!](#) 2018年3月15日
- [聊聊分布式系统的认知故障和弹力设计](#) 2018年3月15日
- [微软发布Azure Redis Cache服务的跨地域复制功能](#) 2018年3月15日



- [虚拟小组: 利用事件溯源获得成功](#) 2018年3月15日



- [这里有25个React Native免费教程, 请您查收](#) 2018年3月15日



相关内容

- [如何把全世界的Web浏览器连成一个超级计算机?](#) 2018年3月15日



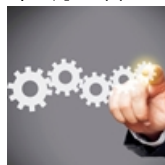
- [微博社交广告系统架构实践](#) 2018年3月15日
- [ArchSummit2018深圳站筹备中, 18大专题征集演讲嘉宾](#) 2018年3月14日
- [Let's Encrypt 宣布支持通配符证书, 所有子域名可轻松开启 HTTPS](#) 2018年3月14日
- [物联网技术周报第 128 期: 深度解读: 深度学习在IoT大数据和流分析中的应用](#) 2018年3月14日
- [Cilium 1.0.0-rc4发布: 使用Linux BPF实现透明安全的容器间网络连接](#) 2018年3月14日
- [Python数据科学平台Anaconda的最新发布中增加了Microsoft VS Code](#) 2018年3月14日
- [苹果开源Swift底层非阻塞I/O框架SwiftNIO](#) 2018年3月14日
- [Reilly软件架构大会上的事件驱动微服务主题](#) 2018年3月14日



- [Kotlin/Native应用程序开发指南](#) 2018年3月14日



- [排查Java的内存问题](#) 2018年3月14日



InfoQ每周精要

订阅InfoQ每周精要, 加入拥有25万多名资深开发者的庞大技术社区。



点击查看
样刊效果

语言 & 开发

[高可用低延时的PayPal风控数据平台](#)

[QUIC在微博中的落地思考](#)

[微软人工智能又一里程碑：微软中-英机器翻译水平可“与人类媲美”](#)

架构 & 设计

[QUIC在微博中的落地思考](#)

[微软人工智能又一里程碑：微软中-英机器翻译水平可“与人类媲美”](#)

[前端每周清单第 53 期：Go 与 WebAssembly, React Suspense 演练, CSS 技巧](#)

文化 & 方法

[如何完善自我](#)

[为孩子和开发团队解密机器学习](#)

[从Instagram到Reddit，浅谈西方工程师文化和管理](#)

数据科学

[为孩子和开发团队解密机器学习](#)

[Propel: 由Node.js之父创建的JavaScript科学计算库](#)

[ArchSummit2018深圳站筹备中，18大专题征集演讲嘉宾](#)

DevOps

[可观测性对测试的影响：QCon伦敦大会上对Amy Phillips的访谈](#)

[如何调试分布式系统：与微服务调试工具“Squash”创始人Idit Levine的对话](#)

[ArchSummit2018深圳站筹备中，18大专题征集演讲嘉宾](#)

- [首页](#)
- [全部话题](#)
- [QCon全球软件开发大会](#)
- [关于我们](#)
- [投稿](#)
- [注销](#)
- [全球QCon](#)
- [伦敦 Mar 6-10, 2017](#)
- [北京 Apr 20-22, 2018](#)
- [圣保罗 Apr 24-26, 2017](#)
- [纽约 Jun 26-30, 2017](#)
- [上海 Oct 18-22, 2018](#)
- [东京, 2017 秋](#)
- [旧金山 Nov 13-17, 2017](#)

InfoQ每周精要

订阅InfoQ每周精要，加入拥有25万多名资深开发者的庞大技术社区。

[点击这里
查看样刊](#)



- [RSS订阅](#)
- [InfoQ官方微博](#)
- [InfoQ官方微信](#)

- [社区新闻和热点](#)

特别专题

- [活动大本营](#)
- [月刊:《架构师》](#)
- [AWS专区](#)
- [百度技术沙龙专区](#)
- [AICon](#)
- [信息无障碍参考文档](#)

提供反馈

错误报告

商务合作

内容合作

市场合作

feedback@cn.infoq.com bugs@cn.infoq.com hezuo@geekbang.org editors@cn.infoq.com hezuo@geekbang.org

InfoQ.com
及所有内
容，版权所
有 ©
2006–
2018
C4Media
Inc.
InfoQ.com
服务器由
[Contegix](#)提
供，我们最
信赖的ISP
伙伴。
极客邦控股
(北京)有
限公司 [隐
私政策](#)



我们发现您在使用ad blocker。

我们理解您使用ad blocker的初衷，但为了保证InfoQ能够继续以免费方式为您服务，我们需要您的支持。InfoQ绝不会在未经您许可的情况下将您的数据提供给第三方。我们仅将其用于向读者发送相关广告内容。请您将InfoQ添加至白名单，感谢您的理解与支持。

This ad is supporting your extension Conveyor: [More info](#) | [Privacy Policy](#) | [Hide on this page](#)