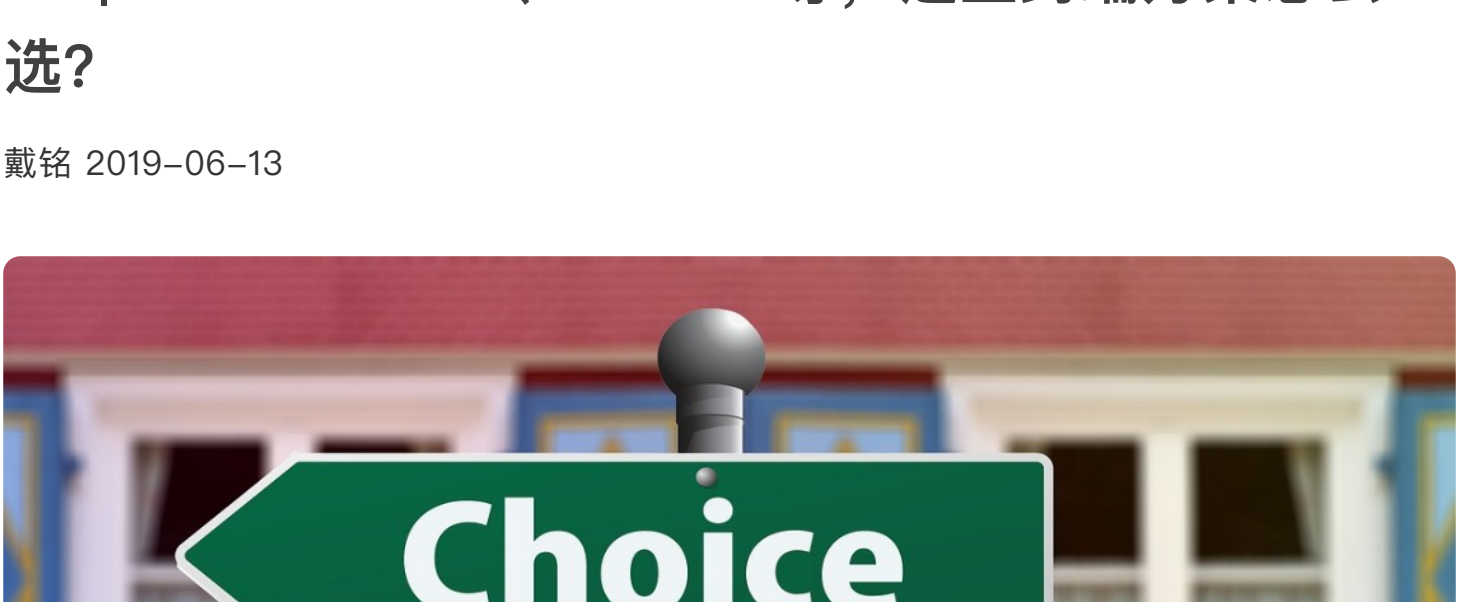
40 | React Native、Flutter 等，这些跨端方案怎么选？

戴铭 2019-06-13



00:00 10:20
讲述：冯永吉 大小：9.47M

你好，我是戴铭。

为了一份代码能够运行在多个平台，从而节省开发和沟通成本，各公司都开始关注和使用跨端方案。目前，主流的跨端方案，主要分为两种：一种是，将 JavaScriptCore 引擎当作虚拟机的方案，代表框架是 React Native；另一种是，使用非 JavaScriptCore 虚拟机的方案，代表框架是 Flutter。

使用跨端方案进行开发，必然会替代原有平台的开发技术，所以我们在选择跨端方案时，不能只依赖于某几项指标，比如编程语言、性能、技术架构等，来判断是否适合自己团队和产品，更多的还要考虑开发效率、社区支持、构建发布、 DevOps、 CI 支持等工程化方面的指标。

所以说，我们在做出选择时，既要着眼于团队现状和所选方案生态，还要考虑技术未来的发展走向。

接下来，我就以 React Native 和 Flutter 为例，和你说明如何选择适合自己的跨端方案。

React Native 框架的优势

跨端方案的初衷是要解决多平台重复开发的问题，也就是说，使用跨端方案的话，多个平台的开发者可以使用相同的开发语言来开发适合不同系统的 App。

React Native 使用 JavaScript 语言来开发，Flutter 使用的是 Dart 语言。这两门编程语言，对 iOS 开发者来说都有一定的再学习成本，而使用何种编程语言，其实决定了团队未来的技术栈。

JavaScript 的历史和流行程度都远超 Dart ，生态也更加完善，开发者也远多于 Dart 程序员。所以，从编程语言的角度来看，虽然 Dart 语言入门简单，但从长远考虑，还是选择 React Native 会更好一些。

同时，从页面框架和自动化工具的角度来看，React Native 也要领先于 Flutter。这，主要得益于 Web 技术这么多年的积累，其工具链非常完善。前端开发者能够很轻松地掌握 React Native，并进行移动端 App 的开发。

当然，方案选择如同擂台赛，第一回合的输赢无法决定最后的结果。

Flutter 框架的优势

除了编程语言、页面框架和自动化工具以外，React Native 的表现就处处不如 Flutter 了。总体来说，相比于 React Native 框架，Flutter 的优势最主要体现在性能、开发效率和体验这两大方面。

Flutter 的优势，首先在于其性能。

我们先从最核心的**虚拟机**说起吧。

React Native 所使用的 JavaScriptCore， 原本用在浏览器中，用于解释执行网页中的 JavaScript 代码。为了兼容 Web 标准留下的历史包袱，无法专门针对移动端进行性能优化。

Flutter 却不一样。它一开始就抛弃了历史包袱，使用全新的 Dart 语言编写，同时支持 AOT 和 JIT 两种编译方式，而没有采用 HTML/CSS/JavaScript 组合方式开发，在执行效率上明显高于 JavaScriptCore 。

除了编程语言的虚拟机，Flutter 的优势还体现在**UI 框架的实现**上。它重写了 UI 框架，从 UI 控件到渲染，全部重新实现了，依赖 Skia 图形库和系统图形绘制相关的接口，保证了不同平台上能有相同的体验。

想要了解 Flutter 的布局和渲染，你可以看看这两个视频“The Mahogany Staircase – Flutter’s Layered Design”和“Flutter’s Rendering Pipeline”。

除了性能上的优势外，Flutter 在开发效率和体验上也有很大的建树。

凭借热重载（Hot Reload）这种极速调试技术，极大地提升了开发效率，因此 Flutter 吸引了大量开发者的眼球。

同时，Flutter 因为重新实现了 UI 框架，可以不依赖 iOS 和 Android 平台的原生控件，所以无需专门去处理平台差异，在开发体验上实现了真正的统一。

此外，Flutter 的学习资源也非常丰富。Flutter 的官方文档，分门别类整理得井井有条。YouTube 上有一个专门的频道，提供了许多讲座、演讲、教程资源。

或许，你还会说 Flutter 包大小是个问题。Flutter 的渲染引擎是自研的，并没有用到系统的渲染，所以 App 包必然会大些。但是，我觉得从长远来看，App Store 对包大小的限制只会越来越小，所以说这个问题一定不会成为卡点。

除了上面两大优势外，我再和你说说 Flutter 对动态化能力的支持。

虽然 Flutter 计划会推出动态化能力，但我觉得动态化本身就是一个伪命题。软件架构如果足够健壮和灵活，发现问题、解决问题和验证问题的速度一定会非常快，再次发布上线也能够快速推进。而如果软件架构本就一团糟，解决问题的速度是怎么也快不起来的，即使具有了动态化能力，从解决问题到灰度发布再到全量上线的过程也一定会很曲折。

所以，我认为如果你想通过动态化技术来解决发布周期不够快的问题的话，那你首先应该解决的是架构本身的问题。长远考虑，架构上的治理和优化带来的收益，一定会高于使用具有动态化能力的框架。

当然，如果你选择使用动态化能力的框架，是抱着绕过 App Store 审核的目的，那就不在本文的讨论范围之内了。

如何选择适合自己的跨端方案？

看到这，你一定在想，跨端方案不是只有 Rect Native 和 Flutter，还有小程序、快应用、Weex 等框架。没错，跨端方案确实有非常多。

但，**我今天与你分享的 React Native 代表了以 JavaScriptCore 引擎为虚拟机的所有方案，对于这一类方案的选择来说，道理都大同小异**。只要你打算转向前端开发，选择它们中的哪一个方案都差不多，而且方案间的切换也很容易。

着眼未来，决定跨端方案最终赢家的关键因素，不是编程语言，也不是开发生态，更不是开发者，而是用户。

如果谷歌的新系统 Fuchsia 能够如谷歌所计划的五年之内应用到移动端的话，那么五年后即使使用 Fuchsia 的用户只有 10%，你的 App 也要去支持 Fuchsia。Fuchsia 系统的最上层就是 Flutter，这时使用 Flutter 来开发 App 就成了首选。而 Flutter 本身就是一种跨端方案，一旦使用 Flutter 开发成为团队的必选项，那么其他技术栈就没有存在的价值了。

其实，我本人还是很看好 Fuchsia 系统的。它的内核是 Zircon，Fuchsia 是整个系统的统称，在 Fuchsia 技术的选择上，谷歌选择了微内核、优于 OpenGL 高内核低开销的图像接口 Vulkan、3D 桌面渲染 Scenic、Flutter 开发框架。谷歌的打算是，三年内在一些非主流的设备上对 Fuchsia 内核进行完善，待成熟后推向移动端。

Fuchsia 架构分为四层，包括微内核的第一层 Zircon，提供系统服务的第二层 Garnet，用户体验基础设施的第三层 Peridot，Flutter 所在基础应用的第四层 Topaz。结合 Android 系统的经验，在设计架构之初，谷歌就考虑了厂商对深度定制的诉求，使得每层都可以进行替换，模块化做得比 Android 系统更加彻底。

Fuchsia 架构，如下图所示：

你可以通过这个视频，查看Fuchsia 最近的动向。如果你有 Pixel 3 XL 手机，可以动手尝试下。你可以点击这个链接，来查看支持 Pixel 3 XL 的 Fuchsia 项目。Fuchsia 官方 Git 仓库的地址是https://fuchsia.googlesource.com，你可以点击查看其源码。

当然，不管操作系统多么牛，最后还要由用户来选。

所以，跨端技术方案的赢家是谁，最终还是要看使用移动设备的用户选择了谁，就好像游戏机市场中的 Nintendo Switch 和 PlayStation Vita。PlayStation Vita 在硬件、性能、系统各方面都领先 Nintendo Switch，但最终游戏开发者还是选择在 Nintendo Switch 上开发，而这其实都取决于购买游戏机的玩家。当 Nintendo Switch 成为了流行和热点以后，所有的游戏开发者都会跟着它走。

虽然我们不能决定未来，但我们可以去预测，然后选择一款大概率会赢的跨端框架，以此来奠定自己的竞争力。

总结

在今天这篇文章中，我将跨平台方案分成了两种：一种是，将 JavaScriptCore 引擎当作虚拟机的方案，代表框架是 React Native；另一种是，使用非 JavaScriptCore 虚拟机的方案，代表框架是 Flutter。

然后，在此基础上，我从编程语言、性能、开发效率和体验等方面和你分析了这两类方案。但是，选择一款适合自己团队的跨平台开发方案，仅仅考虑这几个方面还不够，我们还要着眼于未来。

在我看来，从长远考虑的话，你可以选择 Flutter 作为跨平台开发方案。但是，最终 Flutter 是否能成功，还要看谷歌新系统 Fuchsia 的成败。

课后作业

如果最终 Fuchsia 失败了，而 iOS 继续突飞猛进，SwiftUI 也支持跨端了，那你也就不用换技术栈了，继续使用 Swift 开发就好了。你对此是什么看法呢?

感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎把它分享给更多的朋友一起阅读。

精选留言(16)

少林李达康

个人认为跨平台开发始终是个真需求，伪实现。真需求说的是市场确实需要一个完美的兼容多平台的开发方式来解决人力成本和开发效率。伪实现说的是目前无论哪种方案，真正落地开发的话还是必须要掌握安卓和iOS的开发技能。之前我做过一个RN的项目就是，许多api都是iOS这边兼容，安卓不兼容，这就需要了解安卓开发再去做兼容处理，反之亦然。这样其实更降低了开发效率，而且也无形中给开发人员增加了技能要求，必须都熟悉安卓和iOS开发。而真正熟悉两套开发技能的人在市场上的薪资不见得就比一个安卓 + iOS的薪资低。
所以我认为在谷歌和苹果没有给出官方跨平台解决方案之前，一切跨平台都是小打小闹，只能小试牛刀，不能真正解决问题。
个人愚见，不喜勿喷。

2019-06-13 34

鑫琦

之ggtalk的有一期“向架构师进发”，Casa Taloyum也说过跨平台开发的问题，当时给出的答案和@少林李达康 说的一样“一旦需要做兼容处理时，还是会要求熟悉Android和iOS开发技术”， 这样成本还是会不小，所以从长远来看确实不能真正解决问题。但是我的觉得从快发速度的角度来说，跨平台技术既能达到Web开发的速度，又能在一定程度上保证性能，相对HTML混合开发还是有进步的。

另外说一下Flutter我个人的影响 最近离职在找工作，刚想强化一下iOS基础知识，在复习时间突然发现Flutter越来越火了， 当明连iOS都没学的精通，Texture，SwiftUI还没怎么用过，就要开始转移阵地学其他技术了。 但是要现在不学Flutter，又要掉队了。 对我这种技术普通开发者， 还是蛮纠结的
。

另外，“向架构师进发”这一期，我记得Casa Taloyum还引用了戴铭老师的话， 不知道的可以去听一下

2019-06-13 7

2thousand19

期待swift UI 跨安卓平台的那一天

2019-06-13 6

胖娃瓜

NS性能比PSV强的多，PSV应该和3DS对比...

2019-06-14 4

Kratos

老是能讲下动态热更新方案吗

作者回复: 后面就会讲

2019-06-13 3

轩

从GitHub上的swift语言库来看，swift已经从底层上支持Android了，Linux系统（服务器方向）也支持的比较多，windows 也有支持，再加上今年推出的swiftUI来看，底层和应用层分步演进，再加上苹果公司这几年对metal渲染层的深耕，支持跨平台感觉是有望的！我是打算扎根swift了！

2019-07-09 2

Geek_f0b40f

我们就正在是用Flutter进行开发，也遇到过各种问题，但是正如戴铭所说的那样，一切都要看用户和开发者怎么去选，不过从目前来看，Flutter的增量远远高于RN，建议深耕Swift的情况下，着手开发Flutter吧！并且Flutter已经支持嵌入式/桌面端/web端 等平台了，野心蛮大的..

2019-06-13 2

张松超

SwiftUI渲染机智和UIKit有差别吗？如果没有差别，就不可能有跨安卓的那一天吧?

作者回复: 渲染是底层的。SwitUI 是接口层

2019-06-22 1

小美

不管上层建筑如何变换，目前可以做的就是打好基本功，知其所以然，而不是想当然，与各位共勉。

2019-06-13 1