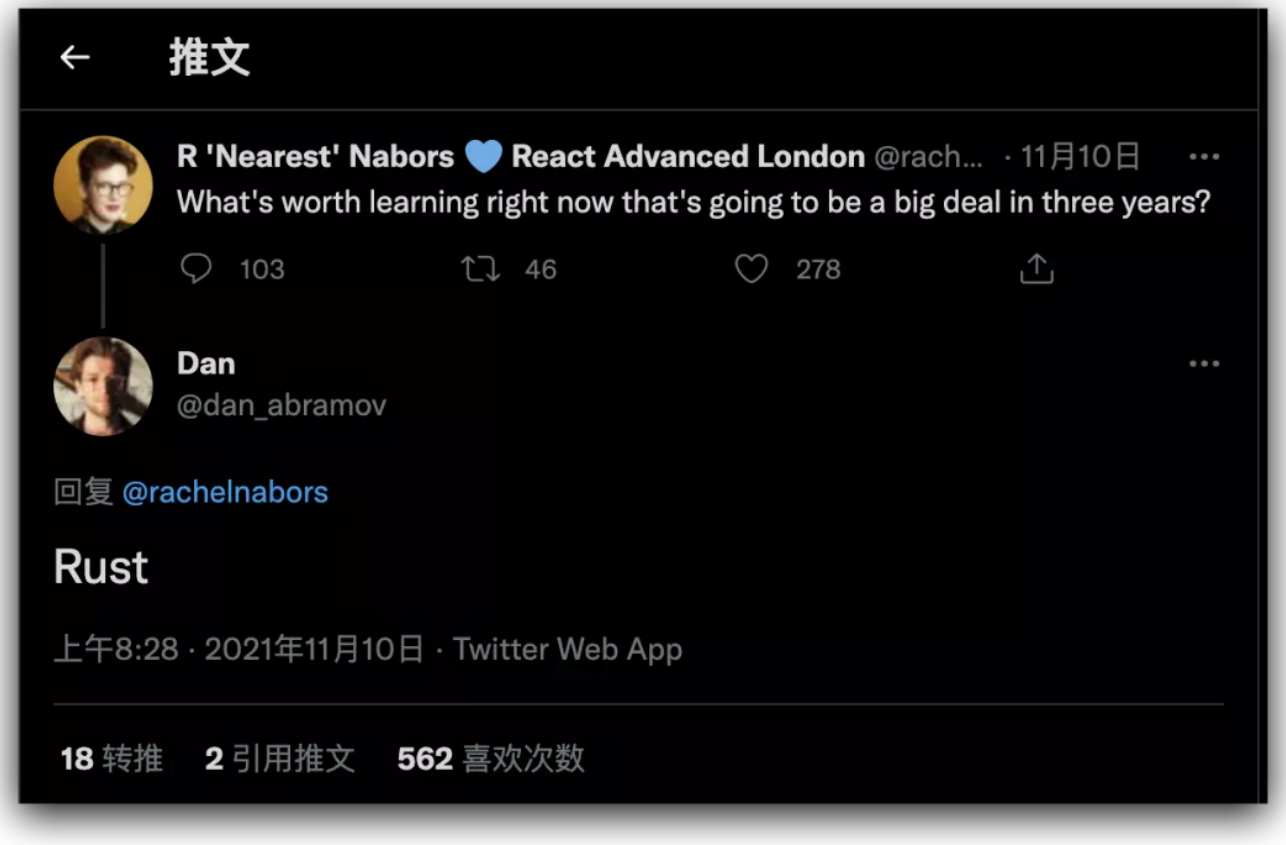
Rust 是 JavaScript 基础设施的未来

原创 sinoon ByteDance Web Infra 2021-11-16 09:51

收录于话题
#rust 1 #Modern.js 3 #译文 8

> 本文翻译自：Rust Is The Future of JavaScript Infrastructure – Lee Robinson[1]，已获得作者授权，原文略做修改以符合中文语境。



(图片镇楼，与原文无关)

Rust[2] 是一种快速、可靠、内存高效的编程语言。它已经连续六年被评为最受 欢[3] 迎[4] 的 编[5] 程[6] 语[7] 言[8]。它由 Mozilla 创建，现在被 Facebook[9]、苹果[10]、亚马逊[11]、微软[12] 和 谷歌[13] 用于系统基础设施、加密、虚拟化和更多底层（low-level）的编程中。

为什么Rust现在被用来取代 JavaScript 成为 Web 生态系统的一部分，如压缩（Terser）、编译（Babel）、格式化（Prettier）、打包（webpack）、linting（ESLint），以及其他场景？

什么是Rust?

Rust 帮助开发人员编写运行快并高效利用内存的软件。它是 C++ 或 C 等语言的现代替代品，聚焦在代码安全和简洁的语法。

Rust 与 JavaScript 有很大不同。JavaScript 采用的是 **垃圾回收**[14] 机制，会试图清除不再使用的变量和对象。JavaScript 将开发者从对手动内存管理的思考中抽离出来。

有了 Rust，开发者对内存分配有了更多的控制，而不像 C++ 或 Go 那样令人痛苦。

> Rust 使用了一种相对独特的内存管理方法，包含了内存“所有权”的概念。基本上，Rust 会记录谁可以读取和写入内存。它知道程序什么时候在使用内存，一旦不再需要就立即释放内存。它在编译时执行内存规则，使得运行时的内存错误几乎不可能出现。你不需要手动跟踪内存。编译器会照顾到这一点。
>
> —— **Discord**[15]

Adoption

在上面提到的公司之外，Rust 还被用于流行的开源库，比如：

- **Firecracker**[16] (AWS)
- **Bottlerocket**[17] (AWS)
- **Quiche**[18] (Cloudflare)
- **Neqo**[19] (Mozilla)

> Rust 一直是我们团队的力量源泉，押注Rust是我们做出的最好决定之一。除了性能之外，它的人机工程学（*ergonomics*）和对正确性的关注也帮助我们控制了同步的复杂性。我们可以在类型系统中对系统的复杂不变性进行编码，让编译器为我们检查。
>
> —— **Dropbox**[20]

从 JavaScript 到 Rust

JavaScript 是使用最广泛的编程语言，存在于每个有浏览器的设备上。在过去的十年中，围绕着 JavaScript 已经建立了一个庞大的生态系统：

- **Webpack**： 开发者希望将多个 JavaScript 文件 bundle 成一个文件。
- **Babel**： 开发者希望在支持旧版浏览器的同时编写现代 JavaScript 。
- **Terser**： 开发者希望生成尽可能小的文件。
- **Prettier** :开发者希望有一个有自带规则（opinionated）的代码格式化器，刚好可用就行。
- **ESLint**： 开发者希望在部署前发现他们的代码存在的问题。

数以百万计的代码被编写出来，甚至更多的 bug 被修复，从而为今天的 Web 应用提供了基础。所有这些工具都是用 JavaScript 或 TypeScript 编写的。目前已经做的很不错了，但我们对 JS 的优化已经达到顶峰。因此，这激发了一类新的工具们的诞生，旨在极大地提高 Web 构建的性能。

SWC

SWC[21] 创建于 2017 年，一个下一代快速开发的工具，是一个基于 Rust 的可扩展平台。Next.js、 Parcel 和 Deno 等工具，以及 Vercel、字节跳动、腾讯、Shopify 等公司都在使用它。

SWC 可用于编译（compilation）、压缩（minification）、打包（bundling）等 —— 并且被设计成可扩展的。你可以调用它来执行代码转换（无论是内置的还是自定义的）。这些转换由更高维度的工具来运行，比如 Next.js 。

Deno

Deno[22] 创建于 2018 年，是一个简单、现代、安全的 JavaScript 和 TypeScript 的运行时，Deno 使用 **V8**[23] 的同时，并以 Rust 构建。它试图取代 Node.js ，由 Node.js 的原作者创建。虽然它是在 2018 年创建的，但直到 2020 年 **5月才达到**[24] **v1.0**[25] 。

Deno 的 linter、code formatter 和文档生成器是使用 **SWC 构建的**[26] 。

esbuild

esbuild[27] 创建于 2020 年 1 月，一个代码打包和压缩的工具，它是用 Go 写的，比当今的工具快 10 到 100 倍。

> 我试图创建一个构建工具：
>
> A）能够在一个合理的场景（bundler JavaScript，TypeScript，也许还有CSS）中工作得很好。
>
> B）能够重新定义社区对构建工具速度的期望。对于JS快速构建工具来说，到底什么才是快？在我看来，我们目前的工具都太慢了。
>
> —— Evan，Creator of esbuild（ **Source**[28] ）

在 esbuild 发布之前，用系统编程语言（如 Go 和 Rust ）构建 JavaScript 工具是相当小众的。因此在我看来，esbuild 激发人们对于让开发工具更快的广泛兴趣。Evan 选择了使用 Go：

> Rust版本也许可以通过足够的努力使其以同等速度工作。但在高层次上，Go 的工作方式更令人愉快。这是一个 side project ，所以它对我来说必须要有乐趣。
>
> —— Evan，Creator of esbuild（ **Source**[29] ）

有人认为 Rust 可以表现得更好，但两者都可以实现 Evan 最初的目标，即影响社区：

> 即使只是进行了基本的优化，Rust也能够胜过超手工调整的 Go 版本。这是一个巨大的证明，与我们不得不对 Go 的深入研究相比，用Rust编写高效程序是多么容易。
>
> **—— Discord**[30]

Rome

Rome[31] 创建于 2020 年 8 月，包含 linter、compiler、bundler、test runner 以及其它东西，适用于 JavaScript、TypeScript、HTML、JSON、Markdown 和 CSS 。他们的目标是取代和统一整个前端开发工具链。它是由 **Sebastian**[32] 创建的，他是 Babel 的创建者。

那么，为什么要重写一切呢？

> 对 Babel 进行必要的修改，使其成为其他工具的可靠基础，这绝对需要对所有东西进行修改。这个架构与我在 2014 年学习解析器、 AST 和编译器时做出的最初设计选择是有紧密联系的。

> —— Sebastian（Source[33]）

Rome 目前是用 TypeScript 编写的，在 Node.js 上运行。但他们现在正致力于用 Rust[34] 重写[35]。

NAPI

Rust 与 Node.js 的整合比其他低级语言更好。

napi-rs[36] 允许你用 Rust 构建预编译的 Node.js add-ons。它为交叉编译（cross-compilation）和向 NPM 发布本地二进制文件提供了一个开箱即用的解决方案，不需要 `node-gyp` 或使用 `postinstall`。

相应的，你也能轻松构建一个被 Node.js 侧直接调用的 Rust 模块，而不需要像 esbuild 那样创建一个子进程。

Rust + WebAssembly

WebAssembly[37]（WASM）是一种可移植的低级语言，Rust 可以编译成它。它在浏览器中运行，可与 JavaScript 互操作，并被所有主要的现代浏览器所支持。

> WASM 肯定比 JS 快很多，但还没有达到原生速度。在我们的测试中，Parcel 编译成 WASM 后的运行速度比使用本地二进制文件慢 10 - 20 倍。
>
> —— Devon Govett[38]

虽然 WASM 还不是完美的解决方案，但它可以帮助开发者创建极快的 Web 体验。Rust 团队致力于[39] 实现高质量和先进（cutting edge）的 WASM 实现。对于开发者来说，这意味着你可以拥有 Rust 的性能优势（相对于 Go），同时还可以为 Web 服务（使用 WASM）。

这个领域的一些早期库和框架：

- Yew[40]
- Percy[41]
- Seed[42]

- Sycamore[43]
- Stork[44]

这些基于 Rust 的、可编译为 WASM 的 Web 框架并不是要取代 JavaScript，而是要与之一起工作。虽然我们还没有达到这个目的，但看到 Rust 在两方面都追随 Web 的发展是很有趣的：使现有的 JavaScript 工具更快，并在未来提出 **编译到 WASM**[45] 的想法。

这就是 Rust 之后的路。

为什么不选择 Rust？

Rust 有一个陡峭的学习曲线。它的抽象程度比大多数 Web 开发者所习惯的要低。

> Rust 会迫使你思考你代码中对系统编程方面具有很大影响的部分。它让你思考内存是如何共享或复制的。它让你思考真实但不可能发生的边界问题，并确保它们得到处理。它可以帮助你写出在所有可能的方面都非常高效的代码。
>
> —— Tom MacWright (**Source**[46])

此外，Rust 在网络社区的使用仍然是小众的。它还没有达到技术选型临界点（critical adoption）。即使学习 Rust 的 JavaScript 工具会有一定的门槛，但有趣的是，开发者宁愿拥有一个 **更快、更难参与贡献**[47] 的工具。**唯快不破**[48] **（Fast software wins）** [49]。

目前，你很难为你喜欢的服务（如登录鉴权、数据库、支付等工作）找到一个 Rust 库或框架。我确实认为，一旦 Rust 和 WASM 达到技术选型临界点（critical adoption），这个问题会自行解决。但现在还不行。**我们需要现有的 JavaScript 工具来帮助我们缩小差距，并逐步采用性能改进。**

JavaScript 工具化的未来

我相信 Rust 是 JavaScript 工具化的未来。**Next.js 12**[50] 已经开始了我们的转型，用 SWC 和 Rust 完全取代 Babel（transpilation）和 Terser（压缩）。为什么会有这样的选择?

- **可扩展性（Extensibility）**。SWC 可以作为 Next.js 内部的一个 Crate 使用，而无需 fork 其它库或面临设计上的限制（workaround design constraints）。

- **性能（Performance）**。通过改用 SWC，我们在 Next.js 中实现了约 3 倍的刷新速度（Fast Refresh）和约 5 倍的构建速度，还有更多的优化正在进行。
- **WebAssembly**。Rust 对 WASM 的支持，使得 Next.js 可以支持所有可能的平台，让 Next.js 遍布所有的地方。
- **社区（Community）**。Rust 社区和生态系统是惊人的，而且还在不断增长。

在逐步使用 SWC 的绝不仅仅是 Next.js：

- Deno[51] linter、代码格式化器和文档生成器都是用 SWC 构建的[52]。
- Rome[53] 正在 用 Rust 重写[54]，并计划使用 SWC。
- dprint[55]，建立在 SWC 之上，一个快 30 倍的用于替代 Prettier 的代码格式化工具。
- Parcel[56] 使用 SWC 将整体构建性能 提高了 10 倍[57]。

> Parcel 像一个库一样使用 SWC。之前我们使用 Babel 的解析器和用 JS 编写的自定义转换。现在，我们使用 SWC 的解析器和 Rust 中的 自定义转换[58]。这包括一个全局 hoisting 的实现、依赖性收集等。它的范围类似于 Deno 在 SWC 之上的构建方式。
>
> —— Devon Govett[59]

现在是 Rust 的早期阶段 —— 有几个重要的部分还在摸索之中：

- **插件（Plungins）**。对于许多 JavaScript 开发者来说，用 Rust 编写插件并不是那么容易的。同时，在 JavaScript 中暴露一个插件系统可能会否定性能的提升。目前还没有一个明确的解决方案。
- **打包（Bundling）**。一个有趣的发展领域是 **swcpack**，它是 SWC 对 Webpack 的替代。它仍在开发中，但可能是非常有前途的。
- **WebAssembly**。如上所述，编写 Rust 并编译成 WASM 的前景很诱人，但仍有工作要做。

不管怎么说，我相信 Rust 会在未来 1 - 2 年以及未来继续对 JavaScript 生态系统产生重大影响。想象一下这样一个世界：Next.js 中使用的所有构建工具都是用 Rust 编写的，给你带来最佳性能。然后，Next.js 可以作为一个 静态二进制文件[60]（static binary）分发，你可以从 NPM 下载。

这就是我想生活（和建设）的世界。

感谢Devon Govett[61] 审阅本文的早期草稿。

全文完

活动推荐：

一个是前 Facebook 程序员，一个是在中国远程的现Facebook程序员，两位都研究过开发编程语言，两位都是在开发领域有所建树的程序员大佬，赵海平 与 张宏波与你聊聊以往经历，聊聊职业发展，聊聊编程语言～ 11月16号 （周二）， 17:00 - 19:00，掘金直播间，海平和宏波与你不见不散～

直播地址👉：https://live.juejin.cn/4354/rescript-lang[62]

👇 扫码回复 “编程语言”，入群交流

参考资料

[1] Rust Is The Future of JavaScript Infrastructure – Lee Robinson: *https://leerob.io/blog/rust*

[2] Rust: *https://www.rust-lang.org/*

[3] 欢: *https://insights.stackoverflow.com/survey/2016#technology-most-loved-dreaded-and-wanted*

[4] 迎: *https://insights.stackoverflow.com/survey/2017#most-loved-dreaded-and-wanted*

[5] 编: *https://insights.stackoverflow.com/survey/2018#technology-_-most-loved-dreaded-and-wanted-languages*

[6] 程: *https://insights.stackoverflow.com/survey/2019#technology-_-most-loved-dreaded-and-wanted-languages*

[7] 语: *https://insights.stackoverflow.com/survey/2019#technology-_-most-loved-dreaded-and-wanted-languages*

[8] 言: *https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted*

[9] Facebook: *https://engineering.fb.com/2021/04/29/developer-tools/rust/*

[10] 苹果: *https://twitter.com/oskargroth/status/1301502690409709568*

[11] 亚马逊: *https://aws.amazon.com/blogs/opensource/why-aws-loves-rust-and-how-wed-like-to-help/*

[12] 微软: *https://twitter.com/ryan_levick/status/1171830191804551168*

[13] 谷歌: *https://security.googleblog.com/2021/04/rust-in-android-platform.html*

[14] 垃圾回收: *https://en.wikipedia.org/wiki/Garbage_collection_(computer_science)*

[15] —— Discord: *https://blog.discord.com/why-discord-is-switching-from-go-to-rust-a190bbca2b1f*

[16] Firecracker: *https://github.com/firecracker-microvm/firecracker*

[17] Bottlerocket: *https://github.com/bottlerocket-os/bottlerocket*

[18] Quiche: *https://github.com/cloudflare/quiche*

[19] Neqo: *https://github.com/mozilla/neqo*

[20] —— Dropbox: *https://dropbox.tech/infrastructure/rewriting-the-heart-of-our-sync-engine*

[21] SWC: *http://swc.rs/*

[22] Deno: *https://deno.land/*

[23] V8: *https://v8.dev/*

[24] 5月才达到: *https://deno.com/blog/v1*

[25] v1.0: *https://deno.com/blog/v1*

[26] SWC 构建的: *https://twitter.com/devongovett/status/1369033422002389000*

[27] esbuild: *https://esbuild.github.io/*

[28] Source: *https://news.ycombinator.com/item?id=22336334*

[29] Source: *https://news.ycombinator.com/item?id=22336284*

[30] —— Discord: *https://blog.discord.com/why-discord-is-switching-from-go-to-rust-a190bbca2b1f*

[31] Rome: *https://rome.tools/blog/2020/08/08/introducing-rome*

[32] Sebastian: *https://twitter.com/sebmck*

[33] Source: *https://rome.tools/blog/2020/08/08/introducing-rome*

[34] 用 Rust: *https://twitter.com/rometools/status/1422616144763097091*

[35] 重写: *https://twitter.com/rometools/status/1422616144763097091*

[36] napi-rs: *https://napi.rs/*

[37] WebAssembly: *https://webassembly.org/docs/use-cases/*

[38] Devon Govett: *https://twitter.com/devongovett*

[39] 致力于: *https://www.rust-lang.org/what/wasm*

[40] Yew: *https://yew.rs/*

[41] Percy: *https://github.com/chinedufn/percy*

[42] Seed: *https://github.com/seed-rs/seed*

[43] Sycamore: *https://github.com/sycamore-rs/sycamore*

[44] Stork: *https://stork-search.net/*

[45] 编译到 WASM: *https://rustwasm.github.io/docs/book/introduction.html*

[46] Source: *https://macwright.com/2021/01/15/rust.html*

[47] 更快、更难参与贡献: *https://twitter.com/devongovett/status/1261379312898306048*

[48] 唯快不破: *https://craigmod.com/essays/fast_software/*

[49] Fast software wins）: *https://craigmod.com/essays/fast_software/*

[50] Next.js 12: *http://nextjs.org/12*

[51] Deno: *https://deno.land/*

[52] SWC 构建的: *https://twitter.com/devongovett/status/1369033422002389000*

[53] Rome: *https://rome.tools/*

[54] 用 Rust 重写: *https://twitter.com/rometools/status/1422616144763097091*

[55] dprint: *https://github.com/devongovett/dprint-node*

[56] Parcel: *https://parceljs.org/*

[57] 提高了 10 倍: *https://v2.parceljs.org/blog/beta3/*

[58] 自定义转换: *https://github.com/parcel-bundler/parcel/tree/v2/packages/transformers/js/core/src*

[59] Devon Govett: *https://twitter.com/devongovett*

[60] 静态二进制文件: *https://en.wikipedia.org/wiki/Static_build*

[61] Devon Govett: *https://twitter.com/devongovett*

[62] https://live.juejin.cn/4354/rescript-lang: *https://live.juejin.cn/4354/rescript-lang*

阅读原文　文章已于2021/11/16修改