链滴

Deno 将永远支持 TypeScript

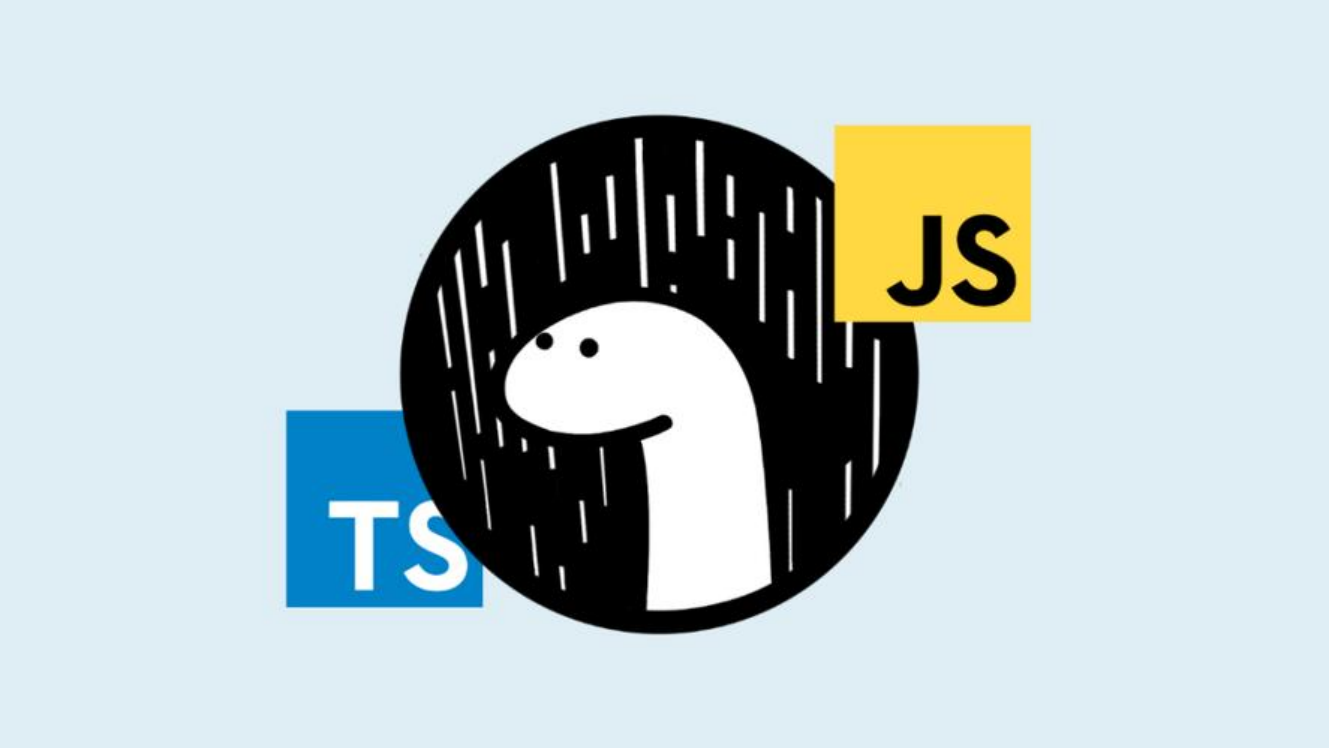
作者：Vanessa

原文链接：https://ld246.com/article/1592881957722

来源网站：链滴

许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

2020-06-23



背景

昨天看到 5 reasons why Deno will stop using TypeScript 这个标题时被吓了一跳。突然想到会不是 1.0 发布中提到的 TypeScript 编译瓶颈，但他的解决方案并不是放弃 TypeScript 呀。很害怕 Den 自己啪啪啪打脸，于是又特意跑到官网上看看 "Supports TypeScript out of the box" 这一特性还不在。哦，放心了，于是慢慢的一探究竟。

最终发现官方的标题明明是《设计文档：使用 JavaScript 代替 TypeScript 做为 Deno 的内部代码》。

声明

2020-06-19：我看到此设计文档的讨论已经远远超出了他的范围和初心。大多数人并没有通过上下来真正理解这份技术文档——此文档仅适用于 Deno 内部非常特殊，非常技术性的情况。这根本不能为对 TypeScript 的评价。这也不是关于 Deno 任何公开可见的讨论。当然，Deno 将永远支持 TypeScript。用 TypeScript 编写的网站或服务器与 Deno 相比是截然不同的程序类型——Deno 只有一小分使用 TypeScript 进行编写。该文档的目标受众为从事此特定内部系统的 5 至 10 人。请不要异想其他子虚乌有的结论。

问题

- 修改文件 cli/js 时会增加编译时长，每当修改他时，就会变得非常缓慢和痛苦。
- 我们在 cli/js 中使用的 TypeScript 组织/结构造成了运行时的性能问题。举例来说，我们最近意到我们无法获得 TS 生成名为 Header 的类，因为他遮盖了我们 d.ts 文件中的声明。因此，我们需要类名修改为 HeaderImpl，并将其赋值给 window.Header。但这会引发 Header.name 错误值的问。因此，我们只能添加一些不必要的运行时代码：Object.defineProperty(HeaderImpl, "name", { value: "Header" }); 来修复 Header.name 问题。但谁知道这是否会脱离 V8 中的某些优化路径。最佳方法是生成 class Header {...}，但这是设计中存在的根本缺陷。

原文链接：Deno 将永远支持 TypeScript

● 假设 TypeScript 可以帮助我们组织代码，但总有人可以说他具有负面的影响。例如，我们有两个立的 Body 类 https://github.com/denoland/deno/issues/4748。这由于生成运行时代码的复杂性难让人看到。

理想情况下，我们的系统中有两个 Body 类显然是错误的。

● 我们的内部代码和运行时的 TS 声明必须使用手动来保持同步。TSC 不能帮助我们生成 d.ts 文件—以前的尝试会产生过多的开销和复杂性。

● 我们有两个独立的 TS 编译器主机：Deno 内部代码使用 //deno_typescript/compiler_main.js 外部用户代码使用 //cli/js/compiler/。这两个主机有相似的目标——他们试图修改 TSC 来运行 Deno 风格的导入（如，文件扩展名）。//deno_typescript 在构建时使用而 //cli/js/compiler 在运行时使
。

Deno 应该专注于为用户提供高性能和安全的运行时环境。Deno 内部代码的组织永远都不会以其交的产品做为代价。目前，我们正在尝试“自我托管”，这样就可以让我们组织内部的 TS 代码看起来和 Deno 用户代码一样。 这是一件好事，但这并不是必需的。我们尝试的“自我托管”越来越明显告诉们，他正在阻碍用户代码的性能和运行时的可维护性。

解决方案

一个根本的解决方案是删除所有内部代码生成时的 TS 类型检查和绑定。我们将所有运行时代码移动单个的大文件 cli/rt.js 中。该文件在构建时将直接放入 V8 中以创建快照。对应的 cli/rt.d.ts 将包含型定义和文档。

在这种方案下就可以编写类似于 class Header { } 的代码，并能清晰地了解运行时提供和执行的内容。

具有 1 万行的 javascript 文件听上去很愚蠢。但我认为这与当前系统相比，他能使新的贡献者快速了 Deno 是如何 bootstrap 的。

使用此方案，构建时的复杂性和性能将会更好。当然，我们依旧需要做一个快照，但这样我们就不需构建一个特殊的 TSIsolate 来进行 tsc 的首次执行。

使用此方案，运行时的复杂性和性能将会更好。我们可以清晰的知道我们是否存在一个类的多个定义我们可以清晰的看到所有正在执行的代码。我们还能在调试过程中分析和逐步执行代码，而不必担心外的源码映射所带来的复杂性。

最需要声明的是：Deno 用户仍然可以使用 typescript 进行编码和类型检查。我们将会进行测试，以保代码可以像 cli/rt.d.ts 一样正常运行。

编译器工作人员将只需要自己单独的 cli/compiler.js 文件，而不需要对应的 d.ts。这种方案的一个缺就是很难在编译器工作和他自身运行时之间共享代码。

返回总目录

每天 30 秒系列之前端资讯

摘自

Design Doc: Use JavaScript instead of TypeScript for internal Deno Code