



Vite: 对下一代前端工具的思考

尤雨溪

2021.07.04 @ GMTC

精彩继续！ 更多一线大厂前沿技术案例

📍 北京站

PCon

全球产品创新大会

时间：2021年8月20-21日

地点：北京·国际会议中心

扫码查看大会
详情>>



📍 深圳站

ArchSummit

全球架构师峰会

时间：2021年9月3-4日

地点：深圳·大中华喜来登酒店

扫码查看大会
详情>>



📍 北京站

AiCon

全球人工智能与机器学习技术大会

时间：2021年9月17-18日

地点：北京·国际会议中心

扫码查看大会
详情>>



Vite 到底是什么？

两个组成部分：

1. No-Bundle 开发服务器

- 源文件无需打包，直接以 原生 ES modules 的形式加载

2. 生产构建

- 基于 Rollup 预先配置好的，针对生产环境高度优化的打包命令。

为什么用 Vite

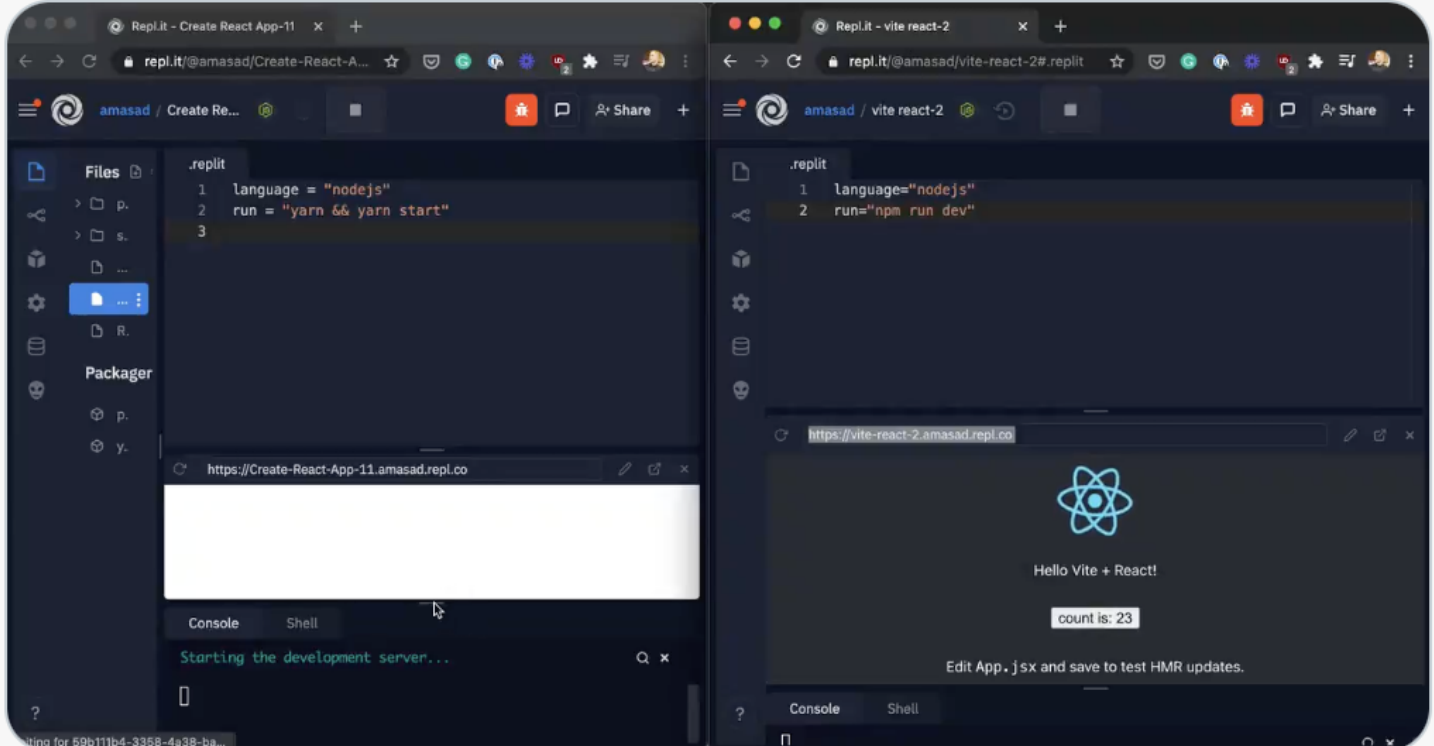
- 快!
- 合理的默认配置
- 灵活可扩展

有多快？

Replit 是一个提供在线编程的服务。在它们的容器中跑 Vite 和 React 官方的 `create-react-app` (CRA) 来启动一个 React 应用，Vite 启动完毕的时候 CRA 甚至还没有下载完 CRA 的依赖。

 **Amjad Masad** · @amasad · Jan 29
Create React App vs Vite React on @replit.

Vite ran before the container could even boot CRA files.

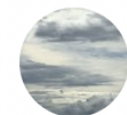


9 116 548

从 Rollup 迁移到 Vite

- 启动时间从 2分15秒 减少到 1.7 秒
- 更新时间从 23 秒减少到 <1 秒

↻ Vite ⚡ Retweeted



Bjorn Lu
@bluwyoo

...

Just migrated a production [@sveltejs](#) app from [@RollupJS](#) to [@vite_js](#). The speed is amazing!

Comparison 👉

Cold start: 2m15s -> 1722ms

Warm start: 2m15s -> 283ms

Reload: 23s -> Instant!

Build: 3m -> 2m22s

Best productivity upgrade for a long time ⚡

1:57 PM · May 13, 2021 · Twitter Web App

5 Retweets 1 Quote Tweet 64 Likes

从 Webpack 迁移到 Vite

- 15 万行代码的巨型 React 项目
- 启动 + 页面加载时间从 2分36秒 减少到 16 秒! (9.75x faster)
- 热更新从 13 秒减少到 1 秒 (13x faster)

Tool	yarn start	app loads in	React component hot reload **	web-worker change "hot" reload **
Webpack*	150s	6s	13s	17s
Vite*	6s	10s	1s	13s

[原文链接](#)

Vite 的设计基于

两个新趋势

1. 现代 JavaScript 支持广泛铺开

- 原生 ES modules 已有 92.83% 的全球浏览器支持率
 - 包括 Vite 依赖的原生 ES modules
- 微软强势推 Edge, 引导 IE11 退役
 - 主流框架开始抛弃 IE11 (Wordpress, Vue, Angular)

2. 新一代的用原生编译语言写的 JS 编译器

- 基于 Go 的 esbuild
- 基于 Rust 的 SWC

两者都比基于 JS 的编译工具快一个数量级 (20~100x, 视任务类型而定)

原生 ESM 开发服务器的优势

- 不需要打包源码
- 自然按需处理
- 可直接利用浏览器缓存
- 可以实现更简单高效的热更新 (HMR)

原生 ESM 服务器的技术挑战

HTTP 请求开销

- 加载大量的小模块 = 大量的并发 HTTP 请求 = 慢
- 举个例子: ``lodash-es`` 包含了超过 700 个内部模块!

如何减少 HTTP 请求开销?

1. 用 esbuild 进行依赖预打包

- 真的快，快出声
- 保证一个依赖对应最多一个 HTTP 请求
- 同时处理 CommonJS 转 ESM 的兼容

2. 利用 HTTP header 缓存依赖

- 对预打包过的依赖的请求会带有 `?v=xxxxxxx` 的指纹
- 直接上强缓存 `Cache-Control: max-age=31536000,immutable`
- 除非依赖版本变化，否则不会再产生 HTTP 请求

3. 优化源文件请求

- 源文件返回的 header 中带有 etag
- 服务器会保存每个文件的更新状态，没有改动的文件直接返回 `304 Not Modified``

整体的性能取舍

- 🚀 服务器启动：极大加快
- ❌ 第一次页面加载会稍慢 (视加载页面用到的源文件数量而定)
- ✅ 服务器启动 + 页面加载整体依然大大加快
- 🟢 启动后全页面刷新速度不变
- 🚀 热更新：极大加快 (项目越大越明显)

进一步优化的空间

- 针对源文件的文件系统缓存 -> 加快热启动的首次页面加载
- 用原生模块替换现有的模块改写链路
 - 目前使用的是 ``acorn`` + ``magic-string``
 - 参考: Parcel 2 将它的 JS 处理逻辑用 SWC + Rust 重写也获得了明显的性能提升

生产构建

为什么生产还是要打包？

- 简单来说：不打包的部署模式会导致大量的级联 HTTP 请求 (network waterfall)，对缓存策略也有复杂的要求，目前来说依然不如打包的部署策略。

为什么用 Rollup 打包？

- Rollup 是一个为 ES module 而生的打包工具，并且有成熟的 tree-shaking 实现
- 针对 **应用打包** 的场景比 esbuild 成熟很多
(对代码分割和相关的优化处理提供更深入的控制)

没错，Rollup 确实比 esbuild 慢.

但是“打包”只是整个构建的一个部分

- 用 Vue, Svelte 或者依赖 Babel 的方案时，时间依然会消耗在 Node.js 中。
- 生产打包的频率比起开发时的启动和更新低很多

采用 Rollup 让 Vite 能够做到这些

(用 esbuild 就很难)

- 自动 CSS 代码分割
- 对异步懒加载请求的自动优化
- 手动的代码分割控制

合理的默认配置

Vite 开箱即用的功能等价于:

- ``webpack``
- ``webpack-dev-server``
- ``css-loader``
- ``style-loader``
- ``postcss-loader``
- ``esbuild-loader``
- ``file-loader``
- ``wasm-loader``
- ``MiniCssExtractPlugin``
- ...以及大量把这些东西串在一起的配置

重复的配置 ==> 约定

- 多年 vue-cli 和生态其它工具的积累形成的共识

Vite 的哲学:

The 90% Happy Path

- 针对 90% 的主流用户需求进行优化
- 大部分进阶需求可以通过插件支持
- 长尾的特殊需求留给其它工具也无妨

进阶用户: 插件系统

- 开发服务器和生产构建共享的插件机制
- 基于 Rollup 插件 API 的超集
- 提供额外的开发服务器 / 转换 HTML / 自定义热更新的能力

SSR Module Loader

- Vite 内置的服务端渲染模块加载器
- 对 ES modules 源码按需转化为虚拟模块从而在 Node.js 中运行（不需要原生 Node.js ESM 支持）
- 精确的缓存失效，无需打包（和浏览器端的热更新类似）

Vite 作为 上层 SSR 框架的基础

- Svelte Kit
- Ream
- vite-plugin-ssr
- vite-ssr
- Marko + Vite

Vite's 的插件系统强大到足够让上层的 SSR 框架逻辑直接包含在一个 Vite 插件中

Vite 的实现离不开以下项目



Rollup



esbuild

其它对 Vite 的设计有启发的方案

- Snowpack
- WMR
- @web/dev-server
- Parcel

了解更多

- 官网 (vitejs.dev)
- 中文文档 (cn.vitejs.dev)
- Twitter (@vite_js)

谢谢！

极客时间 SVIP团队体验卡

畅学千门IT开发实战课



「扫码免费领课」

