

## 1.什么是vite

---



## 2.为什么用vite

---

### 做法：

vite：基于浏览器原生 ES imports 的开发服务器，按需编译。启动前不需要分析模块的依赖和编译

webpack：需要收集所有模块的依赖关系(生成AST)，再打包成bundle.js后启动服务器。

### 为什么webpack要打包（浏览器没模块、http并发）

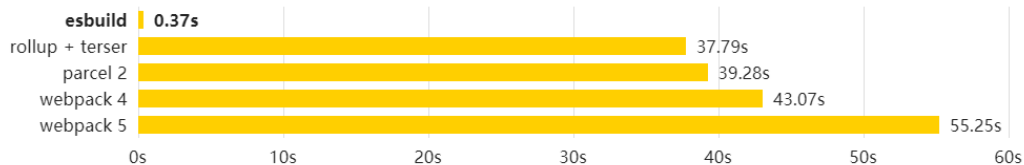
1. 以前浏览器环境不支持模块化，直到 ESM 出现
2. 零散模块文件会产生大量的 http 请求（chrome的并发上限是6）

### vite的优势：

1. 服务器启动速度快
2. 热更新的速度不会随着模块增多而变慢。

### vite本地开发不打包，怎么解决上述问题

1. 利用浏览器原生 ESM，按需编译，只加载当前页面用到的文件
2. 浏览器无法直接使用 .vue，jsx 等，需服务端返回编译后的文件
  - vue 文件使用 sfc 模块（@vue/compiler-sfc）编译
  - ts、tsx 等文件使用 esbuild 解析（利用 go 语言处理高并发的优势），解析 ts，jsx 比 tsc 还快很多



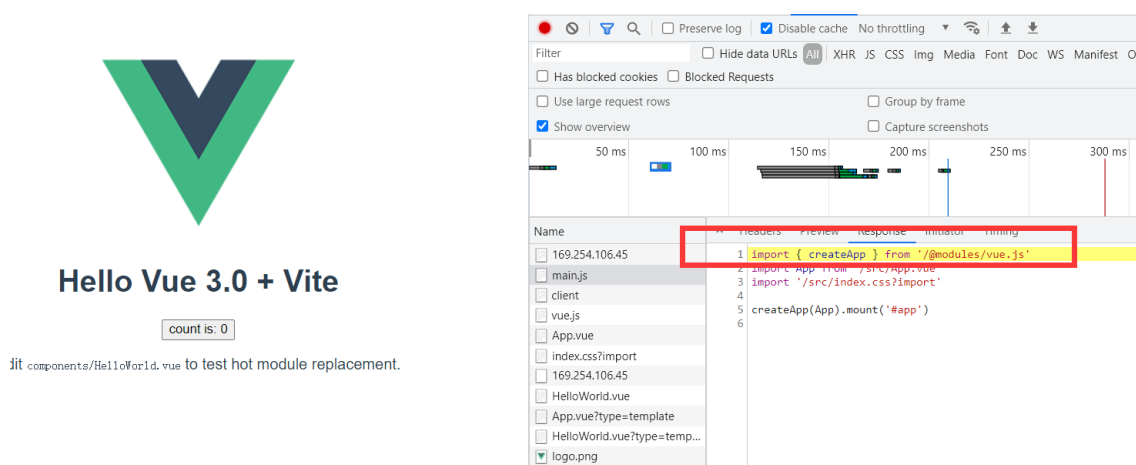
Above: the time to do a production bundle of 10 copies of the [three.js](#) library from scratch using default settings, including minification and source maps. More info [here](#).

## 2. “裸”模块相对地址转换（主要是node\_modules里）

裸模块指引入路径不明确，缺少`'/'`、 `'./'` 或  `'../'`。比如

```
import { createApp } from 'vue';
```

如果参数字符串不以`“./”`或`“../”`开头，则表示加载的是一个默认提供的核心模块（位于Node的系统安装目录中），或者一个位于各级node\_modules目录的已安装模块（全局安装或局部安装）。



可以看到浏览器得到的文件是服务端改写过的，引入模块 `from 'vue'` 被转换成了 `from '/@modules/vue.js'`，不难想到vite是把node\_modules里的模块临时存放到@module中供给浏览器。

## 3.怎么用vite

**开发：**基本是开箱即用，内置了常见的插件配置，有需要时直接安装相应插件。

**打包上线：**

**vite** 目前是直接用 **rollup** 打包

虽然 **esbuild** 快得惊人，并且已经是一个在构建库方面比较出色的工具，但一些针对构建应用的重要功能仍然还在持续开发中——特别是代码分割和CSS处理方面。就目前来说，Rollup在应用打包方面更加成熟和灵活。尽管如此，当未来这些功能稳定后，我们也不排除使用 **esbuild** 作为生产构建器的可能。

## 4.目前状况

目前所有主流的浏览器的最新版本都支持了ES模块（除了ie11）

插件生态（官方文档？）

二级路径问题？

## 错误图片

