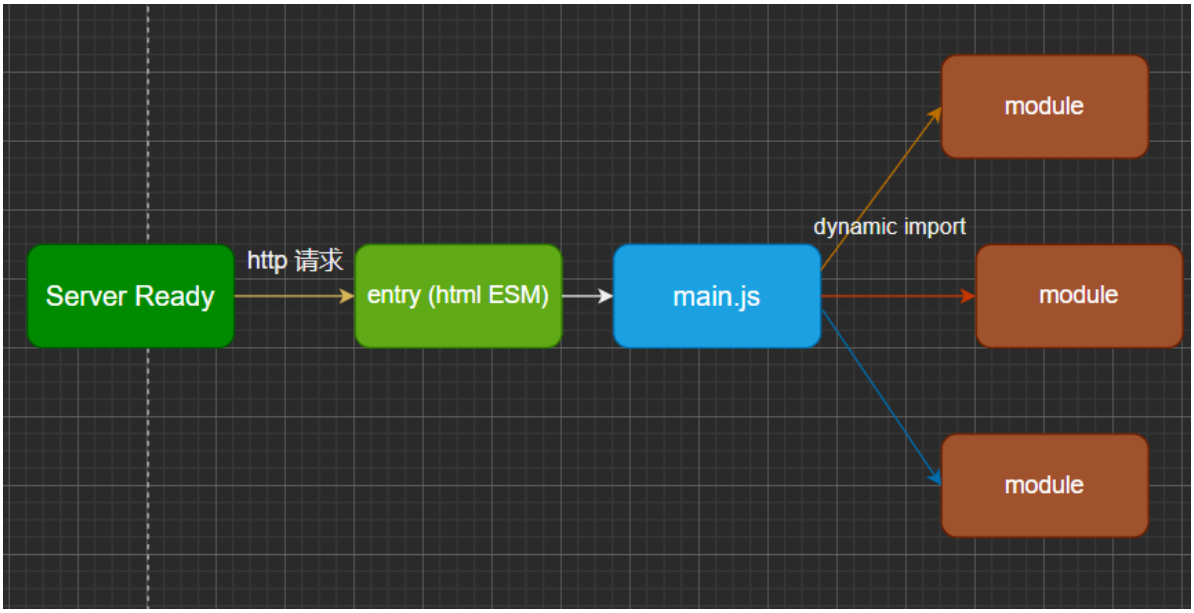
# 1.什么是vite

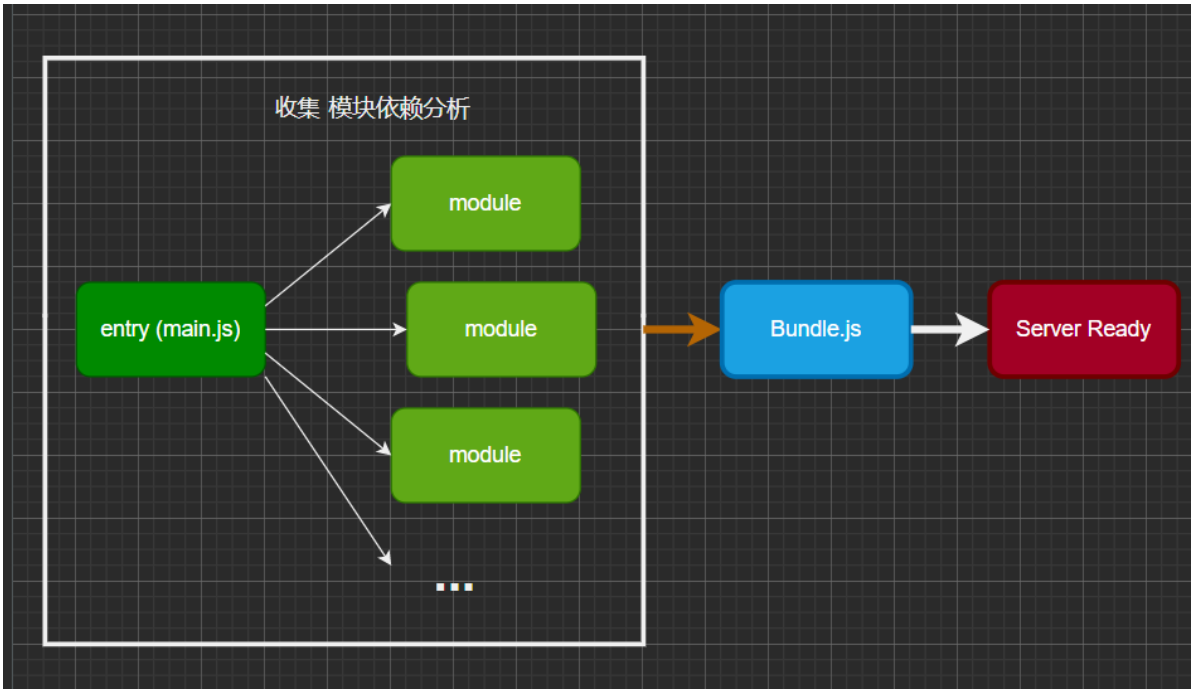

# 2.为什么用vite

## 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)，再编译成bundle.js后完成冷启动。




**为什么打包**

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）

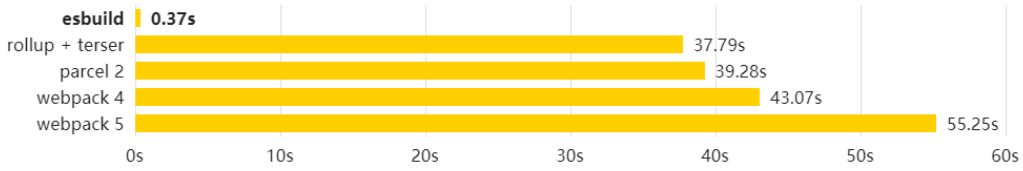
个人觉得本地开发实际上没有打包的必要性（其他观点？），要利用缓存的优势的话，可以到要发布上线时再进行打包。

## vite的优势：

1. 本地开发服务器冷启动速度快
2. 热更新的速度不会随着模块增多而变慢。

## vite原理

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`快20-30倍



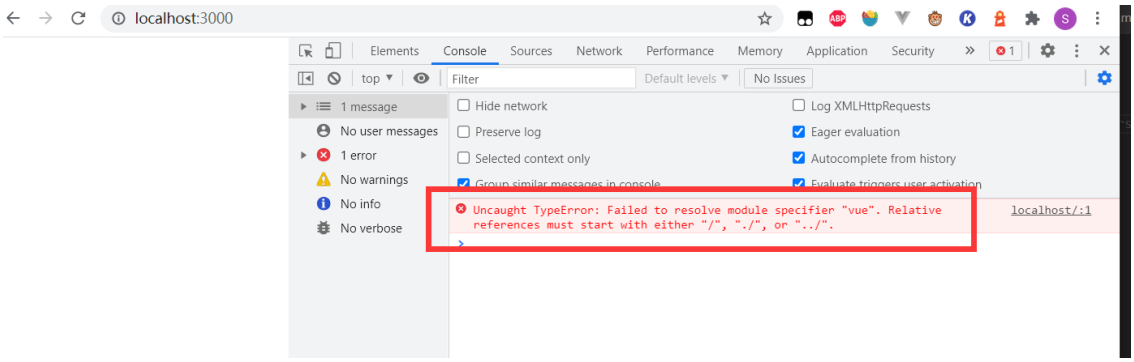

*Above: the time to do a production bundle of 10 copies of the three.js library from scratch using default settings, including minification and source maps. More info here.*

3. **NPM 依赖解析和预构建**

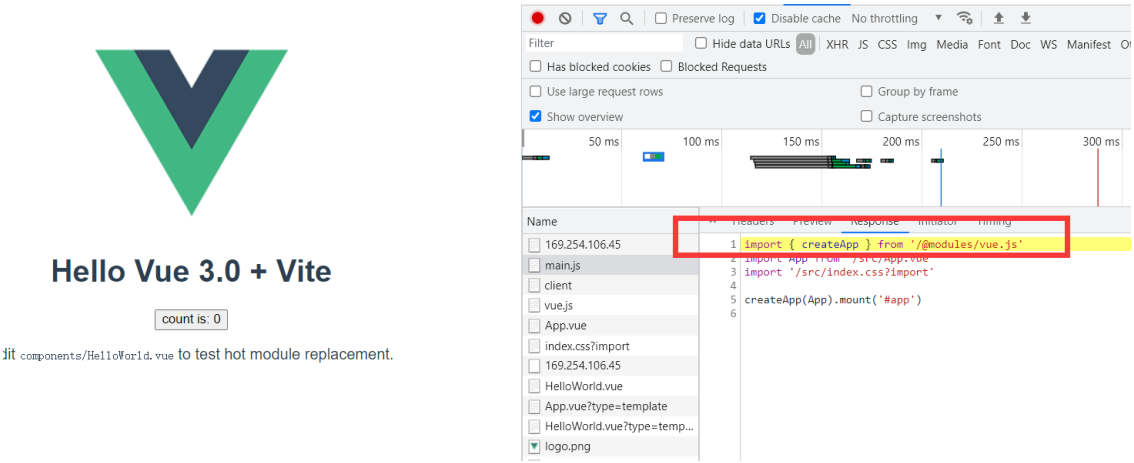
   对“裸”模块路径转换

> 裸模块指引入路径不明确，缺少'/'，'./' 或 '../'。比如
>
> ```
> import { createApp } from 'vue'; // 裸模块
> ```
>
> 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）。

如下，未能正确import模块




vite对这种裸模块进行了路径转换




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'`，

4. 图片等静态资源则直接返回解析后的路径
5. 热更新
    - 监听：websocket监听文件修改
    - 缓存：利用HTTP协商缓存（源码）、强缓存（依赖模块）

# 3.使用

**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`1.0是直接用`rollup`打包。为什么不用esbuild

> 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，目前还比较少，但由于用的是rollup打包，实际上可以兼容rollup生态中的插件
  vite2.0官方文档里已经开始完善插件相关的信息。

二级路径问题?