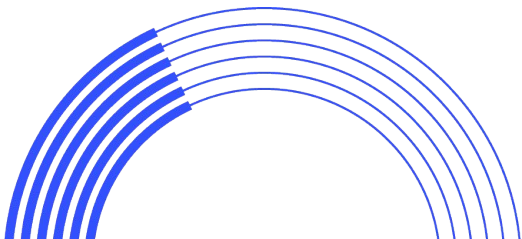
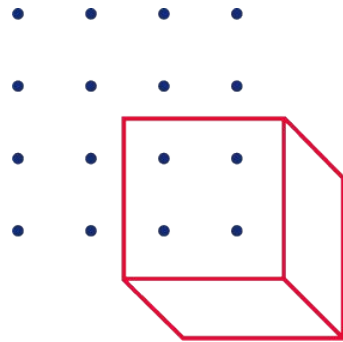




Vite Intro

baurine @ 2021.05.6



What

- Vite, 法语, 快的意思
- “下一代” 前端开发与构建工具
 - 开发环境, 基于浏览器原生支持 ESM, 无须 bundle, 开发服务器启动及热更新时间约为百 ms 级别, 远胜 Webpack
 - 生产环境, 使用 Rollup 构建打包

对比

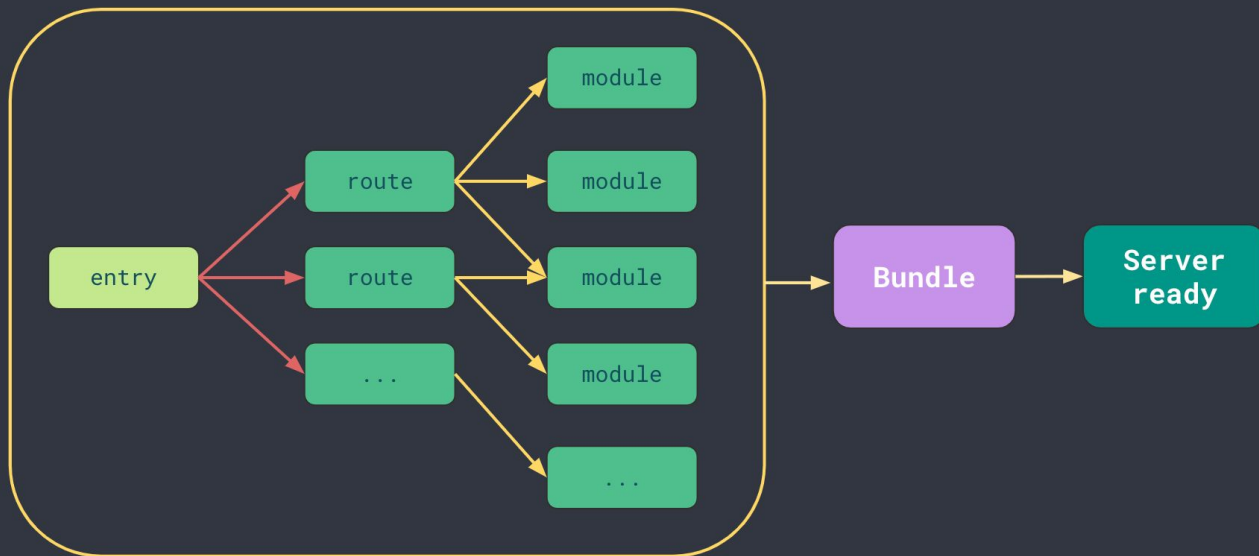
- [replit](#)
- [another example](#)

为什么要 bundle

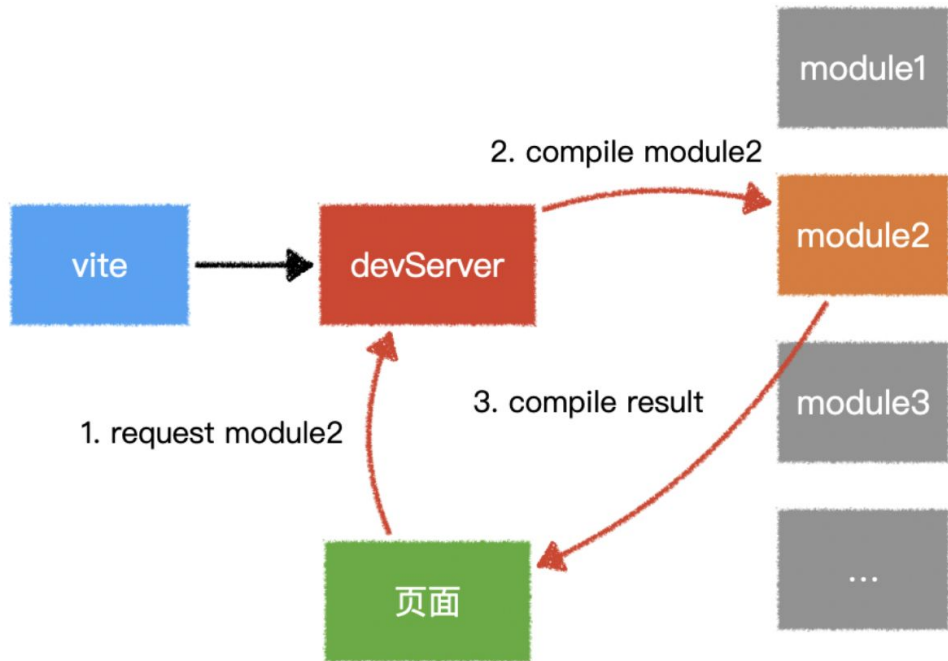
- 为什么后端的 node.js 不需要 bundle?
 - 不需要传输
 - CommonJS 模块管理
- 前端
 - 减少请求数及体积
 - 无模块管理 -> 原生 ESM 还没有普及

Webpack 怎么 bundle

Bundle based dev server



Vite 如何工作



不 bundle 怎么工作？

- aka 什么是浏览器的原生 ESM 支持
- ESM: [ES Module](#), import / export (我们不是已经在用了吗？ - [webpack 进行了转换](#))
- 举个例子？

native esm

- [demo](#)

```
// index.html
<script type="module">
  import { sayHello } from './main.js'
  // import './style.css' // doesn't work

  const btn = document.getElementById('btn_hello')
  btn.onclick = function () {
    const name =
document.getElementById('txt_name').value
    sayHello(name)
  }
</script>
```

```
> assets
> node_modules
❖ .gitignore
<> index.html
JS main.js
{} package.json
# style.css
JS util.js
🐶 yarn.lock
```


native esm



```
// main.js
import { nowTime, minItem } from './util.js'

export function sayHello(name) {
  const arr = [5, 2, 8, 1, 3]
  window.alert(
    `hello ${
      name || 'bro'
    }, now the time is ${nowTime()}, min of ${arr} is ${minItem(arr)}`
  )
}
```

native esm











```
// util.js
import min from './node_modules/lodash-es/min.js'
// import {min} from 'lodash'

export function nowTime() {
  return Date().toString()
}

export function minItem(arr) {
  return min(arr)
}
```

native esm

浏览器是怎么加载的？

Name	Status	Protocol	Type	Initiator	Size
 localhost	304 Not Modified	http/1.1	document	Other	113 B 538 B
 vite-logo.svg /assets	304 Not Modified	http/1.1	svg+xml	(index) Parser	113 B 1.5 kB
 style.css	304 Not Modified	http/1.1	stylesheet	(index) Parser	113 B 75 B
 main.js	304 Not Modified	http/1.1	script	(index):11 Script	113 B 236 B
 util.js	304 Not Modified	http/1.1	script	main.js :1 Script	113 B 281 B
 min.js /node_modules/lodash-es	304 Not Modified	http/1.1	script	util.js :1 Script	113 B 612 B
 _baseExtremum.js /node_modules/lodash-es	304 Not Modified	http/1.1	script	min.js :1 Script	113 B 895 B
 _baseLt.js /node_modules/lodash-es	304 Not Modified	http/1.1	script	min.js :2 Script	113 B 352 B

native esm



使用第三方库？比如 lodash

```
$ yarn add lodash-es // not lodash
```

```
// util.js
```

```
import min from './node_modules/lodash-es/min.js'
```

```
// import { min } from 'lodash-es' // doesn't work
```

```
export function minItem(arr) {  
  return min(arr)  
}
```

限制：

- 只能使用 esm 打包的库
- 必须用完整路径

更多需求



浏览器原生支持 ESM, 仅仅帮助我们免去了 bundle 的过程

其它问题:

- 如何使用 cjs 格式的 npm 包, 继续使用 `import {min} from 'lodash'` 的写法
- 如何使用 typescript
- 如何使用 jsx / vue
- 如何使用 scss/less
- 如何模块热更新 (HMR)
- more...

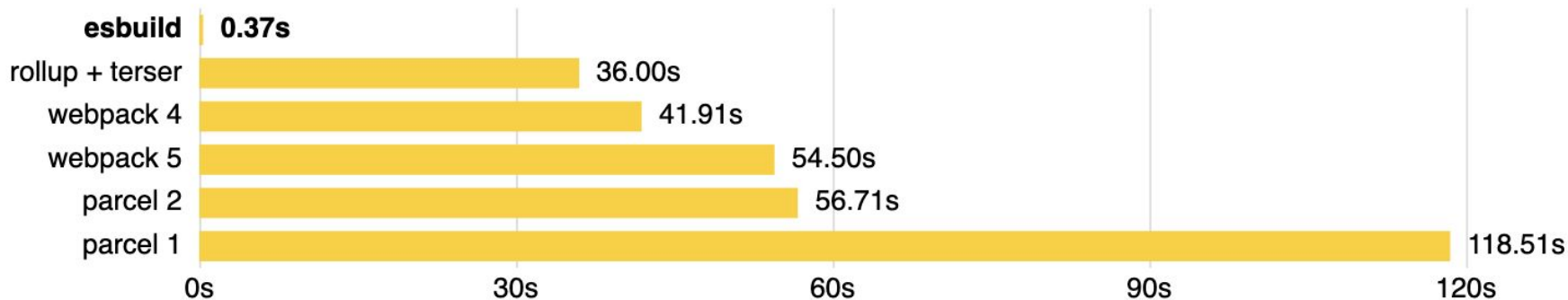
因此, 我们需要在此之上, 提供一整套解决方案。vite 就此出现。类似的还有 snowpack 等。

vite 的工作原理

- 将应用代码分两块
 - 依赖, 比如 react, lodash 这些, 不会变动。使用 esbuild (replace babel) 预构建依赖
 - 将 cjs 模块转换成 esm
 - 将分散的小模块 bundle 成一个文件
 - 源码, 即自己的逻辑代码
 - jsx, css, .vue 组件...
 - 请求时按需实时转码













esbuild

An extremely fast JavaScript bundler and minifier
10~100x faster






vanilla (without vite)

```
// util.js
import min from './node_modules/lodash-es/min.js'
```

 min.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	util.js:1 Script
 _baseExtremum.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	min.js:1 Script
 _baseLt.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	min.js:2 Script
 identity.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	min.js:3 Script
 isSymbol.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	_baseEx... Script
 _baseGetTag.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	isSymbo... Script
 isObjectLike.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	isSymbo... Script
 _Symbol.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	_baseGe... Script
 _getRawTag.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	_baseGe... Script
 _objectToString.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	_baseGe... Script
 _root.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	_Symbol... Script
 _freeGlobal.js	/node_modules/lodash-es	304 Not Modified	http/1.1	script	_root.js:1 Script

with vite

```
// main.tsx
import { min } from 'lodash' // lodash-es
```

Name	Status	Protocol	Type	Initiator	Size
 react.js?v=bbdf76cd /node_modules/.vite	200 OK	http/1.1	script	<u>main.tsx:1</u> Script	(memory cache) 538 B
 react-dom.js?v=bbdf76cd /node_modules/.vite	200 OK	http/1.1	script	<u>main.tsx:1</u> Script	(memory cache) 633 B
 lodash.js?v=bbdf76cd /node_modules/.vite	200 OK	http/1.1	script	<u>main.tsx:11</u> Script	(memory cache) 1.1 MB

- 可以使用 cjs 打包的库, 将 cjs 转换成了 esm
- 将小模块 bundle 成一个文件

依赖预构建

```
vite-tailwind > node_modules > .vite > {} _metadata.json > ...
```

```
15     "react": {
16       "file": "/Volumes/T7/codes/personal/try-vite/
           vite-tailwind/node_modules/.vite/react.js",
17       "src": "/Volumes/T7/codes/personal/try-vite/
           vite-tailwind/node_modules/react/index.js",
18       "needsInterop": true
19     },
20     "react-dom": {
21       "file": "/Volumes/T7/codes/personal/try-vite/
           vite-tailwind/node_modules/.vite/react-dom.js",
22       "src": "/Volumes/T7/codes/personal/try-vite/
           vite-tailwind/node_modules/react-dom/index.js",
23       "needsInterop": true
24     },
25     "react-router-dom": {
26       "file": "/Volumes/T7/codes/personal/try-vite/
           vite-tailwind/node_modules/.vite/react-router-dom.js",
27       "src": "/Volumes/T7/codes/personal/try-vite/
           vite-tailwind/node_modules/react-router-dom/esm/
           react-router-dom.js",
28       "needsInterop": false
29     },
```

node_modules

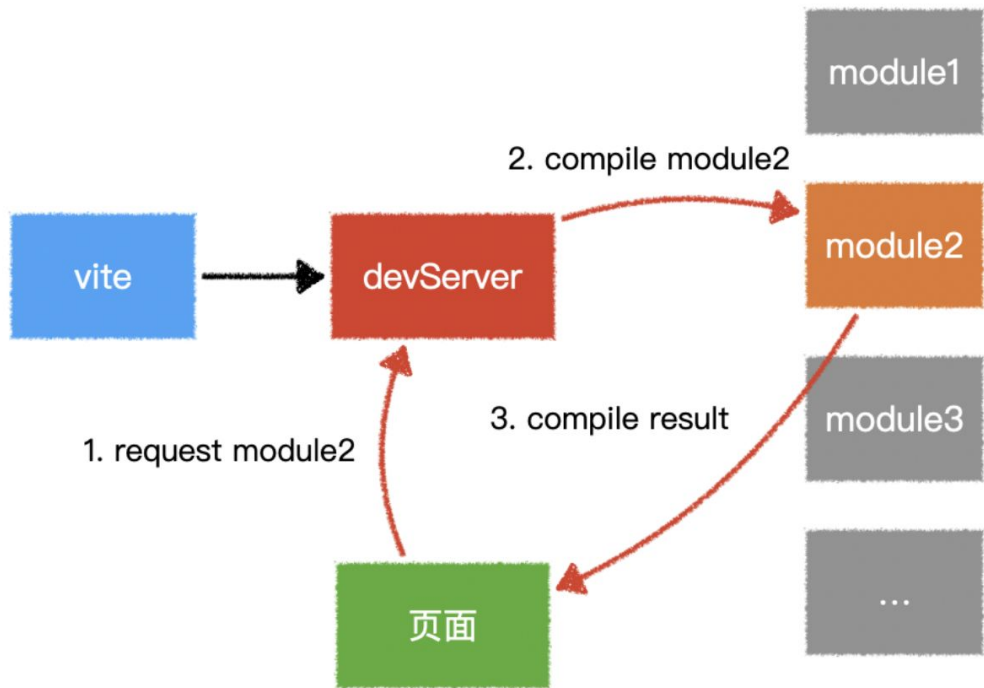
> .bin

> .vite

{} _metadata.json

```
JS @headlessui_react.js
JS @headlessui_react.js.map
JS @heroicons_react_outline.js
JS @heroicons_react_outline.js.map
JS @heroicons_react_solid.js
JS @heroicons_react_solid.js.map
JS chunk-KHGO2BKN.js
JS chunk-KHGO2BKN.js.map
JS chunk-NS64VXF3.js
JS chunk-NS64VXF3.js.map
JS chunk-RZYNWZ5R.js
JS chunk-RZYNWZ5R.js.map
JS chunk-XVNMABU7.js
JS chunk-XVNMABU7.js.map
JS lodash.js
JS lodash.js.map
JS react-dom.js
JS react-dom.js.map
JS react-router-dom.js
JS react-router-dom.js.map
JS react.js
JS react.js.map
```

处理源码



vite 启动了一个开发服务器，拦截所有请求，针对不同的请求，进行不同的转义处理。

simple-vite

```
const handler = async ctx => {
  const {request: {url}} = ctx
  if (url === '/') {
    handleHTML(ctx)
  } else if (url.endsWith(".js")) {
    handleJS(ctx, url)
  } else if (url.startsWith("/@modules/")) {
    handleModule(ctx, url)
  } else if (url.indexOf(".css") > -1) {
    handleCSS(ctx, url)
  } else if (url.indexOf(".vue") > -1) {
    handleVue(ctx, url)
  }
}
```

处理 ts, jsx

// 原始代码, main.tsx

```
import React from 'react'
import ReactDOM from 'react-dom'
```

```
import './index.css'
import App from './App'
```

```
ReactDOM.render(
  <React.StrictMode>
    <App />
  </React.StrictMode>,
  document.getElementById('root')
)
```

//...

vite 开发服务器使用 esbuild 实时转码后返回给浏览器的内容

```
import __vite__cjsImport0_react from
"/node_modules/.vite/react.js?v=bbdf76cd";
const React = __vite__cjsImport0_react.__esModule ?
__vite__cjsImport0_react.default :
__vite__cjsImport0_react;
import __vite__cjsImport1_reactDom from
"/node_modules/.vite/react-dom.js?v=bbdf76cd";
const ReactDOM =
__vite__cjsImport1_reactDom.__esModule ?
__vite__cjsImport1_reactDom.default :
__vite__cjsImport1_reactDom;
import "/src/index.css";
import App from "/src/App.tsx";
ReactDOM.render(/* #__PURE__*/
React.createElement(React.StrictMode, null, /*
#__PURE__*/
React.createElement(App, null)),
document.getElementById("root"));
```

处理 CSS




```
// 原始代码, main.tsx
import './index.css'
import App from './App'
//...
```

```
// index.css
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto',
'Oxygen',
  'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
  sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
  monospace;
}
```

处理 css

实际返回给浏览器的是：

Name	Status	Protocol	Type	Initiator
 index.css /src	304 Not Modified	http/1.1	script ?	main.tsx:2 Script
 App.tsx /src	304 Not Modified	http/1.1	script	main.tsx:3 Script
 logo.svg?import /src	304 Not Modified	http/1.1	script	App.tsx:1 Script
 App.css /src	304 Not Modified	http/1.1	script ?	App.tsx:2 Script

处理 css

vite 开发服务器读取原始 css 内容, 转换成 js 文件返回, css 内容将插入到 style 中

```
import { createHotContext as __vite__createHotContext } from "@vite/client";
import.meta.hot = __vite__createHotContext("/src/index.css");
import { updateStyle, removeStyle } from "@vite/client"
const id = "/Volumes/T7/codes/personal/try-vite/vite-001/src/index.css"
const css = "body {\n  margin: 0;\n  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI',\n  'Roboto', 'Oxygen',\n    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',\n  sans-serif;\n  -webkit-font-smoothing: antialiased;\n  -moz-osx-font-smoothing:\n  grayscale;\n}\n\ncode {\n  font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier\n  New',\n  monospace;\n}\n"
updateStyle(id, css)
import.meta.hot.accept()
export default css
import.meta.hot.prune(() => removeStyle(id))
```

处理 css


```
<!DOCTYPE html>
<html lang="en">
  <head>
    <script type="module" src="/@vite/client"></script>
    <script type="module">...</script>
    <meta charset="UTF-8">
    <link rel="icon" type="image/svg+xml" href="/src/favicon.svg">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vite App</title>
    <style type="text/css">
      body {
        margin: 0;
        font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
          'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
          sans-serif;
        -webkit-font-smoothing: antialiased;
        -moz-osx-font-smoothing: grayscale;
      }

      code {
        font-family: source-code-pro, Menlo, Monaco, Consolas, 'Courier New',
          monospace;
      }
    </style>
```


处理 assets

```
// 原始代码, App.tsx  
import logo from './logo.svg'
```

```
// vite 转码后, 浏览器得到的 App.tsx  
import logo from "/src/logo.svg?import";  
// 这么绕一圈是为了让资源能继续被其他 plugin 处理。html plugin 只做简单的字符串处理  
// 比如加后缀与其他配置。插件化思想, 每个插件做好自己的事  
接着请求 /src/logo.svg?import
```

 logo.svg?import /src	304 Not Modified	http/1.1	script	<u>App.tsx:1</u> Script
---	---------------------	----------	--------	----------------------------

×	Headers	Preview	Response	Initiator
1	export default "/src/logo.svg"			

最后请求 /src/logo.svg 得到 asset

 logo.svg /src	304 Not Modified	http/1.1	svg+xml
---	---------------------	----------	---------

HMR



在 Vite 中, HMR 是在原生 ESM 上执行的。当编辑一个文件时, Vite 只需要精确地使已编辑的模块与其最近的 HMR 边界之间的链失效(大多数时候只需要模块本身), 使 HMR 更新始终快速, 无论应用的大小。

总结 Vite 为什么这么快（开发环境）

- 依赖预构建
- 源码, 无须打包, 按需转义
- 使用 esbuild 将 TypeScript, JSX, TSX 转义成浏览器 ESM, 不使用 babel
- HMR 粒度更小

生产环境为什么还需要 bundle



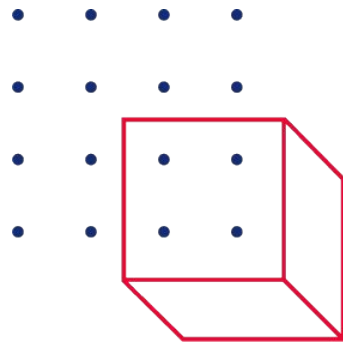
- 开发环境网络加载速度可以忽略, 但生产环境不行
- 即使用上 HTTP2, 过多的请求也会影响加载速度
- esbuild 还不支持 tree-shaking 等特性

限制

- 需要较新的浏览器支持
- legacy
-

参考

- [官方文档](#)
- [Webpack to Vite, 为开发提速！](#)
- [下一代前端构建工具 Vite](#)
- [Develop with Vite | Vite快速入门](#)
- [Why We Switched From Webpack To Vite](#)



Q & A

