说说vite打包优化

生产环境中移除log

在我们开发阶段，经常会用console.log来打断点，测试代码，但这是给我们开发人员看的，代码上线后，这肯定不能被用户看到，所以在打包时可以配置一下 在vite.config.ts中配置如下命令

```
build: {
    //移除生产环境log
    minify: 'terser',
    terserOptions: {
      compress: {
        //生产环境时移除console
        drop_console: true,
        drop_debugger: true,
      },
    },
    //rollup打包后的静态资源名称格式
    rollupOptions: {
      output: {
        chunkFileNames: 'static/js/[name]-[hash].js',
        entryFileNames: 'static/js/[name]-[hash].js',
        assetFileNames: 'static/[ext]/[name]-[hash].[ext]'
      },
    }
  }
```

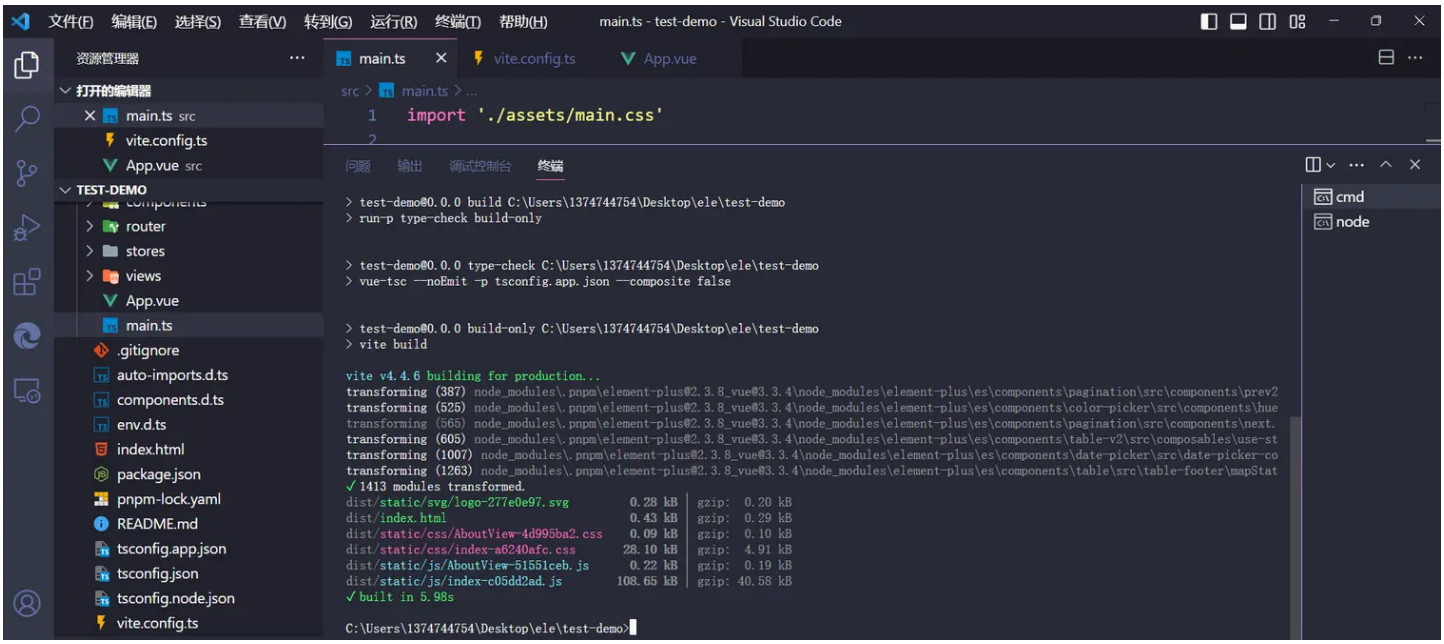
build是和plugin同一级的，注意一下，还有一个注意点，现在vite的terser被移除了,可以通过 `pnpm i terser -D` 来把包下载下来

下面这个因为vite基于rollup打包,而打包后的chunk(代码块)后静态资源名称比较简单,使用命名规则可以确保在每次构建应用程序时，文件的名称都会随着内容的更改而变化，从而帮助缓存管理和版本控制。这样可以避免浏览器缓存旧版本文件的问题，并确保每次部署新的构建版本时，浏览器可以正确加载更新的文件。

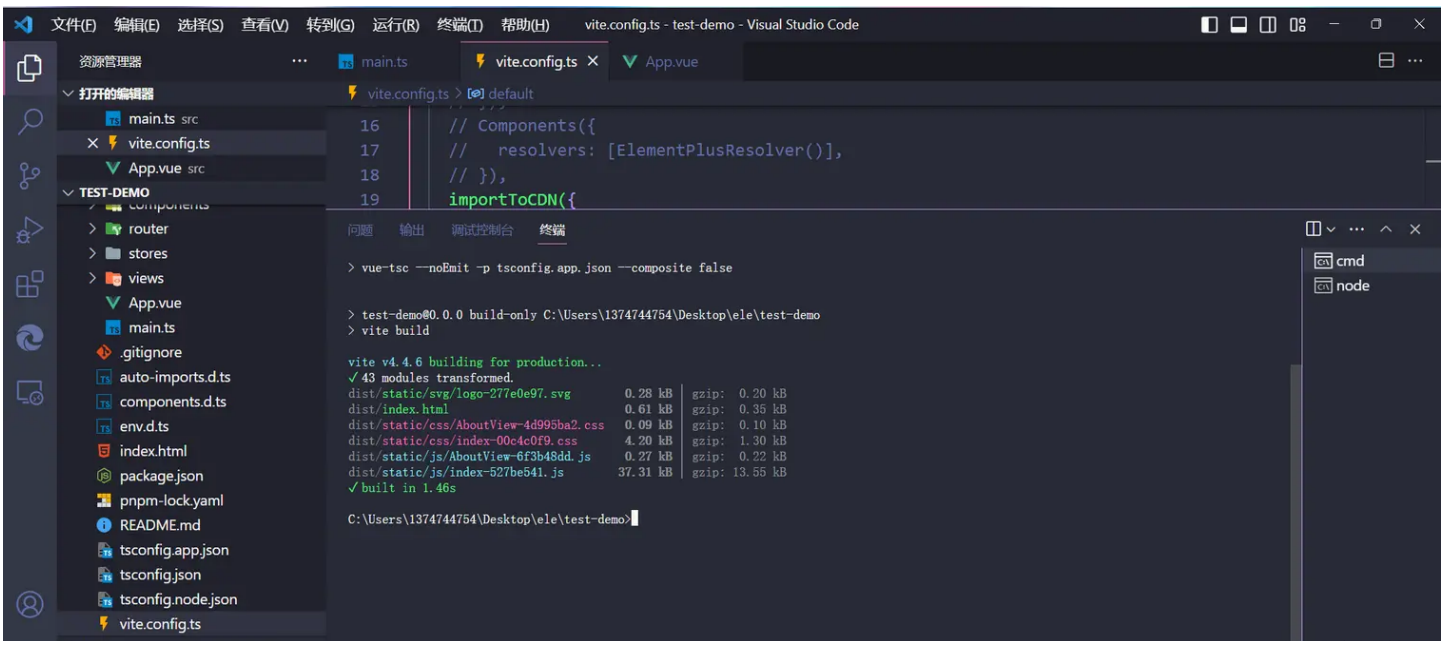
CDN导入

我们常见的element-Plus,在我们考虑到包体积大小时，经常会使用按需引入和自动导入，这两种方式虽然极大的解决了包体积的问题，但是并不是最优化的方案，我们可以来对比一下

这张是用自动导入的，可以看打包时间为5s多



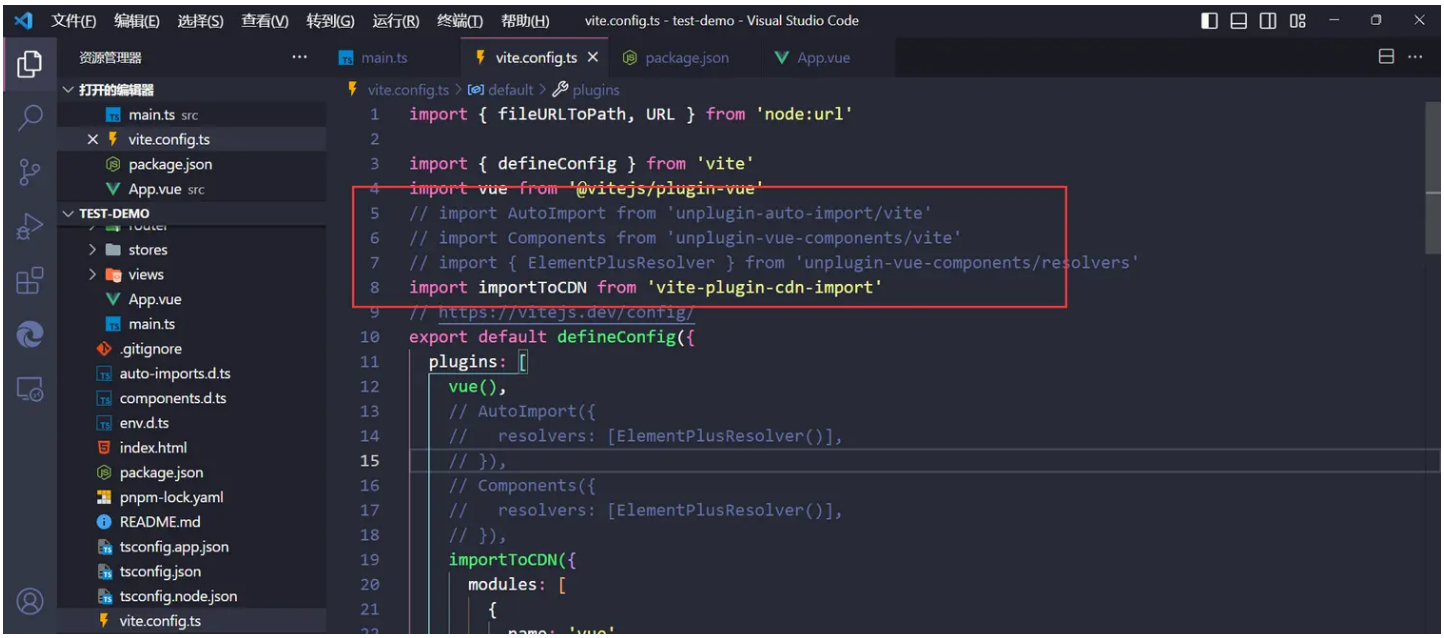
这张是用CDN的，打包只用1s多



可以看到用CDN的速度要快很多,接下来看看用法 用自动导入的就不说了，官网有教程，主要看用CDN的用法

借助一个插件**vite-plugin-cdn-import** `pnpm i vite-plugin-cdn-import -D` github地址

接着在vite.config.ts中引入




```
importToCDN({
      modules: [
        {
          name: 'vue',
          var: 'Vue',
          path: "https://unpkg.com/vue@next"
        },

        {
          name: "element-plus",
          var: 'ElementPlus',
          path: "https://unpkg.com/element-plus",
          css: "https://unpkg.com/element-plus/dist/index.css"
        }
      ]
    })
```

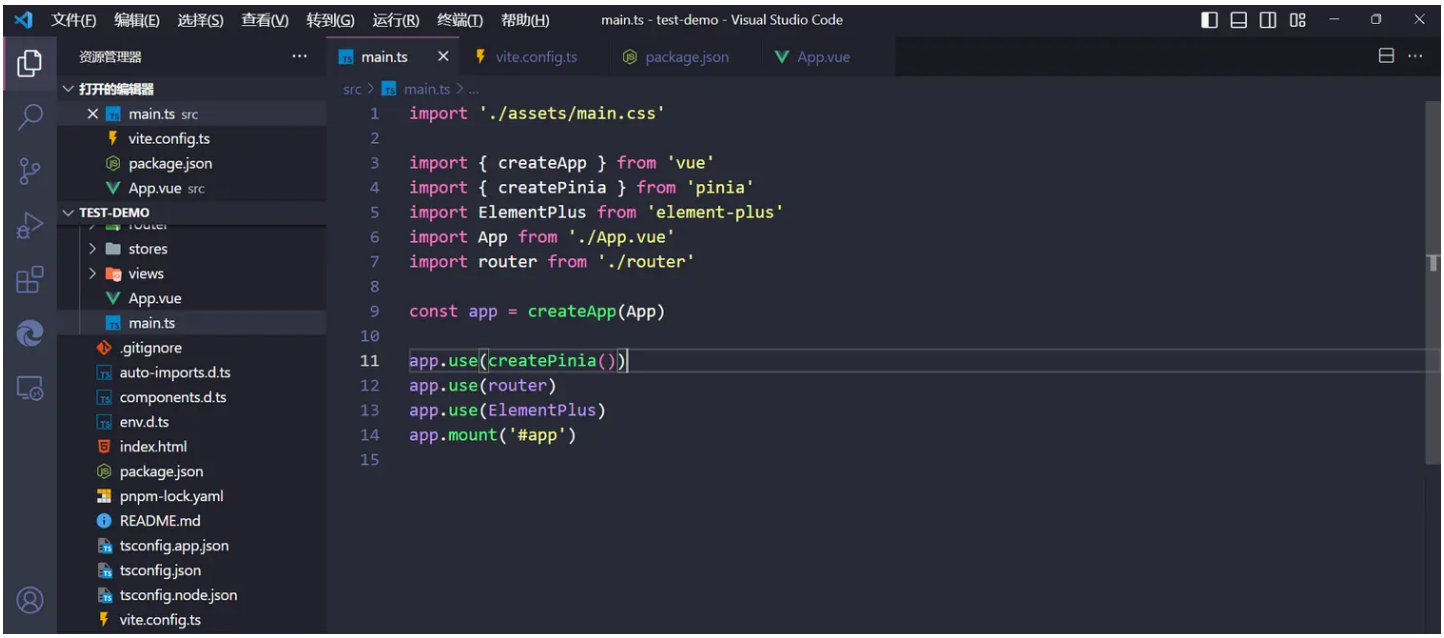
这里的CDN地址可以去bootCDN复制，因为这里借助了vue所以也要引入vue的CDN，另外注意一点，我们用自动导入的时候，这几个地方不用写

```ts
// main.ts
import { createApp } from 'vue'
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'
import App from './App.vue'

const app = createApp(App)

app.use(ElementPlus)
app.mount('#app')
```

但是用CDN的时候,是有不同的,跟我这里一样的配置即可



到这里就已经配置好了，但是我打包之后预览的时候发现，报错了，路径找不到，这是因为，对于pinia这种它还借助了vue-demi这个包，因此还要引入vue-demi，注意要在pinia之前引入，因此，放在vue后面最佳

```ts
importToCDN({
      modules: [
        {
          name: 'vue',
          var: 'Vue',
          path: "https://unpkg.com/vue@next"
        },
        {
          name: 'vue-demi',
          var: 'VueDemi',
          path: 'https://cdn.bootcdn.net/ajax/libs/vue-demi/0.14.0/index.iife.js'
```

```
        },
        {
          name: "element-plus",
          var: 'ElementPlus',
          path: "https://unpkg.com/element-plus",
          css: "https://unpkg.com/element-plus/dist/index.css"
        }
      ]
    })
```

这是最终的配置

图片压缩

我们的图片资源，有点很大，几M,10几M的也有很多，对于我们的代码，几百kb都很大了，更别说这个图片资源了，因此我们在打包前可以压缩一下

对于压缩图片我们可以选择**手动压缩**和**自动压缩**，对于图片压缩，我的建议是先手动压缩再自动压缩，这里推荐一个在线的免费压缩图片网站 (TinyPNG – 智能压缩 WebP、PNG 和 JPEG 图像),可以一次压缩20-30张，并且无损，还是很香的

对于自动压缩，我们在打包时借助插件来帮我们实现 **vite-plugin-imagemin** ,github地址,用法也是在官网写的很清楚，这里的vbenjs很多人都听过，是一个很强的开源项目，它有很多的插件,回到我们这里,首先我们 `pnpm i vite-plugin-imagemin -D` ，先把插件下载下来，这个插件有点点问题，我下载的时候有点慢，卡住了一样，我就挂梯子下了，发现很快，所以需要魔法，另外大家也可以试试 `cnpm i vite-plugin-imagemin -D` 这个命令就不用挂梯子了

看一下在vite中的配置

```
import viteImagemin from 'vite-plugin-imagemin'
//图片压缩
    viteImagemin({
      gifsicle: {
        optimizationLevel: 7,
        interlaced: false,
      },
      optipng: {
        optimizationLevel: 7,
      },
      mozjpeg: {
        quality: 20,
      },
      pngquant: {
        quality: [0.8, 0.9],
        speed: 4,
```

```
      },
      svgo: {
        plugins: [
          {
            name: 'removeViewBox',
          },
          {
            name: 'removeEmptyAttrs',
            active: false,
          },
        ],
      },
    }),
```

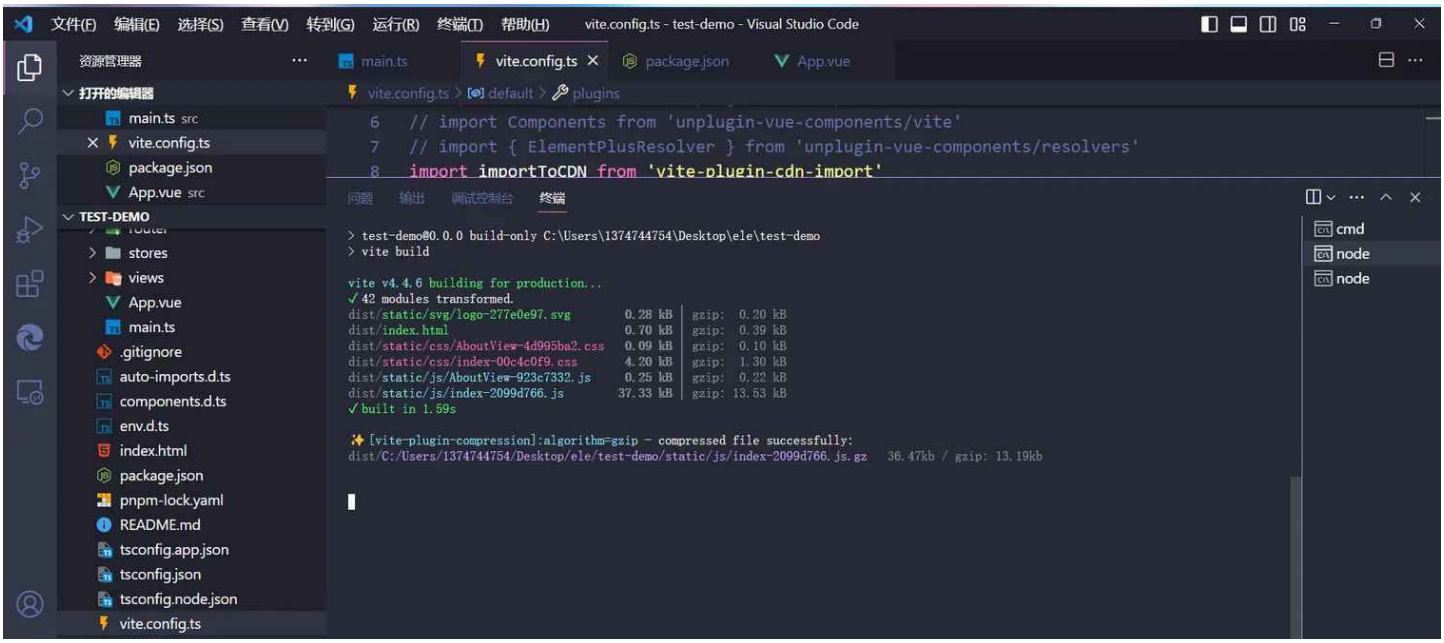
这样就能够在打包时对图片进行压缩，这里也是无损的，跟原图效果差不多。

gzip代码压缩配置

对于gzip代码压缩，有的人说我服务器端压缩了，为啥还要用vite进行gzip打包压缩，这里就涉及到一个知识点，动态压缩和静态压缩，服务器是动态压缩，占用cpu性能，vite等打包好gzip部署后走的静态压缩，服务器压力小，现在google开源的br号称比gizp还小，之前看好像只支持http,不支持https，后面可能也会支持的，先回到我们这里，同样用到一个插件**vite-plugin-compression** `pnpm i vite-plugin-compression -D` ,这个也是vben的插件

```
import viteCompression from 'vite-plugin-compression'
//开启gzip
    viteCompression({
      //生成压缩包gz
      verbose: true,
      disable: false,
      threshold: 10240,
      algorithm: 'gzip',
      ext: '.gz',
    }),
```

注意注释开启gzip下面的代码都是写在plugin里面的，别写错位置了，上面的同样如此, 这样之后我们的代码就又压缩了一下,可以对比前后的代码体积


文件(F) 编辑(E) 选择(S) 查看(V) 转到(G) 运行(R) 终端(T) 帮助(H) vite.config.ts - test-demo - Visual Studio Code
资源管理器
打开的编辑器
main.ts src
vite.config.ts
package.json
App.vue src
TEST-DEMO
stores
views
App.vue
main.ts
.gitignore
auto-imports.d.ts
components.d.ts
env.d.ts
index.html
package.json
pnpm-lock.yaml
README.md
tsconfig.app.json
tsconfig.json
tsconfig.node.json
vite.config.ts
main.ts
vite.config.ts
package.json
App.vue
vite.config.ts > default > plugins
6 // import Components from 'unplugin-vue-components/vite'
7 // import { ElementPlusResolver } from 'unplugin-vue-components/resolvers'
8 import importToCDN from 'vite-plugin-cdn-import'
问题 输出 调试控制台 终端
> test-demo@0.0.0 build-only C:\Users\1374744754\Desktop\ele\test-demo
> vite build
vite v4.4.6 building for production...
✓ 42 modules transformed.
dist/static/svg/logo-277e0e97.svg 0.28 kB | gzip: 0.20 kB
dist/index.html 0.70 kB | gzip: 0.39 kB
dist/static/css/AboutView-4d995ba2.css 0.09 kB | gzip: 0.10 kB
dist/static/css/index-00c4c0f9.css 4.20 kB | gzip: 1.30 kB
dist/static/js/AboutView-923c7332.js 0.25 kB | gzip: 0.22 kB
dist/static/js/index-2099d766.js 37.33 kB | gzip: 13.53 kB
✓ built in 1.59s
✨ [vite-plugin-compression]:algorithm=gzip - compressed file successfully:
dist/C:/Users/1374744754/Desktop/ele/test-demo/static/js/index-2099d766.js.gz 36.47kb / gzip: 13.19kb
cmd
node
node

