Webpack 性能优化

Webpack 可以通过以下方法进行性能优化：

1. 减少入口文件大小：将入口文件拆分为多个较小的模块，使用动态导入（dynamic imports）按需加载，减少初始加载的文件大小。
2. 代码分割（Code Splitting）：通过配置Webpack的代码分割功能，将项目代码分割成多个块（chunks），在需要的时候按需加载。
3. 使用Tree Shaking：通过配置Webpack的Tree Shaking功能，只保留项目中实际使用到的代码，剔除未使用的代码，减少打包后的文件大小。
4. 并行构建：使用Webpack的thread-loader或happypack插件将任务分发给多个子进程并行处理，提高构建速度。
5. 优化加载速度：使用Webpack的插件，如MiniCssExtractPlugin来提取CSS代码，使用babel-loader的缓存机制等，以减少构建时间和加载时间。
6. 优化文件体积：使用Webpack的压缩插件如terser-webpack-plugin来压缩JavaScript代码，使用cssnano等工具压缩CSS代码，减小文件体积。
7. 使用缓存：配置Webpack的缓存功能，使得构建过程中只重新构建发生更改的部分，而不是每次都重新构建整个项目。
8. 懒加载与预加载：对于大型应用，使用Webpack的懒加载（Dynamic Import）功能，按需加载非关键性资源；同时可以使用预加载（Preload）和预解析（Prefetch）机制提前加载关键资源。
9. 优化图片资源：对于图片资源，可以使用Webpack的url-loader或file-loader来压缩和处理图片，并根据需要进行懒加载或响应式加载。
10. 配置合理的模块解析规则：通过配置Webpack的resolve选项，设置合适的模块解析规则，避免过多的文件查找和解析过程。

- 小图片 base64 编码
- bundle 和 hash
- 懒加载
- 提取公共代码
- 使用 CDN 加速
- 使用 production
- Scope Hosting

1. 产出代码

- 体积更小
- 合理分包，不重复加载
- 速度更快，内存使用更小