首先网络层面：

减少dns查询次数，减少域名数，dns预获取

减少http请求数，比如css雪碧图，降低资源大小，资源进行压缩，图片压缩

利用http缓存，缓存可缓存的静态资源

Tcp开启keep-alive保持连接

Cdn缓存分发

服务端接口合并，一个页面接口太多，合并成一个接口

渲染层面：

Css、js文件加载顺序，css放头部，js放底部

避免图片空src

图片懒加载

组件库按需加载

减少js动画使用，或使用css3动画代替

Webpack优化

1. 利用externals外部扩展，排除将import引入的第三方包打包到bundle中，在runtime运行时，利用cdn加载，并结合html-webpack-plugin，动态注入到index.html中
2. 利用compress-webpack-plugin，开启gzip压缩
3. 开启tree-shaking，webpack会跟踪整个应用的import/export语句，如果发现导入的东西没有被使用，则被过滤掉。
4. 开启babel缓存，设置cacheDirectory：true，让第二次构建读取上一个的缓存，保证构建熟读更快。
5. 优化loader，使用include缩小loader的处理范围