样式处理

分离样式文件

webpack插件
Webpack4 之前 extract-text-webpack-plugin
Webpack4 之后 mini-css-extract-plugin

将style标签中的css提取到css文件中

extract-text-webpack-plugin

```
const ExtractTextPlugin = require('extract-text-webpack-plugin');
module.exports = {
    entry: './app.js',
    output: {
        filename: 'bundle.js',
    },
    mode: 'development',
    module: {
        rules: [
            {
                test: /\.css$/,
                use: ExtractTextPlugin.extract({
                    fallback: 'style-loader',
                    use: 'css-loader',
                }),
            }
        ],
    },
    plugins: [
        new ExtractTextPlugin("bundle.css")
    ],
};
```

多样式文件的处理

多入口，bundle.css就会重名
需要输出的css文件也用类似output.finename 一样动态命名

```
plugins:[
    new ExtractTextPlugin('[name].css')
]
```

[name] 指代chunkname

mini-css-extract-plugin

支持按需加载css
extract-text-webpack-plugin区别

- loader规则设置形式不同
  支持publicPath，指定异步css的加载路径
- 不需要设置fallback
- plugins除了指定css资源名（filename），加需要指定异步加载的css资源名（chunkFilename）

样式预处理

样式预编译语言
SCSS Less 等
打包将预编译语言转换为css，降低开发和维护成本

Sass 与 SCSS

Sass是对css语法的增强，loader一般都是sass-loader，后缀.scss

sass-loader就是将scss语法编译为css，还需要搭配css-loader与style-loader，还需要安装node-sass

真正编译SCSS的是node-sass，sass-loader只是黏合作用

想在浏览器调试工具查看源码需要为sass-loader和css-loader配置source map

Less

对CSS的拓展，与SCSS类似，越要安装loader与自身编译模块
less-loader
Less支持多种编译过程中的配置，可以直接在options中配置

PostCSS

并不能算是一个CSS预编译器，只是一个编译插件的容器
接收样式源码，交给编译软件，然后输出CSS

PostCSS 与Webpack

postcss-loader
单独使用postcss-loader不建议使用 @import语句，会产生冗余代码

建议把postcss-loader放在css-loader之后使用

单独的配置文件 postcss.config.js

自动前缀

可以与Autoprefixer结合，自动添加厂商前缀

stylelint

像eslint，统一代码风格，确保代码质量
需要再postcss.config.js中添加相应配置

CSSNext

postcss-cssnext
在postcss.config.js中添加相应配置

CSS Modules

最近流行的开发模式，将CSS模块化

- 每个css文件有自己的单独的作用域，不会命名冲突
- 对css进行依赖管理，通过相对路径引入css文件
- 通过composes轻松复用其他css模块

不需要额外安装模块，只需要开始css-loader中modules配置项

总结

通过SCSS Less等预编译样式语言来提高开发效率
通过PostCSS以及相应插件可以使用更新的CSS特性，保证浏览器兼容性
CSS 模块化，避免样式冲突