webpack工程化实战

暗号：可以做，但没必要

作业：自定义实现less-loader css-loader style-loader

要求：提供代码截图

项目准备

- 初始化

```
npm init -y  # 初始化npm配置文件
npm install --save-dev webpack # 安装核心库
npm install --save-dev webpack-cli # 安装命令行工具
```

- .npmrc

大家一开始使用 npm 安装依赖包时，肯定感受过那挤牙膏般的下载速度，上网一查只需要将 npm 源设置为淘宝镜像源就行，在控制台执行一下以下命令：

`npm config set registry https://registry.npm.taobao.org`

从此过上了速度七十迈，心情是自由自在的生活。

但是大家想想，万一某个同学克隆了你的项目之后，准备在他本地开发的时候，并没有设置淘宝镜像源，又要人家去手动设置一遍，我们作为项目的发起者，就先给别人省下这份时间吧，只需要在根目录添加一个 .npmrc 并做简单的配置即可：

```
# 创建 .npmrc 文件

touch .npmrc

# 在该文件内输入配置

registry=https://registry.npm.taobao.org/
```

- 创建src目录及入口文件
- 创建webpack配置文件，默认配置

```
# webpack.config.js
const path = require("path");
module.exports = {
  entry: "./src/index.js",
  output: {
    path: path.resolve(__dirname, "./dist"),
    filename: "[name].js",
  },
  mode: "development",
};
```

样式处理

- 集成css样式处理：css-loader style-loader
- 创建index.css

```
# 安装
npm install style-loader css-loader -D

# 配置
module: {
  rules: [
    {
      test: /\.css$/,
      use: ["style-loader", "css-loader"],
    },
  ],
},
```

- 集成less sass

```
# sass
npm install node-sass sass-loader -D

# less
npm install less less-loader -D

#配置
rules:[
  {
    test: /\.scss$/,
    use: ["style-loader","css-loader","sass-loader"]
  },
  {
    test: /\.less$/,
```

```
    use: ["style-loader","css-loader","less-loader""]
  }
]
```

- 集成postcss:

  Github:https://github.com/postcss/postcss

  相当于babel于JS

  postcss主要功能只有两个：第一就是把css解析成JS可以操作的抽象语法树AST，第二就是调用插件来处理AST并得到结果；所以postcss一般都是通过插件来处理css，并不会直接处理 比如：

  - 自动补齐浏览器前缀: autoprefixer
  - css压缩等 cssnano

```
npm install postcss-loader autoprefixer cssnano -D
```

```
# 创建postcss.config.js

# 配置postcss.config.js
module.exports = {
  plugins: [require("autoprefixer")],
};

# 配置package.json
"browserslist":["last 2 versions", "> 1%"],


# 或者直接在postcss.config.js里配置
module.exports = {
  plugins: [
    require("autoprefixer")({
      overrideBrowserslist: ["last 2 versions", "> 1%"],
    }),
  ],
};

# 或者创建.browserslistrc文件
> 1%
last 2 versions
not ie <= 8
```

- 样式文件分离

经过如上几个loader处理，css最终是打包在js中的，运行时会动态插入head中，但是我们一般在生产环境会把css文件分离出来（有利于用户端缓存、并行加载及减小js包的大小），这时候就用到 mini-css-extract-plugin 插件。

一般用于生产环境

```
# 安装
npm install mini-css-extract-plugin -D

# 使用
const MiniCssExtractPlugin = require('mini-css-extract-plugin');

module.exports = {
  module: {
    rules: [
      {
        test: /\.less$/,
        use: [
          // 插件需要参与模块解析，须在此设置此项，不再需要style-loader
          {
             loader: MiniCssExtractPlugin.loader,
             options: {
                hmr: true, // 模块热替换，仅需在开发环境开启
                // reloadAll: true,
                // ... 其他配置
             }
          },
           'css-loader',
           'postcss-loader',
           'less-loader'
        ],
      },
    ],
  },
  plugins: [
    new MiniCssExtractPlugin({
      filename: '[name].css', // 输出文件的名字
      // ... 其他配置
    }),
  ]
};
```

over!!