淘票票项目分析：

1.axios 请求数据

2.vuex 状态管理

cnpm install vuex axios --save-dev

3.移动端适配方案 --->rem

cnpm install lib-flexible --save-dev

4.css写less/sass语法预处理

sass有一个包node-sass很大，也很难下载

使用less:less less-loader

cnpm install less less-loader -D

axios拦截器：

①请求拦截

②响应拦截

拦截器运行过程：

拦截器

请求拦截器->

应用 后台服务器

<-响应拦截器

构建工具：webpack,gulp,grunt,fis

Gulp:

https://www.gulpjs.com.cn/

是前端开发过程中一种基于流的代码构建工具，是自动化项目的构建利器；是基于Nodejs的自动任务运行器，他能自动化地完成前端代码的测试、检查、合并、压缩、格式化、浏览器自动刷新、部署文件生成，并监听文件在改动后重复指定的这些步骤

优点：

1.易于使用

通过代码优于配置的策略，Gulp 让简单的任务简单，复杂的任务可管理。

2.构建快速

利用 Node.js 流的威力，你可以快速构建项目并减少频繁的 IO 操作。

3.插件高质

Gulp 严格的插件指南确保插件如你期望的那样简洁高质得工作。

4.易于学习

通过最少的 API，掌握 Gul

安装：

1.cnpm install --global gulp 全局安装

2.gulp -v 检查版本号

3.npm init -y

4.cnpm install gulp --save-dev

5.在站点文件夹根目录创建一个名为gulpfile.js的文件

gulp-api:

1.gulp.task(任务名称,数组,回调函数):任务

2.gulp.src():输入的计算机内存

3.gulp.dest():输出到文件

4.gulp.watch():监控文件变动

gulp-less:编译css

1.下载gulp-less插件：cnpm install gulp-less --save-dev

2.打开https://www.npmjs.com/

3.搜索gulp-less

gulp-concat:合并css

1.下载gulp-less插件：cnpm install gulp-concat --save-dev

2.打开https://www.npmjs.com/

3.搜索gulp-concat

gulp-rename:改名

1.下载gulp-less插件：cnpm install gulp-rename --save-dev

2.打开https://www.npmjs.com/

3.搜索gulp-rename

gulp-babel:编译js

1.下载gulp-babel插件：cnpm install gulp-babel @babel/core @babel/preset-env --save-dev

2.打开https://www.npmjs.com/

3.搜索gulp-babel

gulp-uglify:压缩js

1.下载gulp-less插件：cnpm install gulp-uglify --save-dev

2.打开https://www.npmjs.com/

3.搜索gulp-uglify

浏览器自动刷新：browsersync

1.下载browsersync插件：npm install -g browser-sync cnpm install browser-sync --save-dev

2.打开http://www.browsersync.cn/docs/ <http://www.browsersync.cn/docs/gulp/#gulp-install>

gulp-imagemin:

1.下载gulp-less插件：cnpm install gulp-imagemin --save-dev

2.打开https://www.npmjs.com/

3.搜索gulp-imagemin

Webpack:

WebPack可以看做是模块打包机：

它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

什么是构建：

构建就是把源代码转换成发布到线上的可执行 JavaScrip、CSS、HTML 代码，包括如下内容。

代码转换：TypeScript 编译成 JavaScript、SCSS 编译成 CSS 等。

文件优化：压缩 JavaScript、CSS、HTML 代码，压缩合并图片等。

代码分割：提取多个页面的公共代码、提取首屏不需要执行部分的代码让其异步加载。

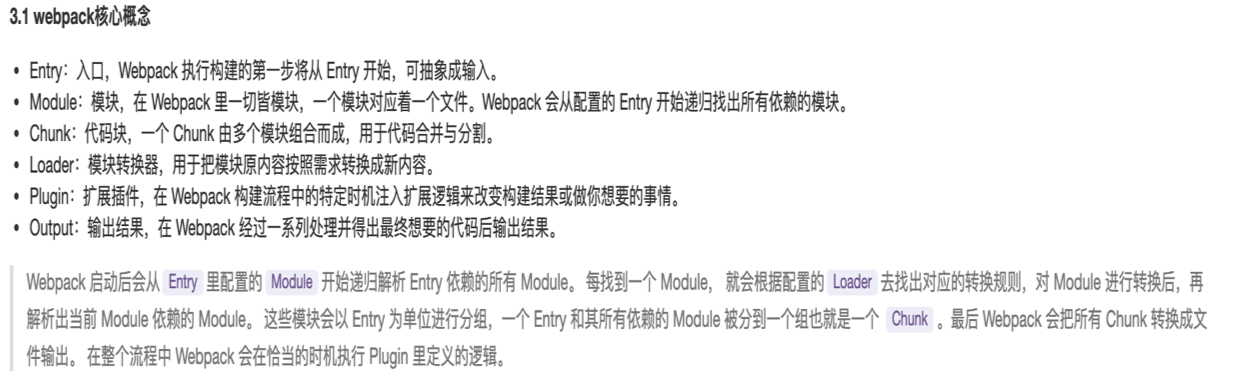
模块合并：在采用模块化的项目里会有很多个模块和文件，需要构建功能把模块分类合并成一个文件。

自动刷新：监听本地源代码的变化，自动重新构建、刷新浏览器。

代码校验：在代码被提交到仓库前需要校验代码是否符合规范，以及单元测试是否通过。

自动发布：更新完代码后，自动构建出线上发布代码并传输给发布系统。

构建其实是工程化、自动化思想在前端开发中的体现，把一系列流程用代码去实现，让代码自动化地执行这一系列复杂的流程。 构建给前端开发注入了更大的活力，解放了我们的生产力。



mkdir webpackProject

cd webpackProject

npm init -y

全局安装webpack：cnpm install webpack webpack-cli -g

配置webpack  cnpm install webpack webpack-cli –D

创建src目录 mkdir src

创建dist目录 mkdir dist

配置文件webpack.config.js

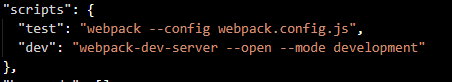


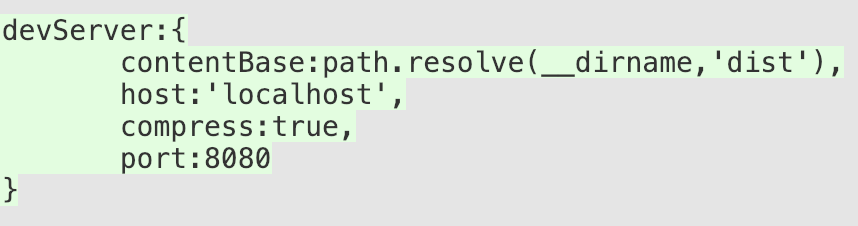
在dist目录下创建index.html文件



webpack --config webpack.config.js

cnpm install webpack-dev-server --save-dev

在package.json



npm run dev

cnpm install style-loader css-loader -D



在src下新建index.css body{background:pink}

在src下的index.js中引入import “./index.css”

删掉dist下的min.js

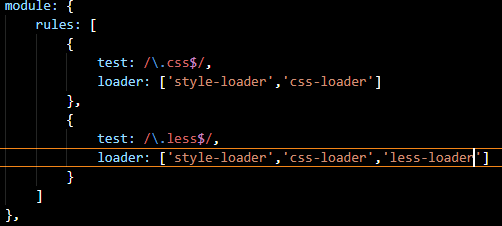
webpack --config webpack.config.js

npm run dev

cnpm install --save-dev less-loader less

在src下新建index.less body{background:green}

在src下的index.js中引入import “./index.less”

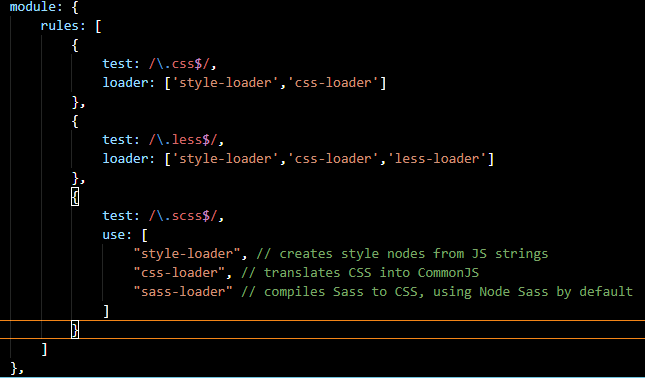


npm run dev

cnpm install sass-loader node-sass --save-dev

在src下新建index.scss body{background:red}

在src下的index.js中引入import “./index.scss”

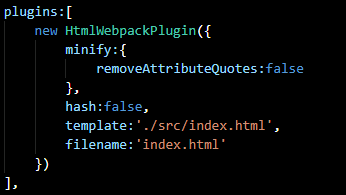


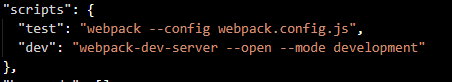
npm run dev

cnpm i --save-dev html-webpack-plugin

在src下新建index.html

在webpack.config.js中引入



在package.json

删除dist下的文件，执行webpack --config webpack.config.js(min.js，index.html在dist文件夹下，多了script标签)

file-loader 解决CSS等文件中的引入图片路径问题

url-loader 当图片小于limit的时候会把图片BASE64编码，大于limit参数的时候还是使用file-loader 进行拷贝npm run test转成base64

limit小于8KB时，将图片转换为base64格式输出

limit大于8KB时，则url-loader会调用file-loader进行处理并输出图片

cnpm i file-loader url-loader -D



在src下放入suo.png

方法一：

在index.js中引入import ‘index.less’;

import logo from "./suo.png";

let img = new Image();

img.src = logo;

document.body.appendChild(img);

npm run dev

方法二：

在index.less中：body{background-color: aquamarine;background-image: url(./suo.png);}

npm run dev

Babel:

Babel其实是一个编译JavaScript的平台,可以把ES6/ES7,React的JSX转义为ES5

cnpm install --save-dev babel-loader babel-core

在index.js中let a = 1;

let b = ()=>{

console.log(a);

}

b();

module: {

rules: [

{ test: /\.js$/, exclude: /node\_modules/, loader: "babel-loader" }

]}

webpack --config webpack.config.js

如果报错Cannot find module '@babel/core'

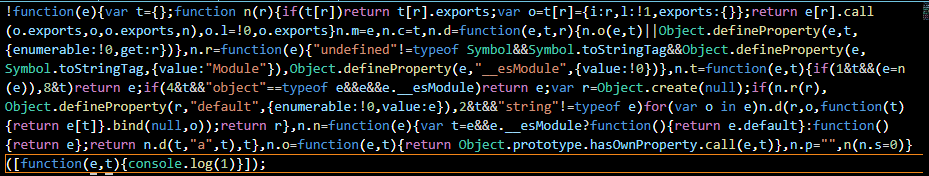
babel-loader@8 requires Babel 7.x (the package '@babel/core'). If you'd like to use Babel 6.x ('babel-core'), you should install 'babel-loader@7'.

下载cnpm install -D babel-loader @babel/core @babel/preset-env webpack或者

cnpm i babel-loader @babel/core @babel/preset-env @babel/preset-react –D

cnpm i @babel/plugin-proposal-decorators @babel/plugin-proposal-class-properties -D

webpack --config webpack.config.js



cnpm install --save-dev @babel/plugin-transform-runtime

常用的插件列表：

文件

raw-loader 加载文件原始内容（utf-8）

val-loader 将代码作为模块执行，并将 exports 转为 JS 代码

url-loader 像 file loader 一样工作，但如果文件小于限制，可以返回 data URL

file-loader 将文件发送到输出文件夹，并返回（相对）URL

JSON

json-loader 加载 JSON 文件（默认包含）

json5-loader 加载和转译 JSON 5 文件

cson-loader 加载和转译 CSON 文件

转换编译

script-loader 在全局上下文中执行一次 JavaScript 文件（如在 script 标签），不需要解析

babel-loader 加载 ES2015+ 代码，然后使用 Babel 转译为 ES5

ts-loader 或 awesome-typescript-loader 像 JavaScript 一样加载 TypeScript 2.0+

coffee-loader 像 JavaScript 一样加载 CoffeeScript

模板

html-loader 导出 HTML 为字符串，需要引用静态资源

pug-loader 加载 Pug 模板并返回一个函数

jade-loader 加载 Jade 模板并返回一个函数

markdown-loader 将 Markdown 转译为 HTML

react-markdown-loader 使用 markdown-parse parser(解析器) 将 Markdown 编译为 React 组件

posthtml-loader 使用 PostHTML 加载并转换 HTML 文件

handlebars-loader 将 Handlebars 转移为 HTML

markup-inline-loader 将内联的 SVG/MathML 文件转换为 HTML。在应用于图标字体，或将 CSS 动画应用于 SVG 时非常有用

样式

style-loader 将模块的导出作为样式添加到 DOM 中

css-loader 解析 CSS 文件后，使用 import 加载，并且返回 CSS 代码

less-loader 加载和转译 LESS 文件

sass-loader 加载和转译 SASS/SCSS 文件

postcss-loader 使用 PostCSS 加载和转译 CSS/SSS 文件

stylus-loader 加载和转译 Stylus 文件

清理和测试

Linting && Testing

mocha-loader 使用 mocha 测试（浏览器/NodeJS）

eslint-loader PreLoader，使用 ESLint 清理代码

jshint-loader PreLoader，使用 JSHint 清理代码

jscs-loader PreLoader，使用 JSCS 检查代码样式

coverjs-loader PreLoader，使用 CoverJS 确定测试覆盖率

框架

vue-loader 加载和转译 Vue 组件

polymer-loader 使用选择预处理器(preprocessor)处理，并且 require() 类似一等模块(first-class)的 Web 组件

angular2-template-loader 加载和转译 Angular 组件

ES6：

let特性：

①无变量提升

②同一个作用域内，禁止重复定义

③绑定块级作用域

const特性：

①声明一个只读的常量。一旦声明，常量的值就不能改变。

undefined：声明变量未赋值，defined：未定义

解构赋值：从数组和对象中提取值，按照对应位置，对变量赋值(解构成功)

Let [a,b,c] = [2,3]解构不成功

Let [a,b] = [2,3,5]不完全解构

Let [foo=0] = [2] console.log(foo)2默认值

Set:

属性：①Set.prototype.constructor：构造函数，默认就是Set函数。

②Set.prototype.size：返回Set实例的成员总数。

方法：①add(value)：添加某个值，返回 Set 结构本身。

②delete(value)：删除某个值，返回一个布尔值，表示删除是否成功。

③has(value)：返回一个布尔值，表示该值是否为Set的成员。

④clear()：清除所有成员，没有返回值。

Map:

属性：①size属性返回 Map 结构的成员总数。

方法：①set(key, value)方法设置键名key对应的键值为value，然后返回整个 Map 结构。如果key已经有值，则键值会被更新，否则就新生成该键。

set方法返回的是当前的Map对象，因此可以采用链式写法。

②get(key)方法读取key对应的键值，如果找不到key，返回undefined。

③has(key)方法返回一个布尔值，表示某个键是否在当前 Map 对象之中。

④delete(key)方法删除某个键，返回true。如果删除失败，返回false。

⑤clear()方法清除所有成员，没有返回值。

Promise：

Promise对象有两个特点：①对象的状态不受外界影响。②一旦状态改变，就不会再变，任何时候都可以得到这个结果。

Promise有三种状态：pending（进行中）、fulfilled（已成功）和rejected（已失败）。

Promise缺点：Promise也有一些缺点。首先，无法取消Promise，一旦新建它就会立即执行，无法中途取消。其次，如果不设置回调函数，Promise内部抛出的错误，不会反应到外部。第三，当处于pending状态时，无法得知目前进展到哪一个阶段（刚刚开始还是即将完成）。

this的优先级：

new(新创建的对象)>显式绑定(指定的对象)>隐式绑定(属于的对象)>默认绑定

Class:

new的过程：①创建一个空对象

②执行constructor

③把值放进第一步的空对象中

④返回空对象

面向对象：

特点：封装，继承，多态，抽象