Webpack4.30.0笔记

1. 配置nodejs环境

在nodejs下新建2个文件夹node\_global和node\_cache

然后执行npm config set prefix "D:\Program Files(x86)\nodejs\node\_global"和npm config set cache "D:\Program Files(x86)\nodejs\node\_cache"

二、webpack项目

cmd中：先安装webpack到当前项目

1. 转到项目文件根目录，然后执行npm install webpack –save-dev，然后执行npm install -D webpack-cli安装webpack-cli。
2. 安装jquery之前，先执行npm init -y，然后执行npm i jquery -S
3. 将src中的main,js打包到dist中的bundle.js中，执行 webpack .\src\main.js --output .\dist\bundle.js --mode development
4. 引入bundle.js时，路径要写对，写成类似这样：<script src="http://localhost:8080/bundle.js"></script>
5. 设置webpack-dev-server的参数：

（1）方法1：在package.json文件中设置

//--open自动打开，--port设置端口号 --contentBase 根目录 –hot热更新

"dev":"webpack-dev-server --open --port 3000 --contentBase src --hot"

或者这样设置在

（2）方法2：在webpack.config.js文件中设置，增加对象:

devServer: {

contentBase: 'src', //指定托管的目录

open:true, //自动打开浏览器

compress: true,

port: 3000, //设置端口号

hot: true //启用热更新，第1步

},

plugins:[ //配置插件的节点

new webpack.HotModuleReplacementPlugin() //new一个热更新的模块对象，启动热更新第3步

],

行首初始化webpack

//启用热更新，第2步

const webpack = require('webpack');

1. 将bundle.js放在内存中执行（原因：执行速度快），先执行npm i html-webpack-plugin -D,

然后在webpack.config.js文件中初始化（只要是插件，都要放到plugins节点中）

const htmlWebpackPlugin = require('html-webpack-plugin'); 然后plugins节点中new htmlWebpackPlugin({ //创建一个在内存中生成的HTML页面插件

template:path.join(\_\_dirname,'./src/index.html'),//指定模板页面，将来会根据指定的路径，生成内存中的页面

filename:"index.html"

})

1. 处理css文件

在main.js中导入文件 import './css/index.css';由于默认只能处理js类型的文件，无法处理其他非js类型的文件，所有需要安装第三方 loader 加载器。

（1）如果想要打包处理css文件，需要安装npm i style-loader css-loader -D

（2）在webpack.config.js中新增一个配置节点:module对象，有个rules属性（数组，存放所有第三方文件的匹配和处理的规则）

module:{

rules:[ //所有第三方文件的匹配和处理的规则

{test:/\.css$/,use:['style-loader','css-loader'],} //从右至左调用

]

},

1. 处理less文件
2. 在main.js中导入文件 import './css/index.css';
3. 需要安装npm i less-loader -D和npm i less -D
4. webpack.config.js中module新增规则：{test:/\.less$/,use:['style-loader','css-loader','less-loader']},//配置处理.less文件
5. 处理scss文件
6. 在main.js中导入文件 import './css/index.scss';
7. 需要安装npm sass-loader -D和npm i node-sass -D
8. webpack.config.js中module新增规则：{test:/\.less$/,use:['style-loader','css-loader','scss-loader']},//配置处理scss文件
9. 处理url地址

默认情况下，webpack无法处理css中的url地址，不管是图片还是字体

1. /css/index.scss中写有背景图的样式
2. 需要安装npm i url-loader file-loader -D和npm i node-sass -D
3. webpack.config.js中module新增规则：{test:/\.(jpg|png|gif|hmp|jpeg)$/,use:'url-loader'}//配置处理图片文件
4. 如果要传参，方式同url传参方式，{test:/\.(jpg|png|gif|hmp|jpeg)$/,use:'url-loader?limit=118297&name=[hash:8]-[name].[ext]'}

//limit值给定的值，是图片的大小，单位是byte，如果引用的图片，大于或等于给定的值，则不会转为base64格式的字符串，否则转为base64字符串

//name显示原名

1. 引入bootstrap的字体图标
2. 需要安装npm i bootstrap@3.3.7 -S
3. 需要确保bootstrap文件中有fonts文件
4. webpack.config.js中module新增规则：{test:/\.(ttf|eot|svg|woff|woff2)$/,use:'url-loader'} //处理字体文件
5. webpack中babel的配置

babel可以将高级的语法(例如ES6)转换为低级的语法

1. 需要安装npm i babel-core babel-loader@7 babel-plugin-transform-runtime -D和

npm i babel-preset-env babel-preset-stage-0 -D。其中babel-preset-env是最新的es语法

(2) webpack.config.js中module新增规则：{test:/\.js$/,use:'babel-loader',exclude:/node\_modules/}//处理js文件，其中必须用exclude排除node\_modules目录（2个原因：不排除的话，会将该目录下所有js文件都打包编译，非常消耗cpu，同时打包速度非常慢 ； 哪怕最终把所有的js文件转换完毕，项目也运行不了）

（3）在项目的根目录中，新建一个叫做.babelrc的json格式配置文件，

{

"presets":["env","stage-0"],

"plugins":["transform-runtime"]

}