### 使用 webpack-dev-server

webpack-dev-server 为你提供了一个简单的 web 服务器，并且能够实时重新加载

**安装包**

npm install --save-dev webpack-dev-server

****修改webpack.config.js****

const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {

entry: {

app: './src/index.js',

print: './src/print.js'

},

devtool: 'inline-source-map',

+ devServer: {

+ contentBase: './build',

+ port:9000, //把默认端口号8080修改成9000

+ },

plugins: [

],

output: {

filename: '[name].bundle.js',

path: path.resolve(\_\_dirname, 'dist')

}

};

以上配置告知 webpack-dev-server，在 localhost:9000 下建立服务器，将 build 目录下的文件，作为可访问文件。

让我们添加一个 script 脚本，可以直接运行开发服务器(dev server)：

**package.json**

{

"name": "development",

"version": "1.0.0",

"description": "",

"main": "webpack.config.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"watch": "webpack --watch",

+ "start": "webpack-dev-server --open",

"build": "webpack"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"clean-webpack-plugin": "^0.1.16",

"css-loader": "^0.28.4",

"csv-loader": "^2.1.1",

"file-loader": "^0.11.2",

"html-webpack-plugin": "^2.29.0",

"style-loader": "^0.18.2",

"webpack": "^3.0.0",

"xml-loader": "^1.2.1"

}

}

现在，我们可以在命令行中运行 npm start，就会看到浏览器自动加载页面。如果现在修改和保存任意源文件，web 服务器就会自动重新加载编译后的代码。试一下！

**模拟接口数据**

webpack.config.js

// 将AnalogData文件夹的json文件require到data中

const datas = require('require-all')({

    dirname: \_\_dirname + '/AnalogData',

})

console.log(datas);

module.exports = {

    devtool: 'inline-source-map',

    mode: "development",

    devServer: {

        contentBase: './build',

        port: 9000,

        before: function (app) {

            // 处理/api开头的post请求

            app.post(/^\/api/i, function (req, res) {

                console.log(req.url)

                // 匹配url

                let regex = /\/api\/([^\/]\*)\/(.\*)/i

                let match = regex.exec(req.url)

                console.log(match);

                if (match == null || match.isSuccess == false) {

                    res.json({

                        isSuccess: true,

                        value: {

                        }

                    });

                    return;

                }

                // /api/controller/action 匹配controller

                let file = match[1]

                // /api/controller/action 匹配action

                let name = match[2]

                res.json({

                    isSuccess: true,

                    value: (datas[file] || {})[name]

                });

            });

        },

        historyApiFallback: {

            //使用正则匹配命中路由

            rewrites: [

                { from: /^\//, to: '/Index.html' },

            ]

        }

    }

}

**基于代理的WebpackServer**

引用'http-proxy-middleware'模块

const serverProxy = require('http-proxy-middleware')

配置devServer的before字段

before: function (app) {

    app.use(

        serverProxy(

            '/api/', {

                target: 'http://localhost:5000/',

                changeOrigin: true

            }

        )

    )

},

该功能将/api/开头的请求转发自http://localhost:5000/