1. 初始化项目

npm init -y

1. 安装webpack和webpack-cli

npm i webpack --save-dev

npm i webpack-cli --save-dev

1. 修改package.json文件

"scripts": {

"build": "webpack"//执行运行项目

}

1. 创建入口文件(webpack4 more配置entry（入口src/index.js 和output（出口） dist/main.js)

/src/index.js

1. 修改package.json文件

"dev": "webpack --mode development",

"build": "webpack --mode production"

1. Webpack4配置总结



1. 项目配置
   1. js的处理：转换 ES6 代码，解决浏览器兼容问题(配置babel-loader)

安装：

npm install babel-loader@8.0.0-beta.0 @babel/core @babel/preset-env

npm install --save @babel/polyfill

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

npm install --save @babel/runtime

添加babelloader配置文件/.babelrc

* 1. css的处理：编译css，自动添加前缀，抽取css到独立文件(编译css，自动添加前缀，抽取css到独立文件)

npm i mini-css-extract-plugin css-loader --save-dev(postcss-loader(抽取css到独立文件)

npm i style-loader postcss-loader --save-dev (用于添加浏览器前缀)

npm install --save-dev autoprefixer

创建postcss配置文件/postcss.config.js

* 1. html的处理：复制并压缩html文件（复制并压缩html文件 html-webpack-plugin）

npm i html-webpack-plugin html-loader --save-dev

* 1. dist的清理：打包前清理源目录文件（clean-webpack-plugin）

npm install clean-webpack-plugin --save-dev

* 1. assets的处理：静态资源处理（file-loader）

npm install file-loader --save-dev

* 1. server的启用：development 模式下启动服务器并实时刷新（webpack-dev-server）

npm i webpack-dev-server --save-dev

7.7添加项目配置文件/webpack.config.js

const path = require('path');

const isProd = process.env.NODE\_ENV === 'production';

const MiniCssExtractPlugin = require("mini-css-extract-plugin");

const WebpackChain = require('webpack-chain');

const config = new WebpackChain();

config.when(isProd,config=>{

    config.entry('index').add('./src/index.js');

}).when(!isProd,config=>{

    config.entry('index').add('./src/index.js');

})

// Modify output settings

    .output

    .path(path.join(\_\_dirname, "dist")).filename('[name].js').end()

    .when(isProd, config => {

        config.mode('production');

    }).when(!isProd,config=>{

    config.mode('development').devtool('source-map');

}).end();

/\*\*

 \* module

 \*/

config

    .module

    .rule("compile")

    .test(/\.js$/)

    .include.add(path.join(\_\_dirname,'src')).end()

    .exclude.add(/node\_modules/).end()

    .use('babel').loader("babel-loader")

    .options({

        presets: ['@babel/preset-env'],

        plugins: ['@babel/plugin-proposal-class-properties']

    });

config.module

    .rule('images')

    .test(/\.(png|jpg|jpeg|gif)/)

    .use('url-loader')

    .loader('url-loader')

    .options({

        limit: 1 \* 1024,

        name: path.posix.join("images","[name].[ext]")

    })

// do not base64-inline SVGs.

// https://github.com/facebookincubator/create-react-app/pull/1180

config.module

    .rule('svg')

    .test(/\.(svg)(\?.\*)?$/)

    .use('url-loader')

    .loader('url-loader')

    .options({

        limit: 1024 \* 3,//30kb

        fallback: 'file-loader'

    })

config.module

    .rule("fonts")

    .test(/\.(woff2?|eot|ttf|otf)(\?.\*)?$/i)

    .use('url-loader')

    .loader('url-loader')

    .options({

        limit: 10000,

        fallback: {

            loader: 'file-loader',

            options: {

                name: path.posix.join("fonts","[name].[ext]")

            }

        }

    });

config.when(isProd,config=>{

    config.module.rule("css").test(/\.(sa|sc|c)ss$/)

        .use("style").loader(MiniCssExtractPlugin.loader);

}).when(!isProd,config=>{

    config.module.rule("css").test(/\.(sa|sc|c)ss$/)

        .use("style-loader").loader("style-loader");

});

config.module.rule("css").test(/\.(sa|sc|c)ss$/)

    .use('css').loader("css-loader").end()

    .use('postcss-loader').loader('postcss-loader');

config.module.rule("scss").test(/\.(sa|sc)ss$/).use("sass-loader").loader("sass-loader");

config.module.rule("lass").test(/\.less$/).use("less-loader").loader("less-loader");

//config.module.rule("html").test(/\.(htm|html)$/i).use("html").loader('html-withimg-loader');

/\*\*

 \* plugin

 \*/

config.when(isProd,config=>{

    const UglifyJSPlugin = require('uglifyjs-webpack-plugin');

    const CleanWebpackPlugin = require('clean-webpack-plugin');

    const CopyWebpackPlugin = require("copy-webpack-plugin");

    config.plugin("clear").use(new CleanWebpackPlugin([path.join(\_\_dirname, 'dist')]));

    config.optimization.splitChunks({

        cacheGroups: {

            commons: {

                chunks: "initial",

                name: "common",

                minChunks: 2,

                maxInitialRequests: 5, // The default limit is too small to showcase the effect

                minSize: 0, // This is example is too small to create commons chunks

                reuseExistingChunk: true // 可设置是否重用该chunk（查看源码没有发现默认值）

            }

        }

    });

    config.plugin("js").use(new UglifyJSPlugin({}));

    config.plugin('extract-css')

        .use(MiniCssExtractPlugin, [{

            filename: "css/[name].css",

            chunkFilename: "css/[name].css"

        }]);

    // config.plugin('copy').use(new CopyWebpackPlugin([

    //     {

    //         from:"./src/sass",

    //     }

    // ]))

})

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

config.plugin("html").use(HtmlWebpackPlugin, [{

    /\*

     template 参数指定入口 html 文件路径，插件会把这个文件交给 webpack 去编译，

     webpack 按照正常流程，找到 loaders 中 test 条件匹配的 loader 来编译，那么这里 html-loader 就是匹配的 loader

     html-loader 编译后产生的字符串，会由 html-webpack-plugin 储存为 html 文件到输出目录，默认文件名为 index.html

     可以通过 filename 参数指定输出的文件名

     html-webpack-plugin 也可以不指定 template 参数，它会使用默认的 html 模板。

     \*/

    template: "./public/index.html",

    filename:"index.html",

    /\*

     因为和 webpack 4 的兼容性问题，chunksSortMode 参数需要设置为 none

     https://github.com/jantimon/html-webpack-plugin/issues/870

     \*/

    chunksSortMode: 'none',

    xhtml: true,

    minify: {

        collapseWhitespace: false, //删除空格，但是不会删除SCRIPT、style和textarea中的空格

        conservativeCollapse: false, //删除空格，总是保留一个空格

        removeAttributeQuotes: false, //删除引号，删除不需要引号的值

        useShortDoctype: false, //使用短的文档类型

        removeComments: true,

        collapseBooleanAttributes: true,

        removeScriptTypeAttributes: true

        // more options:

        // https://github.com/kangax/html-minifier#options-quick-reference

    }

}]);

config.when(isProd,config=>{

}).when(!isProd,config=>{

    config.devServer.host('localhost').port(8080).open(process.os === 'darwin');

})

config.resolve.alias.set("@",path.join(\_\_dirname,"src"));

// Export the completed configuration object to be consumed by webpack

module.exports = config.toConfig();

其他配置请查看

Webpack.config.all.js

package.json中配置

"scripts": {

"start": "webpack-dev-server --mode development --open",

"build": "webpack --mode production"

}