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",

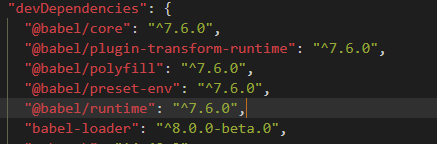
"build": "webpack --mode production"

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

{

  "presets": [

      "@babel/preset-env"

  ],

  "plugins": ["@babel/plugin-transform-runtime"]

}

添加项目配置文件/webpack.config.js

const path = require('path');

module.exports = {

  entry: [

    "@babel/polyfill",

    path.join(\_\_dirname, './src/index.js')

  ],

  module: {

    rules: [

      {

        test: /\.js$/,

        exclude: /node\_modules/,

        loader: 'babel-loader',

        options: {}

      }

    ]

  }

}

* 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

module.exports = {

  plugins: {

      autoprefixer: {}

  }

}

修改webpack.config.js文件

const path = require('path');

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

module.exports = (env, argv) => {

  const devMode = argv.mode !== 'production'

  return {

    module: {

      entry: [

        "@babel/polyfill",

        path.join(\_\_dirname, './src/index.js')

      ],

      rules: [

        {

          test: /\.js$/,

          exclude: /node\_modules/,

          loader: 'babel-loader',

          options: {}

        },

        {

          test: /\.css$/,

          use: [

            devMode ? 'style-loader' : MiniCssExtractPlugin.loader,

            'css-loader',

            'postcss-loader'

          ]

        }

      ],

    },

    plugins: [

      new MiniCssExtractPlugin({

        filename: "[name].css",

        chunkFilename: "[id].css"

      })

    ]

  }

}

* 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

Webpack.congfig.json中配置

const HtmlWebPackPlugin = require("html-webpack-plugin");

module.exports = {

module: {

rules: [

// ...,

{

test: /\.html$/,

use: [{

loader: "html-loader",

options: {

minimize: true

}

}]

}

]

},

plugins: [

new HtmlWebPackPlugin({

template: "./src/index.html",

filename: "./index.html"

})

]

};

* 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

配置：

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

module.exports = {

plugins: [

new CleanWebpackPlugin(['dist']),

]

};

* 1. assets的处理：静态资源处理（file-loader）

npm install file-loader --save-dev

配置

* 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

package.json中配置

"scripts": {

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

"build": "webpack --mode production"

}