1、创建一个webpack.prod.config.js 文件，把开发环境webpack.config.js 的配置copy过来

2、在package.json文件的scripts对象中增加打包命令配置

"build": "webpack --config webpack.prod.config.js"

3、修改配置信息，吧打包的图片放进一个文件夹中

{ test: /\.(jpg|png|gif|bmp|jpeg)$/, use: ['url-loader?limit=1000&name=images/[hash:8]-[name].[ext]'] },

4、每次打包前删除之前的dist，防止文件重复。

需要安装一个 插件 npm i clean-webpack-plugin --dev

在webpack.prod.config.js中做如下配置

//导入每次打包时删除文件夹的插件

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

在plugins对象中配置如下

//创建删除文件插件，删除dist文件

new cleanWebpackPlugin(),

5、把第三方引入包单独放到一个js文件中。

//抽离第三方包的配置

optimization: {

//给固定的hash值，使webpack不打包未修改的文件，只打包已经修改的文件

moduleIds: 'hashed',

//为所有第三方包单独创建一个运行包，吧所有第三方包分开

runtimeChunk: 'single',

splitChunks: {

chunks: 'all',

cacheGroups: {

vendor: {

test: /[\\/]node\_modules[\\/]/,

name: 'vendors',

chunks: 'all'

}

}

},

6、配置js代码压缩。

安装 uglifyjs-webpack-plugin 插件

//压缩js代码插件

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

//压缩js代码

minimizer: [new UglifyJsPlugin({

parallel: true,//并行

cache: true,

uglifyOptions: {

output: {

comments: false,//注释不参与打包

},

},

})],

},

7、由于生产环境的配置是从开发环境复制过来的。所以需要删除一些开发环境的配置

如：

devServer

new webpack.HotModuleReplacementPlugin(),//一个热更新的模块对象,启动热更新的第三步

8、压缩html代码

在plugins 下 new htmlWebpackPlugin中传入如下参数

minify: { //压缩html代码

collapseWhitespace: true, //合并多余空格

removeComments: true, //移除注释

removeAttributeQuotes: true, //移除属性上的双引号

}

9、打包css文件

安装extract-text-webpack-plugin插件

//打包css文件插件

const ExtractTextPlugin = require("extract-text-webpack-plugin");

在匹配规则rules数组中配置如下，

{

test: /\.css$/,//配置处理css文件的第三方loader规则

use: ExtractTextPlugin.extract({

fallback: 'style-loader',

use:['css-loader'],

publicPath:'../'

})

},

{

test: /\.less$/, //配置处理sass文件的第三方loader规则

use: ExtractTextPlugin.extract({

fallback: 'style-loader',

use:['css-loader', 'less-loader'],

publicPath:'../'

}),

},

{

test: /\.scss$/, //配置处理sass文件的第三方loader规则

use: ExtractTextPlugin.extract({

fallback: 'style-loader',

use:['css-loader', 'sass-loader'],

publicPath:'../'

}),

},

删除之前的配置

{ test: /\.css$/, use: ['style-loader', 'css-loader'] }, //配置处理css文件的第三方loader规则

{ test: /\.less$/, use: ['style-loader', 'css-loader', 'less-loader'] }, //配置处理less文件的第三方loader规则

{ test: /\.scss$/, use: ['style-loader', 'css-loader', 'sass-loader'] }, //配置处理sass文件的第三方loader规则

10、压缩css

安装 optimize-css-assets-webpack-plugin 插件

//压缩css插件

var OptimizeCssAssetsPlugin = require('optimize-css-assets-webpack-plugin');

在生产环境的plugins数组中添加

new OptimizeCssAssetsPlugin()