**Babe部分**

1. 安装node;

初始化package.json;

npm init –y (-y就是都选择默认设置)

1. 安装babel;

npm i @babel/core @babel/cli @babel/preset-env

npm i @babel/polyfill 兼容更低版本的浏览器（ie6,ie5）

1. 添加脚本

“build”: “babel src –d dest” //在package.json文件的script中添加build命令，指示babel从src中读取文件，写入（-d）到dest文件夹下。

1. 添加配置文件.babelrc

{

    "presets": [

        "@babel/preset-env"

    ]

}

1. 执行命令

npm run build

**异步操作部分**

下载http-server启动服务；

Promise 异步编程解决方案；

try-catch async await 编程

**node部分**

nodejs的用处简介.png

**webpack部分**

1. 认识webpack

常用功能：压缩（把大的文件压缩成小的文件）、打包（把不同文件夹的打包到一个文件中）、多文件的编译（编译浏览器不识别的文件为浏览器识别的语言）、搭脚手架

安装：npm i –g webpack-cli

webpack.config.js:

mode:

none 不优化

development 输出调试信息，设置process.env.NODE\_EVN环境变量

production 最高优化，启用压缩、忽略错误

entry:入口

单入口---SPA

多入口—MPA

output:

{

path: 路径---必须是绝对路径 path.resolve

filename:文件名

}

单入口和多入口打包的配置：

const path = require('path');

module.exports = {       //单页面

    mode: 'development',

    entry: './src/index.js',

    output: {

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

        filename: 'bundle.min.js'

    }

}

module.exports = {      //多页面

    mode: 'development',

    entry: {    //是一个JSON,它下面的模块名字可以随意命名

        index: './src/index.js',

        news: './src/news.js'

    },

    output: {

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

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

    }

}

多页面中的filename中的[name]就是指entry中的index和news。

1. style-loader（输出字符串<style>）和css-loader（读取css，输出js字符串）

const path = require('path');

module.exports = {

    mode: 'development',

entry: './src/js/index.js',

output: {

  path: path.resolve(\_\_dirname, './dest'),

  filename: 'bundle.js'

   },

  module: {

    rules: [

          { test: /\.css$/i, use: ['style-loader', 'css-loader'] }

       ]

    }

}

1. postcss-loader和autoprefixer

可以单独建一个postcss.config.js，内容如下：

module.exports = {

plugins: [require(‘autoprefixer’)]

}

也可以放在webpack.config.js中，如下：

const autoprefixer = require('autoprefixer');

module: {

        rules: [

            {test: /\.css$/i, use:[

                'style-loader',

                'css-loader',

                {

                    loader: 'postcss-loader',

                    options: {

                        plugins: [autoprefixer]

                    }

                }]

            }

        ]

    }

npx autoprefixer -–info 查看npm中autoprefixer的支持情况。

新建一个 .browserslistrc 文件来定义autoprefixer的支持情况，如下：

last 5 version

> 1%

也可以配置在package.json下，如下：

"broswerslist": [

    "last 5 version",

    "> 1%"

  ]

1. file-loader和url-loader

module: {

        rules: [

            //css

            {

                test: /\.css$/i,

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

            },

            // 图片

            // { test: /\.(jpg|jpeg|gif|png)$/i, use: {

            //     loader: 'file-loader',

            //      options: { outputPath: 'imgs/',  //相对于output.path

            //             publicPath: 'dest/imgs/',    //输出到css的路径

            //             limit: 4\*1024 }

            //     }

            // },

            {// url-loader内部是调用了 file-loader的，所以用url-loader的时候要安装 file-loader

                test: /\.(jpg|jpeg|gif|png)$/i, use: {

                    loader: 'url-loader',

                    options: { outputPath: 'imgs/',  //相对于output.path

                        publicPath: 'dest/imgs/',    //输出到css的路径

                        limit: 10\*1024

                    }

                }

            },

            // 字体文件

            { test: /\.(eot|svg|ttf|woff|woff2)/i, use: {

                loader: 'url-loader',

                options: { outputPath: 'fonts/',

                    publicPath: 'dest/fonts',

                    limit: 4\*1024 }

                }

            }

        ]

    }

1. less-lader

module: {

        rules: [

            { test: /\.less$/i, use: ['style-loader', 'css-loader', 'less-loader']},

            { test: /\.(js|jsx)$/i, exclude: /node\_modules/, use: {

                loader: 'babel-loader',

                options: {

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

                }

            }}

        ]

    },

    devtool: 'source-map'   //调试工具

注意除了上面三个loader，还要装 less才能用。

1. webpack 核心

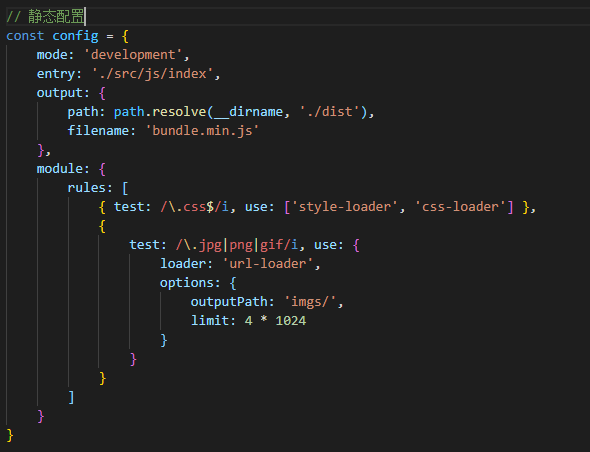
webpack-cli 命令行

webpack-dev-server dev-server核心

1. 操作html的plugin html-webpack-plugin

webpack.config.js

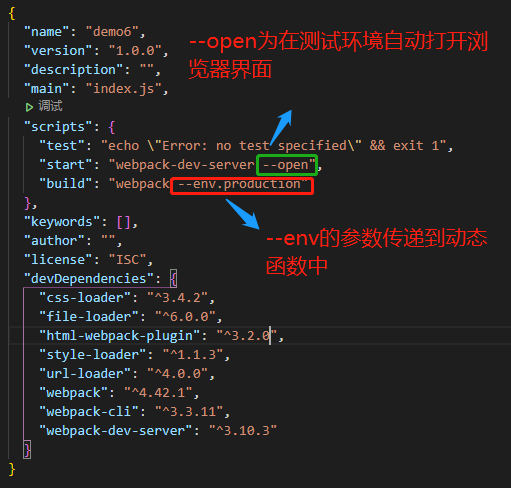
静态配置（原来的）



动态配置（新的）



动态配置根据env的参数来判断使用的动态配置项，env为执行命令时传的参数，如下：

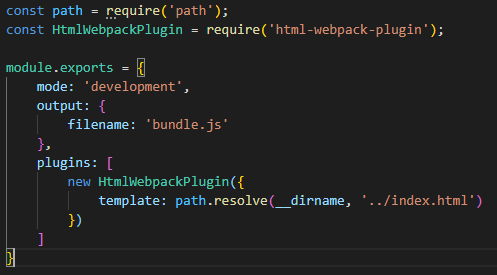


Html中可以放一些占位的模板，如下：



开发环境的配置：

开发环境是打包存在内存中的，不需要输出实际的文件，只需要文件名，所以output中没有path选项。



生产环境配置：

需要输出实际的内容，\_\_dirname加上../build实际的结果就是当前目录的上一级目录。因为使用实际是在webpack.config.js文件中，当webpack.config.js引用该生产配置时，正好指向的是webpack.config.js文件的位置。

