安装

npm install webpack-cli -g

npm install -g webpack webpack-cli

查看

webpack -v

webpack4新功能mode =》 开发/生产 环境

安装/卸载

npm install => npm i // npm i jquery

npm uninstall => npm un

安装多个空格隔开

例如：npm i jquery,react-redux,vue-router

初始化

package.json 项目主要依赖配置文件

npm init 或 npm init -y（快捷）

开发环境：

npm i jquery --save-dev

npm i jquery -D

生产环境

npm i jquery --save

npm i jquery -S

命令

查看cat package.json

清楚命令 clear

目录 ls

展示依赖项 ls node\_modules

淘宝镜像cnpm如何配置

http://npm.taobao.org/

$ npm install -g cnpm --registry=https://registry.npm.taobao.org

区别

cnpm i jquery 不会放在package.json

npm i jquery -D 放在devDependencies 开发环境 dev-开发

npm i jquery -S 放在dependencies

打包到index

webpack src/index.js --output dist/bundle.js // src下的index.js文件 打包到dist目录下的bundle.js。

Index.js 入口 bundle.js 出口

// webpack 配置文件 Webpack.config.js

直接运行webpack,生成出口文件bundle.js

const path = require('path') // node自带,path模块引入，路径问题

// 导出模块

module.exports = {

// 入口配置

entry:{

entry:'./src/index.js'// 入口文件，命名随意

},

// 出口配置

output:{

//path:\_\_dirname+'/dist',// Node提供的\_\_dirname为绝对路径，

path:path.resolve(\_\_dirname,'dist'), //第二种写法，绝对路径合并,需要require('path')

filename:'bundle.js'// 出口文件,不需要写文件前缀

},

// module 模块， 配置rules规则 loaders

module:{

},

// 插件

plugins:[],

// 开发服务器

devServer:{}

}

单独给webpack.config.js 取名

例如：xxx.config.js

生成打包文件，运行的时候需要输入 webpack --config xxx.config.js

Webpack4 --- 0配置

必须有个src， index.js

可以直接webpack

消除警告，提示生产或开发模式

webpack --mode production

webpack --mode development

-

自定义build命令

"build":"webpack --mode development"

可执行 npm run build 命令

多入口

entry:['./src/index.js','./src/index2.js'] // 多入口

/\*

// 多入口,webpack4不得取名index

entry:{

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

index2:'./src/index2.js',

},

output:{

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

filename:'[name].bundle.js'

}

\*/

Index.html 引入index.bundle.js index2.bundle.js

- Html-webpack-plugin自动生成index.html

注意：需要依赖，装到本地

webpack, npm i webpack -D

webpack-cli, npm i webpack-cli -D

安装

npm i html-webpack-plugin -D

引入

const HtmlWebpackPlugin = require('html-webpack-plugin') // 自动打包

plugins:[

new HtmlWebpackPlugin({

//模板配置项 模板地址

template:'./src/index.html'

})

]

// 插件

plugins:[

new HtmlWebpackPlugin({

hash:true, // 消除缓存

// <%= htmlWebpackPlugin.options.title%> <%%>全局输出 模板页面不能注释

title:'i love u', // 页面标题，一定得在模板里面使用

template:'./src/index.html'// 模板地址

})

]

参数参考

// https://www.npmjs.com/package/html-webpack-plugin

// 插件

plugins:[

// https://www.npmjs.com/package/html-webpack-plugin

new HtmlWebpackPlugin({

// 压缩

minify:{

collapseWhitespace:true, //折叠空白区域

removeAttributeQuotes:true // 删除双引号

},

hash:true, // 消除缓存

// <%= htmlWebpackPlugin.options.title%> <%%>全局输出 模板页面不能注释

title:'i love u', // 页面标题，一定得在模板里面使用

template:'./src/index.html'// 模板地址

})

]

// 多入口---- 多文件

plugins:[

// https://www.npmjs.com/package/html-webpack-plugin

new HtmlWebpackPlugin({

// 各个页面对应各自模板文件（打包的JS），如果没有所有打包的JS都会到这个页面

chunks:['index'],

filename:'index.html',// 生成文件名

// 压缩

minify:{

collapseWhitespace:true, //折叠空白区域

removeAttributeQuotes:true // 删除双引号

},

hash:true, // 消除缓存

// <%= htmlWebpackPlugin.options.title%> <%%>全局输出 模板页面不能注释

title:'i love u', // 页面标题，一定得在模板里面使用

template:'./src/index.html'// 模板地址

}),

new HtmlWebpackPlugin({

chunks:['index2'],

filename:'index2.html',

title:"第二页面",

template:'./src/index2.html'

})

]

-清楚- dist

安装 clean

npm i clean-webpack-plugin -D

const CleanWebpackPlugin = require('clean-webpack-plugin') // 清楚打包

new CleanWebpackPlugin(), // new CleanWebpackPlugin(['dist']),报错

---devServer

npm i webpack-dev-server -D

配置devServer

devServer:{

//设置服务器访问的基本目录

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

// 服务器IP地址 localhost

host:'localhost',

// 设置端口

port:8090

}

配置 package.json

"dev":"webpack-dev-server --mode development"

Npm i internal-ip // 有的会报错，我的不会，直接可以 npm run dev

用rm - rf删除依赖项 如果一直报错，删除node\_modules

------------热更新和自动打开也没见

const webpack = require('webpack')

new webpack.HotModuleReplacementPlugin(), // 启动热更新，有的会报错，目前新版本不需要

// 或者可以在package.json文件中配置 "dev":"webpack-dev-server --mode development"

open:true, // 自动打开页面

// 热更新

hot:true

、

-------------------5

Loader

安装

处理css style-loader css-loader

npm i style-loader css-loader -D

配置

module:{

// 规则

rules:[

{

test:/\.css$/, // 以.css结尾的正则

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

loader:['style-loader','css-loader'] // 第二种写法

}

]

},

--压缩打包文件

Package.json build-- 配置为生产环境 --mnode production

插件方式的压缩--

npm i uglifyjs-webpack-plugin -D

Const uglify = require(‘uglifyjs-webpack-plugin’)

new Uglify(),// 打包JS压缩

------------图片

处理图片，安装

npm i file-loader url-loader -D

{

test:/\.(.png|jpg|gif)$/,

use:[

{

loader:'url-loader',

options:{

limit:50 // 转化Img,控制字节，取消base64

}

}

]

}

import youImg from '../src/images/you.jpg' // 引入图片

let oImg = new Image() // IMG对象

oImg.onload = function(){

document.body.appendChild(oImg)

}

oImg.src = youImg;

---分离CSS文件

安装

npm i extract-text-webpack-plugin -D

Webpack 3.x能使用 --

new ExtractTextPlugin('css/index.css')

use:ExtractTextPlugin.extract({

fallback:'style-loader', // 类似于回滚

use:'css-loader'

})

使用最新的版本 webpack4

npm i extract-text-webpack-plugin@next -D

配置分离CSS

{

test:/\.css$/, // 以.css结尾的正则

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

//loader:['style-loader','css-loader'], // 第二种写法

/\*

第三种写法

use:[

{loader:"style-loader"},

{loader:"css-loader"},

]

\*/

// 如果分离css，以上不需要

use:ExtractTextPlugin.extract({

fallback:'style-loader', // 类似于回滚

use:'css-loader',

publicPath:'../' //配置图片路径

})

},

{

test:/\.(.png|jpg|gif)$/,

use:[

{

loader:'url-loader',

options:{

limit:50, // 转化Img,控制字节，取消base64

outputPath:'image'

}

}

]

}

--新的插件

npm i mini-css-extract-plugin -D

new MiniCssExtractPlugin({// 新插件分离CSS，效果同ExtractTextPlugin

filename:'css/index.css',// 提取文件名

}),

use:[

{

loader:MiniCssExtractPlugin.loader,

options:{

publicPath:'../'

}

},

'css-loader'

]

Less,Scss

npm i less less-loader -D

{

test:/\.less$/,

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

}

{

test:/\.less$/,

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

use:ExtractTextPlugin.extract({

fallback:'style-loader',// 分离

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

})

},

npm i node-sass sass-loader -D

{

test:/\.(sass|scss)$/,

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

}

{

test:/\.(sass|scss)$/,

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

/\*

use:ExtractTextPlugin.extract({

fallback:'style-loader',// 分离

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

})

\*/

}

自动处理前缀

npm i postcss-loader autoprefixer -D

// autoprefixer 处理前缀的插件

Postcss.config.js 配置postCss

module.exports = {

plugin:[

require('autoprefixer')

]

}

配置loader

use:[

{loader:"style-loader"},

{loader:"css-loader"},

{loader:'postcss-loader'} // 自动处理前缀

]

消除冗余CSS

Purifycss

1. 下载 npm i purifycss-webpack purify-css -D
2. 引入 const PurifycssWebpack = rewquire(‘purifycss-webpack’)
3. 额外包,glob, npm i glob -D
4. 在plugins 配置路径

/\*

Node知识,以下区别不大

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

path.resolve(\_dirname,'src/index.html')

\*/

调试

x开启调试，

进入开发模式可以进入调试

在浏览器source中可以看到资源

Webpack3

需要配置 sourceMap

加入devtool:”source-map”

转化es678,jsx

-bable

-轻松使用ESnext,转化

//// 看babel 官网

Babel-core bable-loader // 过时babel-preset-es2015

使用babel-preset-env 当前环境代替babel-preset-es2015

1下载

--npm i babel-loader babel-core babel-preset-env -D

{

test:/\.(js|jsx)$/,

user:['babel-loader'],

exclude:/node\_modules/ // 不检测这个文件

}

可以使用touch .babelrc 创建babelrc文件

{

"presets": [

"env"

]

}

出现报错

babel-loader@8 requires Babel 7.x (the package '@babel/core').

If you'd like to use Babel 6.x ('babel-core'), you should install 'babel-loader@7'.

安装npm install babel-loader@7 -D

第7版本解决报错

--配置JSX语法

Npm i babel-preset-react - D

安装react

npm i react react-dom -D 安装react

{

"presets": [

"env","react"

]

}

关注老师GIT https//github.com/itstrive/webpack-yarn

// 这里配置react --出现错误，网上搜索一些配置react 的文章，reactp配置比较复杂

// 疑点 npm install -D babel-polyfill