# Webpak学习

Webpack是一个模块打包工具

## 安装

**npm init**  初始化

**npm webpack webpack-cli –g** 安装webpack

**webpack –v** 版本号

**npm info webpack** 查看版本等相关信息

## 运行webpack

**npx webpack index.js**

**npx webpack**  //会根据项目下的webpack.config.js进行打包

**npx webpack --config webpackconfig.js** //指定某个文件为配置文件进行打包操作

## webapck的基础配置

webpack.config.js

const path = require("path");

module.exports = {

mode:'production', //production（压缩）或者development（不压缩）模式

entry: {

main: './src/index.js'

},

output: {

filename: 'bundle.js',

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

}

}

## 模块loader（module）

### File-loader

const path = require("path");

module.exports = {

mode: 'production', //production（压缩）或者development（不压缩）模式

entry: {

main: './src/index.js'

},

module: {

rules: [

{

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

use: {

loader: 'file-loader',

options: {

//placeholder 占位符

name: '[name]\_[hash].[ext]',

outputPath: 'images/'

},

}

}

]

},

output: {

filename: 'bundle.js',

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

}

}

### url-loader

const path = require("path");

module.exports = {

mode: 'production', //production（压缩）或者development（不压缩）模式

entry: {

main: './src/index.js'

},

module: {

rules: [

{

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

use: {

loader: 'url-loader',

options: {

//placeholder 占位符

name: '[name]\_[hash].[ext]',

outputPath: 'images/',

limit: 2048 //小于2kb则把图片以base64的形式打包的js文件中

},

}

}

]

},

output: {

filename: 'bundle.js',

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

}

}

### 样式相关loader.

#### 安装

**npm I sass-loader node-sass –save-dev**

**npm I postcss-loader –save-dev**

#### Webpack.conf.js

const path = require("path");

module.exports = {

mode: 'production', //production（压缩）或者development（不压缩）模式

entry: {

main: './src/index.js'

},

module: {

rules: [

{

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

use: {

loader: 'url-loader',

options: {

//placeholder 占位符

name: '[name]\_[hash].[ext]',

outputPath: 'images/',

limit: 2048 //小于2kb则把图片打包的js文件中

},

}

},

{

test: /\.css$/,

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

},

{

test: /\.scss$/,

use: [

'style-loader',

{

loader: 'css-loader',

options: {

importLoaders: 2, //js直接引入之前都会一次从上至下走一遍loader

modules: true //开启css模块化打包

}

},

'sass-loader',

'postcss-loader' //能自动添加css浏览器前缀

]

},

]

},

output: {

filename: 'bundle.js',

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

}

}

#### postcss.config.js

module.exports = {

plugins: [

require('autoprefixer')

]

}

#### 字体图标相关

{

test: /\.(eot|ttf|svg|woff)$/,

use: {

loader: 'file-loader'

}

}

## Plugins

plugins可以在webpack运行到某个时刻的时候帮你做一些事情

### html-webpack-plugin

会在打包结束后自动生成一个html文件，并把打包生成的js自动引入到这个html文件中

### clean-webpack-plugin

会在打包之前运行 清理指定目录

Webpack.config.js

const path = require("path");

const HttpWebpackPlugun = require("html-webpack-plugin"); //会在打包结束后自动生成一个html文件，并把打包生成的js自动引入到这个html文件中

const CleanWeboackPlugin = require("clean-webpack-plugin");

//plugin 可以在webpack运行到某个时刻的时候帮你做一些事情

module.exports = {

mode: 'development', //production（压缩）或者development（不压缩）模式

entry: {

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

sub: './src/index.js'

},

module: {

rules: [

{

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

use: {

loader: 'url-loader',

options: {

//placeholder 占位符

name: '[name]\_[hash].[ext]',

outputPath: 'images/',

limit: 2048 //小于2kb则把图片以base64的形式打包的js文件中

},

}

},

{

test: /\.scss$/,

use: [

'style-loader',

{

loader: 'css-loader',

options: {

importLoaders: 2

}

},

'sass-loader',

'postcss-loader'

]

}

]

},

plugins: [ // plugin 可以在webpack运行到某个时刻的时候帮你做一些事情

new HttpWebpackPlugun({ // 会在打包之后运行

template: 'src/index.html'

}),

new CleanWeboackPlugin(['dist']) // 会在打包之前运行 清理dist目录

],

output: {

publicPath: 'http://cdn.com.cn/', // html引入相应的js时会在js前面增加http://cdn.com.cn/前缀

filename: '[name].js',

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

}

}

## source-map

配置项区别

none, 不做映射

source-map, 打包出单独的map文件

inline-source-map, 把map文件转为base64的形式嵌入到打包出的文件中具体到第几行第几个字符

cheap-inline-source-map cheap指定到某一行

cheap-module-inline-source-map 不仅指定业务代码，也显示第三方模块快中的错误

eval

source-map配置建议

development devtool:'cheap-module-eval-source-map'

production devtool:'cheap-module-source-map'

webpack.config.js相关代码

mode: 'production', //production（压缩）或者development（不压缩）模式

devtool: 'cheap-module-eval-source-map', //具体内容看webpack笔记

//development devtool:'cheap-module-eval-source-map', 建议

//production devtool:'cheap-module-source-map', 建议

entry: {

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

},

## Webpack-devServer

### 安装

npm I webpack-dev-server -D

### Package.js

{

"name": "test3",

"version": "1.0.0",

"description": "",

"main": "webpack.config.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"bundle": "webpack",

"watch": "webpack --watch", // 实现监听

"start": "webpack-dev-server", // devserver打开

"server": "node server.js" // 自定义server

},

"author": "",

"license": "ISC"

}

### Server.js

const express = require('express');

const webpack = require('webpack');

const webpackDevMiddleware = require("webpack-dev-middleware");

const config = require("./webpack.config"); //在node中使用webpack

const complier = webpack(config);

const app = express();

app.use(webpackDevMiddleware(complier,{

publicPath : config.output.publicPath

}))

app.listen(3000,() => {

console.log('listen 3000...')

})

### Webpack.config.js

devServer: {

contentBase: './dist',

open: true,

port: '8888'

}

## 热模块更新

**HotModuleReplacementPlugin**

### Webpack.config.js

devServer: {

contentBase: './dist',

open: true,

port: '8888',

hot: true, //热模块替换

// hotOnly: true //强值只是用热模块更新，而不刷新页面

},

plugins: [

new HttpWebpackPlugun({ //会在打包之后运行

template: 'src/index.html'

}),

new CleanWeboackPlugin(['dist']), //会在打包之前运行 清理dist目录

new webpack.HotModuleReplacementPlugin() //热模块替换

],

Index.js

function counter() {

var div = document.createElement('div');

div.setAttribute('id','counter');

div.innerHTML = 1;

div.onclick = function(){

div.innerHTML = parseInt(div.innerHTML,10) + 1

}

document.body.appendChild(div)

}

function number() {

var div = document.createElement('div');

div.setAttribute('id','number');

div.innerHTML = 1000;

document.body.appendChild(div)

}

counter()

number()

if(module.hot){ //如果number发生了变化则执行里面的代码

module.hot.accept('./number', () => {

var ele = document.getElementById('number');

document.body.removeChild(ele);

number();

});

}

## Babel和react

### Webpack.config.js相关代码

module: {

rules: [

{

test: /\.js$/,

exclude: /node\_modules/,

loader: "babel-loader"

},

]

}

### .babelrc文件：

{

//平时开发时使用下面的配置

"presets": [

["@babel/preset-env",{

"targets": { //指定版本后无需转换

"edge": "17",

"firefox": "60",

"chrome": "67",

"safari": "11.1",

},

"useBuiltIns": "usage" //当转es5的时候实现按需加载 @babel/polyfill

}],

"@babel/preset-react" //react相关

]

//若开发第三方组件或库时使用下面代码 可以有效的避免全局污染的问题

// "plugins": [["@babel/plugin-transform-runtime",{

// "corejs": 2,

// "helpers": true,

// "regenerator": true,

// "useESModules": false

// }]]

}