Webpack基础

### 一、了解什么是webpack

Webpack 是一个前端的静态模块资源打包工具，能让浏览器也支持模块化。它将根据模块的依赖关系进行静态分析，然后将这些模块按照指定的规则生成对应的静态资源。

Webpack 核心主要进行 JavaScript 资源打包，它可以结合其他插件工具，将多种静态资源css、png、sass 分类转换成一个个静态文件，这样可以减少页面的请求。可集成 babel 工具实现 EcmaScript 6 转 EcmaScript 5 ，解决兼容性问题可集成 http 服务器可集成模块热加载，当代码改变后自动刷新浏览器 。

也就是说webpack是优化性能的工具

### webpack的基础组成部分

Webpack基础学习目录

创建以下目录结构、文件和内容：

mkdir webpack-demo

cd webpack-demo

npm init -y

npm install webpack webpack-cli --save-dev

webpack-demo

|- package.json

|- package-lock.json

|- webpack.config.js

|-/dist

| - bundle.js

| - index.html

|- /src

|- index.js

|- /node\_modules

****package.json****

{ "name": "webpack-demo",

"version": "1.0.0",

"description": "",

"private": true,

"scripts": {

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

"build": "webpack"

},

"keywords": [],

"author": "",

"license": "ISC",

"devDependencies": {

"webpack": "^5.4.0",

"webpack-cli": "^4.2.0"

},

"dependencies": {

"lodash": "^4.17.20"

}

}

****src/index.js****

****npm install --save lodash****

import \_ from 'lodash';

function component() {

const element = document.createElement('div');

// lodash 在当前 script 中使用 import 引入

element.innerHTML = \_.join(['Hello', 'webpack'], ' ');

return element;

}

document.body.appendChild(component());

在这个设置中，index.js 显式要求引入的 lodash 必须存在，然后将它绑定为 \_没有全局作用域污染）。通过声明模块所需的依赖，webpack 能够利用这些信息去构建依赖图，然后使用图生成一个优化过的 bundle，并且会以正确顺序执行。

****dist/index.html****

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8" />

<title>起步</title>

</head>

<body>

<script src="bundle.js"></script>

</body>

</html>

****webpack.config.js中配置以下属性****

****webpack.config.js****

入口(entry) + 输出(output)

const path = require('path');

module.exports = {

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

output: {

filename: 'bundle.js',

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

},};

loader（预处理）

module中配置模块 loader 可以链式调用。链中的每个 loader 都将对资源进行转换。链会逆序执行。第一个 loader 将其结果（被转换后的资源）传递给下一个 loader，依此类推。最后，webpack 期望链中的最后的 loader 返回 JavaScript。

配置对应的loader就可以，使用对应的文件

可以使用之前的配置的命令npm run build

如果打包出错，可能是mode没有配置

在 webpack 5 中，可以使用内置的 [Asset Modules](https://webpack.docschina.org/guides/asset-modules/)，我们可以轻松地将这些内容混入我们的系统中：常用的图片和字体包裹在其中

const path = require('path');

module.exports = {

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

output: {

filename: 'bundle.js',

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

},

module: {

rules: [

{

test: /\.css$/i,

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

},

{

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

type: 'asset/resource',

},

{

test: /\.(woff|woff2|eot|ttf|otf)$/i,

type: 'asset/resource',

},

],

},

};

1. 插件(plugins)

npm install --save-dev html-webpack-plugin

const path = require('path');

const HtmlWebpackPlugin = require('html-webpack-plugin');

module.exports = {

entry: {

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

},

output: {

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

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

}

plugins: [

new HtmlWebpackPlugin({

title: '管理输出',

}),

],

};

5、mode(模式)

mode分为两种一种是生产环境和开发环境

production 和 development(本文使用)