**基于webpack的React项目开发**

**参考文档（<https://www.cnblogs.com/raion/p/8053799.html>）**

#### ****安装react, react-dom, webpack：****

npm install react react-dom --save  
npm install webpack --save-dev



3、webpack.config.js文件

const path = require('path');

const webpack = require('webpack');

module.exports = {

entry: path.resolve(\_\_dirname, '../src/index.js'), //指定入口文件，程序从这里开始编译,\_\_dirname当前所在目录, ../表示上一级目录, ./同级目录

output: {

path: path.resolve(\_\_dirname, '../dist'), // 输出的路径

filename: 'bundle.js' // 打包后文件

},

module: {

rules: [

{

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

loader: 'babel-loader',

exclude: /node\_modules/

}

]

}

}

1. index.js

import React from 'react';

import { render } from 'react-dom';

import App from './App'

const renderDom = Component => {

render(

<Component />,

document.getElementById('app')

);

}

renderDom(App);

1. 在项目根目录下新建json文件.babelrc，将babel的配置单独提取出来。

{

  "presets": [

    "es2015",

    "react"

  ]

}

1. index.html

<!DOCTYPE html><html lang="en"><head>

<meta charset="UTF-8">

<title>Title</title></head><body>

<div id="app">

/\* react DOM\*/

</div>

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

1. 在package.json中添加

webpack --config webpack/webpack.config.js // 更多webpack命令尽在webpack --help查阅

1. 安装相关配置：

**npm install babel-core babel-loader babel-preset-es2015 babel-preset-react --save**

1. 在src下新建App.js。

import React, { Component } from 'react';

export default class App extends Component {

render() {

return (

<div>Hello React!</div>

);

}

}

1. 运行webpack打包

Npm run build

1. 然后点击build/index.html，即可看到Hello React！

**(注意：package.json文件**

{

"name": "react",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

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

"build": "webpack --config webpack/webpack.config.js",

"dev": "node bin/dev-server.js"

},

"keywords": [],

"author": "",

"license": "ISC",

"dependencies": {

"acorn-dynamic-import": "^3.0.0",

"anymatch": "^2.0.0",

"babel-core": "^6.26.3",

"babel-loader": "^7.1.5",

"babel-preset-es2015": "^6.24.1",

"babel-preset-react": "^6.24.1",

"bluebird": "^3.5.1",

"brace-expansion": "^1.1.11",

"chownr": "^1.0.1",

"collection-visit": "^1.0.0",

"copy-descriptor": "^0.1.1",

"eslint-scope": "^4.0.0",

"express": "^4.16.3",

"has-values": "^2.0.1",

"html-webpack-plugin": "^3.2.0",

"is-accessor-descriptor": "^2.0.0",

"is-data-descriptor": "^1.0.0",

"is-extendable": "^1.0.1",

"mississippi": "^3.0.0",

"once": "^1.4.0",

"path-dirname": "^1.0.2",

"posix-character-classes": "^1.0.0",

"promise-inflight": "^1.0.1",

"pseudomap": "^1.0.2",

"react": "^16.4.2",

"react-dom": "^16.4.2",

"safe-regex": "^1.1.0",

"schema-utils": "^1.0.0",

"set-immediate-shim": "^2.0.0",

"snapdragon-util": "^5.0.1",

"uglifyjs-webpack-plugin": "^1.3.0",

"union-value": "^1.0.0",

"unique": "^0.0.1",

"unique-slug": "^2.0.0",

"unset-value": "^1.0.0",

"upath": "^1.1.0",

"webpack": "^4.17.1",

"webpack-cli": "^3.1.0",

"webpack-dev-server": "^3.1.7",

"wrappy": "^1.0.2"

},

"devDependencies": {

"webpack": "^4.17.1",

"webpack-dev-server": "^3.1.7"

}

}

**)**

1. webpack-dev-server是一个小型的静态文件服务器，为webpack打包的资源文件提供Web服务。**安装**webpack-dev-server：（千万不要用cnpm,否则需要手动下载以上配置文件）

npm install webpack-dev-server --save-dev

1. 在项目根目录下创建bin目录，进入bin目录，创建dev-server.js文件

'use strict'

const WebpackDevServer = require('webpack-dev-server');

const config = require('../webpack/webpack.config');

const webpack = require('webpack');

const path = require('path');

const compiler = webpack(config);

const server = new WebpackDevServer(compiler, {

contentBase: path.resolve(\_\_dirname, '../dist'), //默认会以根文件夹提供本地服务器，这里指定文件夹

historyApiFallback: true, //在开发单页应用时非常有用，它依赖于HTML5 history API，如果设置为true，所有的跳转将指向index.html

port: 9090, //如果省略，默认8080

publicPath: "/"

});

server.listen(9090, 'localhost', function (err) {

if (err) throw err

})

1. 删除根目录下dist目录，刚刚我们是自己编写和配置index.html，将打包后的js引入到index.html中。现删除后启动服务会报错，现在我们使用插件实现自动引入，免去手工配置，安装**html-webpack-plugin**。
2. 在src目录新建index.template.html。

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title><%= htmlWebpackPlugin.options.title %></title>

</head>

<body>

<div id="app">

</div>

</body>

</html>

1. 编辑webpack.config.js

const path = require('path');

const webpack = require('webpack');

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

module.exports = {

entry: path.resolve(\_\_dirname, '../src/index.js'), //指定入口文件，程序从这里开始编译,\_\_dirname当前所在目录, ../表示上一级目录, ./同级目录

output: {

path: path.resolve(\_\_dirname, '../dist'), // 输出的路径

filename: 'app/[name]\_[hash:8].js' // 打包后文件

},

module: {

rules: [

{

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

loader: 'babel-loader',

exclude: /node\_modules/

}

]

},

　　 plugins: [

　　 　　new HtmlWebpackPlugin({

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

　　　　　　 inject: true

　　　　 })

　　 ]

}

1. 运行项目：

Npm run dev

1. 热更新(HMR)参考文档：（<https://www.cnblogs.com/raion/p/8111396.html>）——可以在代码修改后重新打包并发送到浏览器，浏览器将获取的新模块替换老模块，在不刷新浏览器的情况下实现对应用的更新。