## **构建：webpack一步一步构建**

****安装webpack****

在安装webpack之前，我们先建立一个文件夹并进入，当然你可以用其他方法，我这里就用命令了。

mkdir react-webpack

cd react-webpack

进入文件夹后对我们的webpack项目进行****初始化****，命令如下：

npm init

初始化成功后可以在项目根目录下看到package.json文件。package.json文件建立好以后，你可以安装webpack了。

npm install --save-dev webpack

建议使用cnpm来进行安装，这样速度会快很多。安装好后，你会在package.json里看到你安装的版本号。这样我们webpack就安装好了。

****配置webpack.config.js****

在项目根目录建立webpack.config.js文件，这个文件是进行webpack配置的，先建立基本的入口和出口文件。

var path =require('path');

module.exports = {

//入口文件

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

//出口文件

output:{

filename:'index.js',

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

}

}

****新建index.html文件****

在根目录新建index.html文件，并引入webpack设置中的出口文件，代码如下。

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

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>React全家桶-jspang</title></head><body>

</body><!--引入出口文件--><script src="./dist/index.js"></script></html>

****测试webpack配置****

通过上面的步骤，配置好了wbpack，现在可以测试一下我们webpack配置是否有问题。在路口文件中写入下面的代码，并进行打包测试。

****/app/index.js文件****

function component(){

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

element.innerHTML = ('Hello JSPang');

return element;

}

document.body.appendChild(component());

这段代码非常简单，不作过多的解释。如果你对这个代码看不懂，再去啃两边犀牛或者红宝书。

****加入打包命令****

打开package.json文件，在scripts属性中加入build命令。

"scripts": {

"build": "webpack"

},

在终端中输入npm run build ，就可以看到打包结果了。如果没有出现错误，在浏览器中打开，可以看到我们的编写结果。到这部为止，我们正确安装了webpack，并进行了出入口的配置，也看到了webpack的输出结果。

****开发服务器配置****

你会很容易的发现，我们现在缺少一个实时更新的服务，那动手马上配置一个。

在命令行安装webpack-dev-server,在终端中输入下面的命令。

cnpm install --save-dev webpack-dev-server

安装完成后，配置webpack.config.js文件。

devServer:{

contentBase:'./',

host:'localhost',

compress:true,

port:1717}

配置好后再packeage.json里增加一个scripts命令，我们起名叫server。

"scripts": {

"build": "webpack",

"server": "webpack-dev-server --open"

},

这里的–open是直接打开浏览器的意思。这些都配置完成后，就可以在终端输入npm run server 看到结果了。

****自动刷新浏览器****

在我们修改代码时，现在并不能自动刷新浏览器，不会立即呈现我们编写的代码结果，而时要再次npm run build才可以。其实只要在出口文件配置中加一个publicPath:’temp/’,然后在index.html引入JS时写这个temp地址就可以实时预览了。

output:{

filename:'index.js',

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

publicPath:'temp/'},

index.html文件引入JS

<script src="./temp/index.js"></script>

****Babel安装配置****

在webpack中配置Babel需要先加入babel-loader，这里我们继续使用npm来进行安装，但是我们一般还需要支持es2015和React，所以要安装四个包。

cnpm install --save-dev babel-core babel-loader babel-preset-es2015 babel-preset-react

这里安装四个包的安装，这个四个包不能省略。安装后你会在package.json里看到这些包的版本。

"devDependencies": {

"babel-core": "^6.26.0",

"babel-loader": "^7.1.2",

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

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

"webpack": "^3.8.1",

"webpack-dev-server": "^2.9.3"

}

****配置module****

在安装完成后，可以到webpack.config.js里配置module，也就是配置我们常说的loader。

module:{

loaders:[

{

test:/\.js$/,

exclude:/node\_modules/,

loaders:"babel-loader",

query:{

presets:['es2015','react']

}

}

]

}

****编写React****

webpack通过上边的步骤，基本已经配置完成了，这里我们写一个React文件来进行测试一下。在进行编写React文件前，要记得安装React和React-dom这两个包。

安装React和React-dom

cnpm install --save react react-dom

安装完成后，我们改写app/index.js文件，把原来原生的JavaScript代码改成React代码。

****index.js****

import React from 'react';import ReactDOM from 'react-dom';

ReactDOM.render(

<div>Hello JSPang</div>,

document.getElementById("app")

);

因为上边我们加了一个app的选择器，所以在index.html中增加一个div标签，并把id设置成app。

****index.html****

<div id="app"></div>

都配置完了，我们可以用之前配置好的npm run server 命令进行预览了。如果在浏览器中看到了Hello JSPang 这段文字，说明我们的配置成功了，以后就可以愉快的进行开发了。

****总结：****其实自己配置React开发环境在实际工作中也并不多，因为我们的配置毕竟不是很成熟，而且支持较少。这里学习只是为了你更好的掌握React构建过程和设置参数，实际工作中尽量使用合适的脚手架工具。

转载自：jspang.com/2017/10/22/react-all-01/

webpack.config.js的完整代码

var path = require('path');

module.exports = {

// 入口文件

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

// 服务器配置

devServer: {

contentBase: './',

host: 'localhost',

compress: true,

port: 1717

},

// 配置Babel

module:{

loaders:[

{

test:/\.js$/,

exclude:/node\_modules/,

loaders:"babel-loader",

query:{

presets:['es2015','react']

}

}

]

},

// 出口文件

output: {

filename: 'index.js',

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

publicPath: 'temp/'

}

}