



webpack+react项目初体验——记录我的webpack环境配置

react.js javascript webpack

yisha0307 2月5日发布

这两天学习了一些webpack的知识，loaders+plugins真的很强大！不过配置起来也很复杂，看了一些文章，自己也写了项目练手，写下来加深自己的印象。

首先，非常非常推荐的几篇文章，初学者一定要看！

<入门Webpack，看这篇就够了>

<http://www.jianshu.com/p/42e1...>

（标题一点也不夸张，非常适合0基础）

<Webpack傻瓜式指南>

<https://zhuanlan.zhihu.com/p/...>

<https://zhuanlan.zhihu.com/p/...>

<https://zhuanlan.zhihu.com/p/...>

（这个系列有三篇文章，第三章是一个webpack+react的小项目，跟着做一遍会很有收获~）

另外，也推荐看一下阮一峰es6书中module这一章，弄清楚export/import/export default等等命令，毕竟webpack的各个模块是靠export/import/require(commonjs)链接起来的，所以这些都要掌握。

具体到项目的话，webpack有几个比较基本的概念：

- 1、**loaders**：通过不同的loaders，webpack可以处理各式各样的文件，然后打包到一个文件中（比如bundle.js）；
- 2、**plugins**：plugins是为了拓展webpack的功能的，和loaders不同的是，loader是用来处理单个文件的（比如json-loader处理.json,sass-loader处理.scss），但是plugins是直接对整个构建过程进行处理（比如自动生成html文件的html-webpack-plugin）；
- 3、**others**：这些我也不知道要归到哪里去，但是在配置中也是必不可少，包括webpack-dev-server/source-map等等，后面会具体说；
- 4、**配置文件**：我这个小项目包括的文件有.babelrc(用来处理babel)，webpack.config.js(webpack项目基础配置文件)，package.json(这个文件会记录所有的devDependencies)。

然后我们就一项一项来分析吧：

1、loaders

1) style-loader / css-loader / sass-loader

这几个loader用来处理.css和.scss文件，一起安装用空格隔开：

```
$ npm install --save-dev style-loader css-loader sass-loader
```

同时修改webpack.config.js:

```
module: {
  loaders: [
    {
      test: /\.scss$/,
      //loaders是依靠正则表达式来测试这个文件是不是这个loader来处理，所以test不能少
      loaders: ['style-loader', 'css-loader', 'sass-loader'],
      //"-loader"一定要写，不然会报错
      //loaders的处理顺序是从右向左，就是会先用sass-loader，其次css-loader，再次style-loader
    }
  ]
}
```

2) url-loader

这个loader是用来处理url链接，就是图片或者其他静态文件。

安装：

```
$ npm install --save-dev url-loader
```



问答



头条



专栏



讲堂



更多

webpack.config.js (写在module里):

```
{
  test: /\. (png|jpg)$/,
  loader: 'url? limit = 40000'
}
```

3) json-loader

安装和配置和之前一样~用来处理json文件

4) babel相关的loaders:

这个包括的就比较多, 有babel-core/babel-loader/babel-preset-es2015/babel-preset-react, 后面两个是为了写es6和react服务。

```
//webpack.config.js
{
  test: /\.jsx$/,
  loader: 'babel-loader',
  include: APP_PATH,
  //这个include是说只对这里面的文件负责, 还有一个对应的exclude, 就是忽略范围内的文件, 比如: exclude: './node_modules/';
}
```

另外因为babel需要写的选项比较多, 可以配一个.babelrc在根目录下:

```
//.babelrc
{
  'presets': ['react', 'es2015'],
}
```

以上就是用的比较多的loaders, 配完这些webpack就可以处理json/sass/es6啦~

2、plugins

1) html-webpack-plugin

这个插件的作用就是自动生成html (其实也可以自己写, 就是加了个bundle.js的script而已, 不过感觉比较酷):

plugins安装好了之后要放在webpack.config.js的plugins的数组里, 不要写在modules里呀~

```
//webpack.config.js
plugins: [
  new HtmlWebpackPlugin({
    //在最前面先定义下HtmlWebpackPlugin--
    //var HtmlWebpackPlugin = require('html-webpack-plugin');
    title: 'searchBar', //配合html-webpack-plugin的配置
  })
],
```

2) react-transform-hmr

HMR是一个webpack插件, 它让你能浏览器中实时观察模块修改后的效果, 但是如果你想让它工作, 需要对模块进行额外的配额;

Babel有一个叫做react-transform-hmr的插件, 可以在不对React模块进行额外的配置的前提下让HMR正常工作;

安装:

```
$ npm install --save-dev babel-plugin-react-transform react-transform-hmr
```

配置:

```
//webpack.config.js (plugins里)
new webpack.HotModuleReplacementPlugin();
```

然后修改下.babelrc:

```
{
  "presets": ["react", "es2015"],
  "env": {
    "development": {
      "transform": ["react-transform-hmr"]
    }
  }
}
```



问答



头条



专栏



讲堂


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

var $ = require('jquery');
var moment = require('moment');
```

然后就可以愉快地写各种js、jsx文件啦~

4、配置文件

最后我们来讲一讲几个配置文件的问题：

1) webpack.config.js

上面提到的都是各种肢解，我这次的config文件是这样的：

```
var path = require('path');
var HtmlWebpackPlugin = require('html-webpack-plugin');
var ROOT_PATH = path.resolve(__dirname);
var APP_PATH = path.resolve(ROOT_PATH, 'app');
var BUILD_PATH = path.resolve(ROOT_PATH, 'build');

module.exports = {
  devtool: 'eval-source-map',
  entry: __dirname + '/app/index.jsx',
  //webpack的入口文件只有一个，所以写的所有components甚至包括css/json什么的，都要引用在这里
  output: {
    path: __dirname + '/public',
    filename: 'bundle.js',
  },
  //我这边是新建了一个folder叫public，用来放index.html和bundle.js
  devServer: {
    contentBase: './public', //本地服务器所加载的页面所在的目录
    historyApiFallback: true, //不跳转
    inline: true //实时刷新
  },
  plugins: [
    new HtmlWebpackPlugin({
      title: 'searchBar', //配合html-webpack-plugin的配置
    })
  ],
  module: {
    loaders: [
      {
        test: /\.scss$/,
        loaders: ['style-loader', 'css-loader', 'sass-loader']
      }
    ]
  }
}
```

2) package.json

这个文件会在你最开始npm init的时候就生成，一路回车就可以，后来都可以改~

```
{
  "name": "search-bar",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "webpack-dev-server --progress --profile --colors --hot",
    "build": "webpack --progress --profile --colors",
    "test": "karma start"
  },
  //scripts这边可以改一下，改start可以，在终端用npm start,上面有例子~这边的dev要改的话在终端的命令是'npm run dev';
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "babel-core": "^6.22.1",
    "babel-loader": "^6.2.10",
    "babel-plugin-react-transform": "^2.0.2",
    "babel-preset-es2015": "^6.22.0",
    "babel-preset-react": "^6.22.0",
    "babel-preset-react-hmre": "^1.1.1",
    "bootstrap": "^4.0.0-alpha.2",
    "css-loader": "^0.26.1",
    "file-loader": "^0.10.0",
    "html-webpack-plugin": "^2.28.0",
    "jquery": "^3.1.1",
    "jshint": "^2.9.4",
    "jshint-loader": "^0.8.3",
    "json-loader": "^0.5.4",
    "node-sass": "^4.5.0",
    "react": "^15.4.2"
  }
}
```

装了很多dev，其实用不着那么多哈哈~

3) .babelrc



问答



头条



专栏



讲堂

```
'development':{
  'presets':['react-hmre']
}
}
```

ok，这样就差不多啦~另外还要注意的是index.jsx/index.js，所有的components都要引用过来，css/scss也是，css文件的话最好有一个main.css进行整合，这样不会漏掉。

看一眼这次的index.jsx：

```
// 注意这些import
import '../node_modules/bootstrap/scss/bootstrap.scss';
import './main.scss';
import React from 'react';
import ReactDOM from 'react-dom';
import Search from './components/search';
import Plist from './components/plist';

class App extends React.Component{
  constructor(props){
    super(props);
    this.state={ 'keyword':'' };
    this.refreshKeyword = this.refreshKeyword.bind(this);
  }
  refreshKeyword(name){
    this.setState({ 'keyword':name });
  }
  render(){
    return (
      <div className = 'container'>
        <section className = 'jumbotron'>
          <h3 className = 'jumbotron-heading'>Search Github Users</h3>
          <Search sendAction = {this.refreshKeyword} />
        </section>
        <Plist keyword={this.state.keyword} />
      </div>
    )
  }
};
```

恩，差不多就是这样啦~~项目初始化的时候不要忘记npm install --save-dev webpack哦！coding愉快！

2月5日发布

▲ | 0

收藏 | 6

- 你可能感兴趣的文章
- [react基于webpack和Babel 6上的开发环境搭建](#) 135 收藏，5.5k 浏览
- [webpack自动刷新](#) 37 收藏，4.5k 浏览
- [React技术栈一览](#) 21 收藏，519 浏览

3 条评论

默认排序 | 时间排序

JeremyChen · 2月10日

作者应该把你的写这篇文章的小demo贴出来，方便学习。

👍 赞 回复

不咬人的蚊子 · 3月3日

简直了。女生太细致了。写的好。

👍 赞 回复

嘿嘿~我又写了篇新的 — yisha0307 作者 · 3月6日

添加回复

发布评论



yisha0307
62 声望

关注作者

发布于专栏

前端自学记录

记录一些自己的学习心得，欢迎大家指点拍砖~

2 人关注

关注专栏



问答



头条



专栏



讲堂