Webpack傻瓜指南（三）和React配合开发

张轩 · 2 年前

经过两章的学习，大家都了解Webpack的基本和高阶一点的用法，那么最后一章来说说和现在前端最火的明星React一起合作的故事。本文适合对React有一定了解的朋友，如果还不知道React是

如果您还没有阅读前面两篇关于webpack的基础和进阶，请先务必阅读之前的文章。

- Webpack傻瓜式指南（一） - 前端外刊评论 - 知乎专栏
- Webpack傻瓜指南（二）开发和部署技巧

什么是React

React是一个由Facebook开发的library，它的口号是“A JAVASCRIPT LIBRARY FOR BUILDING USER INTERFACES”，用于构建用户界面的库。他的特点是仅仅关注于UI层，和其他的一系列大型的框架（Ember.js和Angular.js等）的理念不同，上述两个框架给你提供了一整套的解决方案。还有一个重大的改革就是React采用了一种独特的技术被称为virtual DOM ，和其他传统框架的DOM渲染过程不同，提供了更高性能的渲染，

> 如果你对virtual DOM感兴趣，可以看看这个库：Matt-Esch/virtual-dom
>
> 如果你想深入了解React 请前往官方网站React.js

为什么要用React和Webpack配合在一起

现在有很多的构建工具，采用Webpack的原因就是简单易用，容易上手，并且对React完美的支持，比如说JSX语法，使用babel-loader以后可以很完美的支持，支持Hot Loading Component,让你不用忍受页面刷来刷去的痛苦，还有就是配置简单，loaders的魔力是没人能够拒绝的。

如果你的下一个项目采用React做为View的展示框架，不妨再使用Webpack一起搭建一个开发环境。

配置React和Webpack

现在开始配置，把两个库的魔力融合在一起，在这里同样是实践出真知，来做一个小项目一步步完成这部教程，项目很简单，不是已经被别人做烂了的Todo了，这个项目可以搜索github上的用户名称，并且展示出来搜索出来的信息。 先画一个原型图：

从上图可以看出 项目有两个components，一个是搜索框，一个是展示列表，列表有几种状态，初始状态，正在读取数据，数据读取完毕展示列表。

根据原型图，项目应该有如下简单的结构

程序结构

- app/
  - index.jsx(程序入口)
  - components/(组件文件夹)
    - plist.jsx(展示用户列表)
    - search.jsx(搜索框组件)
  - utils/(一些小工具)
- package.json
- webpack.config.js

配置Babel让React支持ES6

同的React也做了更新，更好的支持了ES6的写法。

如果你现在还是用这种方法写你的component

```
var List = React.createClass({
    getInitialState: function() {
        return ['a', 'b', 'c']
    },
    render: function() {
        return ( ... );
    }
});
```

那么是时候采用ES6的写法了 React ES6 classes 这种写法看起来更棒，可读性也更强。一个直观的发现就是不用写getInitialState方法了，可以直接在constructor里面定义this.state的值，前端一直都是在高速前进的，采用最新的标准，使用最新的技术，这应该是每个人的追求，对吧。

```
import React from 'react';

class List extends React.Component {
    constructor() {
        super();
        this.state = ['a', 'b', 'c'];
    }
    render() {
        return (...);
    }
}
```

安装babel-loader

为了让上述的写法变成现实。需要安装bable和babel的两个preset：

```
//install babel core
npm install babel-loader --save-dev
//install es6 support and react support
npm install babel-preset-es2015 babel-preset-react --save-dev
```

这里安装了babel的主体和两个babel的preset, 什么是preset呢，你可以把preset看成一个包，里面有各种各样一系列的插件。

- babel-preset-es2015 es6语法包，有了这个，你的代码可以随意的使用es6的新特性啦，const,箭头操作符等信手拈来。

React ES6 classes的写法，同时直接支持JSX语法格式。更多详情，请到Babel Plugins

看一下这两个语法包都包括什么插件，每个插件都有什么特性。

配置webpack

像我们在前面做到一样，创建webpack.config.js

```
var path = require('path');
var webpack = require('webpack');
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= {
  entry: {
    app: path.resolve(APP_PATH, 'index.jsx')
  },
  output: {
    path: BUILD_PATH,
    filename: 'bundle.js'
  },
  //enable dev source map
  devtool: 'eval-source-map',
  //enable dev server
  devServer: {
    historyApiFallback: true,
    hot: true,
    inline: true,
    progress: true
  },
  //babel重要的loader在这里
  module: {
    loaders: [
      {
        test: /\.jsx?$/,
        loader: 'babel',
        include: APP_PATH,
        query: {
          //添加两个presents 使用这两种presets处理js或者jsx文件
          presets: ['es2015', 'react']
```

```
  },
  plugins: [
    new HtmlwebpackPlugin({
      title: 'My first react app'
    })
  ]
}
```

这里还需要添加一个resolve的参数，把jsx这种扩展名添加进去，这样就可以在js中import加载jsx这种扩展名的脚本

```
...
resolve: {
    extensions: ['', '.js', '.jsx']
},
...
```

npm中添加webpack启动命令

就像第一章里面介绍的，把命令添加到package.json里面。

package.json

```
...
"scripts": {
  "dev": "webpack-dev-server --progress --profile --colors --hot",
  "build": "webpack --progress --profile --colors",
  "test": "karma start"
},
...
```

添加首页

安装React和React-Dom

```
npm install react react-dom --save
```

让样式好看点，添加一个Bootstrap 4

```
npm install bootstrap@4.0.0-alpha.2 --save-dev
```

为了处理scss 需要添加sass loader 第一节讲过 大家都没忘记吧？

在webpack.config.js中添加处理的loader

```
...
 module: {
   loaders: [
   ...
     {
       test: /\.scss$/,
       loaders: ['style', 'css', 'sass']
     }
   ...
   ]
 },
 ...
```

index.jsx 使用es6的格式

```
import '../node_modules/bootstrap/scss/bootstrap.scss';
import React from 'react';
import ReactDOM from 'react-dom';

class App extends React.Component{
    constructor() {
        super();
    }
    render() {
        //JSX here!
        return (
          <div className="container">
            <section className="jumbotron">
              <h3 className="jumbotron-heading">Search Github Users</h3>
            </section>
          </div>
        )
    }
};

const app = document.createElement('div');
document.body.appendChild(app);
ReactDOM.render(<App />, app);
```

再用webpack运行就可以看到结果了。

```
npm run dev
```

添加React Transform支持

更新上面所说的React Hot Loading已经过时了，开发者也宣布已经停止维护，现在有一个更强大的babel plugin： React Transform来代替他。

现在每次修改一个component的代码，页面都会重新刷新，这会造成一个很不爽的问题，程序会丢失状态，当然现在在简单的程序中这个完全无所谓，但是假如程序变得越来越复杂，想要返回这种状态你可能又要经历一系列的点击等操作，会耗费一些时间。

隆重推出Babel-plugin-react-transform 名字挺长, 看起来挺吓人，其实你就可以想象用这个东西可以实时的对你的React Component做各种处理，它是基于Babel Plugin。 废话不多说，花点时间感受一下它是怎么玩的。

先安装

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

这是个基本的架子，可以通过它完成各种transform，如果想实现Hot Module Replacement (说白了就是页面不刷新，直接替换修改的Component)，再安装一个transform.

```
npm install --save-dev react-transform-hmr
```

依赖就安装完毕了。

如果我们还要再来一个在页面上直接显示catch到的错误的transform，（不想打开console看到底有啥错误，直接显示在页面上多好），简单！再安装一个transform:

```
npm install --save-dev react-transform-catch-errors redbox-react
```

依赖安装完毕，配置Babel，上面说到把Babel的配置都写在webpack.config.js中，这是一个不好

在跟目录新建一个Babel的配置文件: **.babelrc**, 把原来的配置写进去。

```
{
  "presets": ["react", "es2015"]
}
```

现在在webpack里面的config就可以简化了，把那些query参数都删掉，简单了很多：

```
...
  module: {
    loaders: [
      {
        test: /\.jsx?$/,
        loader: 'babel',
        include: APP_PATH
    }]
  },
...
```

要让新建的两个transform生效,只需再安装一个present。

```
npm install babel-preset-react-hmre --save-dev
```

安装完毕，将支持HMR和Catch Error的present添加到.babelrc

```
{
  "presets": ["react", "es2015"],
  //在开发的时候才启用HMR和Catch Error
  "env": {
   "development": {
     "presets": ["react-hmre"]
   }
  }
}
```

配置完毕！ 启动npm run dev

看看效果。然后随便改动h1标题里面的文字，发现页面没有刷新，但是自动内容自动改变了，在render方法中故意弄一些错误，出现了可爱的红色错误提示，大功告成～~~

继续项目

完美配置了Webpack和React的开发环境，现在让我们把小项目完成吧。

根据上面的图 把项目分为两个主要的component，一个是Search Box用来让用户填写用户名， 一个是List，用来展示搜索到用户的列表.

Search Box非常的简单 就是两个input，当用户点击Search的时候，把输入的名字发送到List的组件里面。

```
import React from 'react';
import ReactDOM from 'react-dom';
export default class Search extends React.Component {
  constructor(props) {
    super(props);
    this.handleSearch = this.handleSearch.bind(this);
  }
  handleSearch() {
     let name = ReactDOM.findDOMNode(this.refs.name).value;
     if (name === '') {
      return;
    }
    this.props.sendAction(name);
  }
  render() {
    return (
      <div>
        <input type="text" ref="name" placeholder="enter the name you wanna search"/>
        <button onClick={this.handleSearch}>Search</button>
      </div>
     )
  }
}
```

的时候，发起ajax请求，并且显示一个loading的提示。 完成状态，当请求完毕，渲染列表并且显示出来。

这个列表有不同的生命周期，这里简单介绍一下React Component的生命周期.

初始化的生命周期

如上图所描述的一样，当实例化一个Component的时候，会依次调用这些方法，所以在render完后做什么操作的话，可以把代码放到componentDidMount中。

当props属性发生变化以后的生命周期

如上图描述，当这个组件的props发送变化以后，会依次调用这些方法，当大体清楚了整个lifeCycle，是时候继续列表组件了

```
//自定义了一个ajax的方法，非常简单，支持promise

import {get} from '../utils/ajax';

export default class Plist extends React.Component {

  constructor(props) {
    super(props);
    this.state = {"loading":false, "list": []};
  }
  //当初次渲染完毕 设置该组件的属性firstView为true
  componentDidMount() {
    this.setState({"firstView": true});
  }
  //当传入的props有变化，请注意看上面第二张图，就是时候发起请求 更新列表的内容了
  componentWillReceiveProps(nextProps) {
    let keyword = nextProps.keyword;
    //loading设为true，firstView设为false
    this.setState({"loading": true, 'firstView': false});
    let url = `https://api.github.com/search/users?q=${keyword}`;
    //发起ajax请求
    get(url).then((data) => {
      //更新本组件的state
      this.setState({"loading":false, "list": data.items});
    }).catch((error) => {
      console.error(error);
    });
  }

  render() {
    const imgStyle = {
      width: '50px'
    }
    //添加一些if else的判断，用来展示不同的内容
    if (this.state.firstView) {
      return (
        <h2>Enter name to search</h2>
      )
    }
    if (this.state.loading) {
      return (
        <h2>Loading result...</h2>
      );
```

```
            <h2>No result.</h2>
          )
        } else {
          return (
            <div className="row">
              {this.state.list.map(people=>{
                return (
                  <div className="card">
                    <img src={people.avatar_url} style={imgStyle}/>
                    <p className="card-text">
                      {people.login}
                    </p>
                  </div>
                )
              })}
            </div>
          );
        }
      }
    }
  }
```

最后把两个组件引用到App里面

```
import '../node_modules/bootstrap/scss/bootstrap.scss';
import React from 'react';
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 (
```

```
          <h3 className="jumbotron-heading">Search Github Users</h3>

          <Search sendAction={this.refreshKeyword}/>
        </section>

        <Plist keyword={this.state.keyword}/>
      </div>
    );
  }
}
```

大功告成！现在让webpack跑起来，看看成果。随便搜索一个关键词试一试:

没有搜索的初始化界面

搜索以后的结果

总结

这一节其实没有多少新的东西 经过前两节的学习 我们应该很容易自己就能做到React的配置了。这里我们使用一个小项目，更好的帮助大家了解。Webpack给React带来更多的便利。希望以后再做React的项目的时候，不妨采用Webpack来进行配合。由于篇幅的限制，这里没有介绍使用karma和webpack测试React Component，以后会专门介绍。

很硬的链接

这一节关于和React配合的代码单独的创建了一个Repo，如果你想使用webpack和React开始一个新项目，不妨可以把它当作一个Starter Kit。

vikingmute/webpack-react-kit: a simple webp...

同时这一系列关于Webpack的文章也可以在github找到

vikingmute/webpack-for-fools

欢迎关注我们的新浪微博：前端外刊评论

56 条评论

写下你的评论...

寸志

我是赞题图的

2 年前 1 赞

郁也风

可以把 babel 的配置转移到 package. json 里面，这样就不用在 webpack 里多次重复了

2 年前

尤雨溪

现在不是都用 react-transform 了么

2 年前 1 赞

寸志 回复 **尤雨溪** 查看对话

这是啥？

2 年前

通过 babel 插件对 React 组建进行热重载 instrumentation，这样只需要 babel-loader 即可

2 年前 1 赞

张轩（作者） 回复 **郁也风** 查看对话

谢谢建议 现在把babel配置移到了.babelrc文件中

2 年前

张轩（作者） 回复 **尤雨溪** 查看对话

谢谢大神提醒 已经更新了 去掉了React Hot Loader 采用了React Transform

2 年前 2 赞

蒋馨

ReactDOM.render(<app />, app);这句的<app /> 应该改为 <App /> 在我这边的环境会出问题的

1 年前 1 赞

张轩（作者） 回复 **蒋馨** 查看对话

嗯 忘记改了 在repo里面改了 这里忘改了 谢谢

1 年前

蒋馨

请教, 我卡在弄不出那个red-box出错提示了, 我把你的那个项目clone下来了,测试也不行, 测试方法是去掉个 ">" 只会在chrome console中出错误, 另外我安装了 chrome react plugin?

1 年前

下一页

文章被以下专栏收录

前端外刊评论

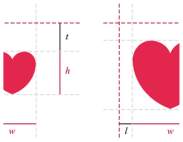
关注前端前沿技术，探寻业界深邃思想 qianduan.guru

进入专栏

React vs Angular 2: 冰与火之歌

最近前端圈撕得厉害，正能量的我们还是用干货来表达自己的态度吧～本文译自 Angular 2 versu… 查看全文 >

黄玄 · 1 年前 · 发表于 前端外刊评论



Twitter "like" 动画实战

原文1地址：Twitter’s Heart Animation in Full CSS原文2地址：How Did They Do That? The … 查看全文 >

kmokidd · 2 年前 · 发表于 前端外刊评论

罗德胤：哑铃社会与乡村遗产

最近创业者有一个特别热的词叫认知壁垒，现在我遇到的壁垒就是：乡村遗产处在一个大的社会结… 查看全文 >

清华同衡规划院 · 2 天前 · 编辑精选 · 发表于 规划论衡

Passerillage，甜在西元前

拿到这个选题，我内心是拒绝的。我知道这个单词，但却一直没弄明白是如何被生硬的翻译过来的… 查看全文 >

小酌日历 · 19 天前 · 编辑精选