

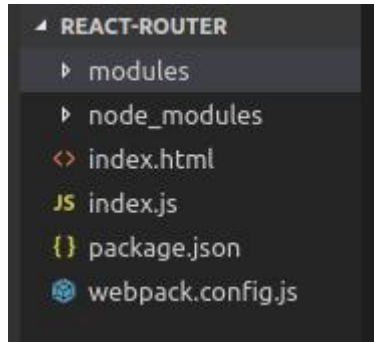
React-router (上) 学习笔记

参考网址：<http://www.mrfront.com/2016/12/11/react-router-tutorial-part1/>

1. 项目创建

首先创建文件夹 react-router

然后按照下面的结构创建文件夹和文件（不包括 node_modules）



Index.html 中实现如下内容：

Index 中有一个 div 标签，并且夹在编译后的文件 bundle.js, div 标签的内容在 index.js 中进行实现

```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
  <meta charset="UTF-8">
  <title>My First React Router App</title>
</head>
<body>
  <div id=app></div>

  <!-- 编译后的 js 文件 -->

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

<!-- 编译后打包的 bundle.js 所在的目录为：
react-router/node_modules/crypto-browserify/example
-->
```

package.json 文件的内容如下：

在 package.json 文件中加载相应的模块：包括 react, react-dom, react-router

```
{
  "name": "tutorial",
```

```

    "version": "1.0.0",
    "description": "",
    "main": "index.js",
    "scripts": {
      "start": "webpack-dev-server --inline --hot
--content-base ."
    },
    "author": "",
    "license": "ISC",
    "dependencies": {
      "react": "^0.14.7",
      "react-dom": "^0.14.7",
      "react-router": "^2.0.0"
    },
    "devDependencies": {
      "babel-core": "^6.5.1",
      "babel-loader": "^6.2.2",
      "babel-preset-es2015": "^6.5.0",
      "babel-preset-react": "^6.5.0",
      "http-server": "^0.8.5",
      "webpack": "^1.12.13",
      "webpack-dev-server": "^1.14.1"
    }
  }
}

```

Webpack.config.js 的内容如下：

```

module.exports = {
  // 入口文件
  entry: './index.js',

  // 输出文件
  output: {
    filename: 'bundle.js',
    publicPath: ''
  },

  module: {
    loaders: [

      // 匹配到 js 或 jsx 文件后 使用 babel-loader 来处理

      // '?'后面是该 loader 的参数设置(使用了 es6 和 react 转码器)
    ]
  }
}

```

```

    {
      test: /\.js[x]?$/,
      exclude: /node_modules/,
      loader:
'babel-loader?presets[]=es2015&presets[]=react'
    }
  ]
}
};

```

在 webpack.config.js 文件中，entry 里面是唯一入口文件，并且要注意文件的路径要以 ./ 开头，否则默认是模块，并且会到 node_modules 中去找，output 表明将编译后的文件打包成 bundle.js 以及放到对应的路径下。

下面这个文件，执行 \$npm run build 后最后会生成 bundle.js 并且放到 node_modules/crypto-browserify/example/bundle.js 中

```

module.exports = {
  // 入口文件
  entry: './index.js',

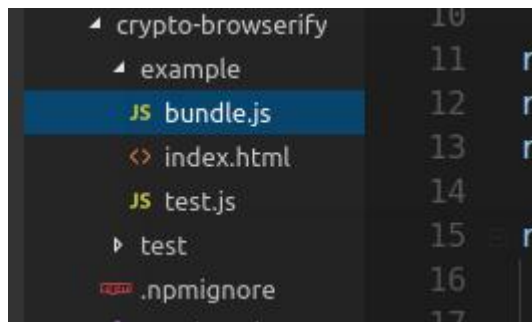
  // 输出文件
  output: {
    filename: 'bundle.js',
    publicPath: ''
  },

  module: {
    loaders: [

      // 匹配到 js 或 jsx 文件后 使用 babel-loader 来处理

      // '?'后面是该 loader 的参数设置(使用了 es6 和 react 转码器)
      {
        test: /\.js[x]?$/,
        exclude: /node_modules/,
        loader:
'babel-loader?presets[]=es2015&presets[]=react'
      }
    ]
  }
};

```



Index.js 的内容如下：

Index.js 表明将<App />组件的内容挂载到 id 为 app 的父节点上面，也就是我们在 index.html 中的<div id=' app' ></app>

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './modules/App';

ReactDOM.render(
  <App/>,
  document.getElementById('app')
);
```

在 modules 中新建文件 App.js，并且在 App.js 中定义一个组件
该组件中的 JSX 对象 render 会在页面上显示一个标题
export default 定义了模块对外接口

```
import React, {Component} from 'react';
// 定义 App 组件
export default class App extends Component {
  constructor(props) {
    super(props);
  }
  render() {
    return (
      <h1>React Router Tutorial</h1>
    );
  }
}
```

关于上面代码的 React, {Component}：

import {Component} from 'react' 表达的意思

<https://segmentfault.com/q/1010000007539016>

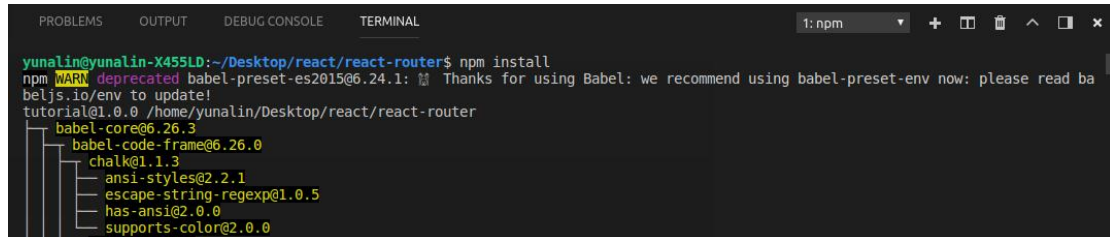
Import React from 'react' 是从 react 模块中引入 react 类，但是{Component}是从 react

中进行解构赋值

{Component} 实际上是一个对象，将 react 对象中的 Component 属性赋值给 {Component} 对象中的 Component，前提是名称必须要相同，及 React.Component 中的 Component 名称要和 {Component} 中的名称相同，变量和属性的名称要保证相同

使用 vsCode 对代码进行编辑，用 ctrl+` 调出终端

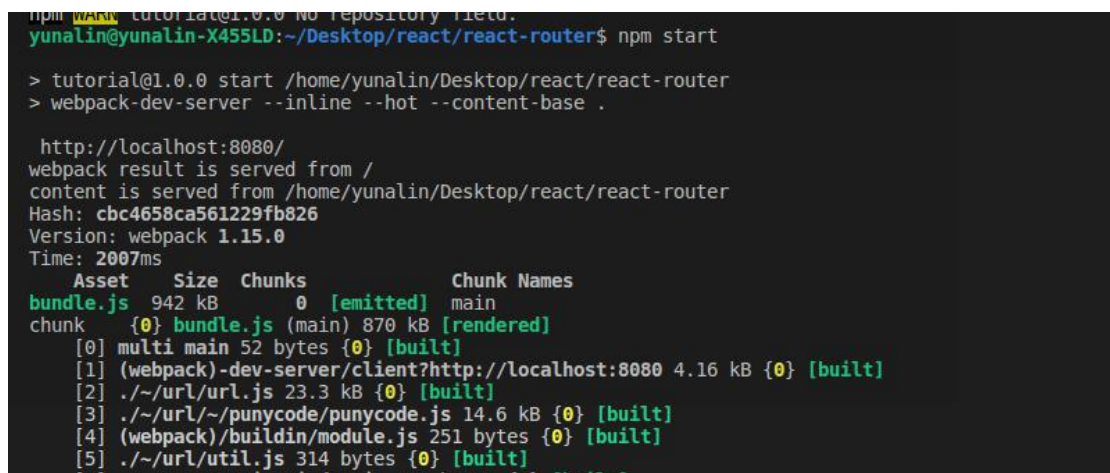
\$npm install #加载 package.json 中的依赖并且生成 node_modules 文件



```
yunalin@yunalin-X455LD:~/Desktop/react/react-router$ npm install
npm WARN deprecated babel-preset-es2015@6.24.1: Thanks for using Babel: we recommend using babel-preset-env now: please read babeljs.io/env to update!
tutorial@1.0.0 /home/yunalin/Desktop/react/react-router
├─┬ babel-core@6.26.3
│ └─┬ babel-code-frame@6.26.0
│   ├── chalk@1.1.3
│   ├── ansi-styles@2.2.1
│   ├── escape-string-regexp@1.0.5
│   ├── has-ansi@2.0.0
│   └── supports-color@2.0.0
```

\$npm run build #对编译后的文件进行打包

\$npm start #启动项目，在浏览器打开可以看到页面



```
yunalin@yunalin-X455LD:~/Desktop/react/react-router$ npm start
> tutorial@1.0.0 start /home/yunalin/Desktop/react/react-router
> webpack-dev-server --inline --hot --content-base .

http://localhost:8080/
webpack result is served from /
content is served from /home/yunalin/Desktop/react/react-router
Hash: cbc4658ca561229fb826
Version: webpack 1.15.0
Time: 2007ms
   Asset      Size  Chunks             Chunk Names
bundle.js  942 kB          0  [emitted]  main
chunk       {0} bundle.js (main) 870 kB [rendered]
   [0] multi main 52 bytes {0} [built]
   [1] (webpack)-dev-server/client?http://localhost:8080 4.16 kB {0} [built]
   [2] ./~/url/url.js 23.3 kB {0} [built]
   [3] ./~/url/~punycode/punycode.js 14.6 kB {0} [built]
   [4] (webpack)/buildin/module.js 251 bytes {0} [built]
   [5] ./~/url/util.js 314 bytes {0} [built]
```

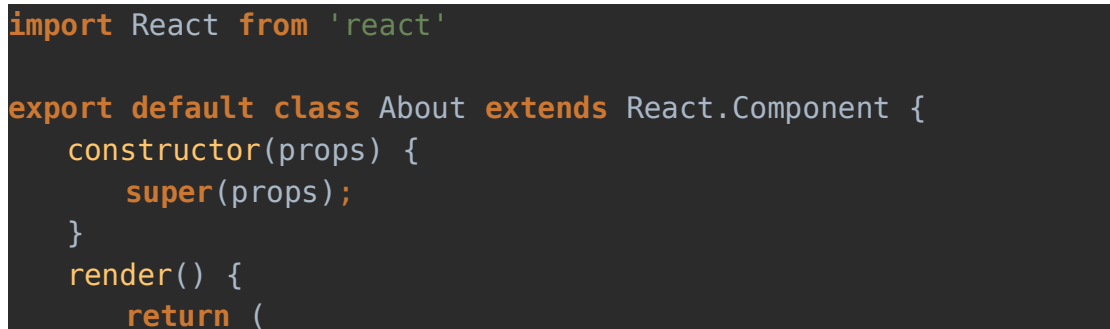
http://localhost:8080/

React Router Tutorial

3. 创建一个简单的路由

(1) 首先在 modules 中创建 About.js 和 Repos.js，创建两个新的组件
内容分别为：

About.js



```
import React from 'react'

export default class About extends React.Component {
  constructor(props) {
    super(props);
  }
  render() {
    return (
```

```

        <div>About</div>
    );
}
};

```

Repos.js

```

import React from 'react';

export default class Repos extends React.Component {
    constructor(props) {
        super(props);
    }

    render() {
        return (
            <div>Repos</div>
        );
    }
}

```

修改 index.js 的内容如下，添加路由

```

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

// index.js 和 modules 在同级目录下

import App from './modules/App';
import About from './modules/About';
import Repos from './modules/Repos';
import {Router, Route, hashHistory} from 'react-router'

// ReactDOM.render(
//     <About/>,
//     document.getElementById('app')
// );
ReactDOM.render((
    <Router history={hashHistory}>
        <Route path='/' component={App} />
        <Route path='/about' component={About} />
        <Route path='/repos' component={Repos} />
    </Router>
), document.getElementById('app'))
);

```

重新启动项目，在浏览器打开 localhost:8080/#/ 显示的是 App 页面，打开 localhost:8080/#/about 显示的是 About 页面，同理可以显示 Repos 页面

后面的#是hashHistory在起作用，path属性值定义了url中#之后的路径参数

4. 导航链接 link 组件

Link 组件等同于[<a/>](#)标签

修改 index.js 的内容如下：

首先从 react-router 中引入 Link，然后添加导航

实现页面的跳转
注意：render 函数返回的对象有多个标签的话要用 div 标签包含起来

```
import React, {Component} from 'react';
import {Link} from 'react-router';
// 定义 App 组件
export default class App extends Component {
  // 这个构造函数是为了后面能够调用 this.props
  constructor(props) {
    super(props);
  }
  // 添加 Link 组件链接
  render() {
    return (
      <div>
        <h1>React Router Tutorial</h1>
        <ul role='nav'>
          <li><Link to='/about'>About</Link></li>
          <li><Link to='/repos'>Repos</Link></li>
        </ul>
      </div>
    );
  }
}
```

打开浏览器，得到下面的界面，并且点击 About 和 Repos 可以实现页面的跳转



React Router Tutorial

- [About](#)
- [Repos](#)

5 · 嵌套路由

将 index.js 中的

```
<Route path="/" component={App} />
<Route path="/about" component={About} />
<Route path="/repos" component={Repos} />
```

修改为：

```
{/* 嵌套路由 */}

<Route path="/" component={App}>
  <Route path="/about" component={About} />
  <Route path="/repos" component={Repos} />
</Route>
```

然后修改 App.js 中内容，让 About 和 Repos 作为 App 的子组件显示在 App 里面

```
render() {
  return (
    <div>
      <h1>React Router Tutorial</h1>
      <ul role='nav'>
        <li><Link to='/about'>About</Link></li>
        <li><Link to='/repos'>Repos</Link></li>
      </ul>
      {/* 嵌套路由，将子路由的组件内容显示在 App 组件中 */}
      <div>{this.props.children}</div>
    </div>
  );
}
```

运行结果如下：

