## **react router 4.0以上的路由应用**

安装：

npm install react-router-dom --save

## **4.0有HashRouter和BrowserRouter**

### HashRouter

如果你使用过react-router2或3或者vue-router，你经常会发现一个现象就是url中会有个#，例如localhost:3000/#，HashRouter就会出现这种情况，它是通过hash值来对路由进行控制。如果你使用HashRouter，你的路由就会默认有这个#。

### BrowserRouter

很多情况下我们则不是这种情况，我们不需要这个#，因为它看起来很怪，这时我们就需要用到BrowserRouter。

**improt**

import { BrowserRouter, Route, Link, Switch } from 'react-router-dom';

或者：

import { HashRouter, Route, Link, Switch } from 'react-router-dom';

**示例**

在4.0以下的react router中，嵌套的路由可以放在一个router标签中  
<Route component={App}>

<Route path="groups" components={Groups} />

<Route path="users" components={Users}>

<Route path="users/:userId" component={Profile} />

</Route>

</Route>

但是在4.0以后，嵌套的路由与之前的就完全不同了

正确形式如下

<Route component={App}>

<Route path="groups" components={Groups} />

<Route path="users" components={Users}>

//<Route path="users/:userId" component={Profile} />

</Route>

</Route>

上面将嵌套的路由注释掉

const Users = ({ match }) => (

<div>

<h2>Topics</h2>

<Route path={`${match.url}/:userId`} component={Profile}/>

</div>

)

## **一个完整的嵌套路由的例子如下**

### **说明及注意事项**

1. 以下代码采用ES6格式
2. react-router-dom版本为4.1.1
3. 请注意使用诸如HashRouter之类的history，否则一直会有warning，不能渲染

import React, { Component } from 'react';

import ReactDOM from 'react-dom';

import {

HashRouter,

Route,

Link,

Switch

} from 'react-router-dom';

class App extends Component {

render() {

return (

<div>

<h1>App</h1>

<ul>

<li><Link to="/">Home</Link></li>

<li><Link to="/about">About</Link></li>

<li><Link to="/inbox">Inbox</Link></li>

</ul>

{this.props.children}

</div>

);

}

}

const About = () => (

<div>

<h3>About</h3>

</div>

)

const Home = () => (

<div>

<h3>Home</h3>

</div>

)

const Message = ({ match }) => (

<div>

<h3>new messages</h3>

<h3>{match.params.id}</h3>

</div>

)

const Inbox = ({ match }) => (

<div>

<h2>Topics</h2>

<Route path={`${match.url}/messages/:id`} component={Message}/>

</div>

)

ReactDOM.render(

(<HashRouter>

<App>

// 如果匹配，Home组件会被加载

<Route exact path="/" component={Home} />

// 如果匹配，About组件会被加载

<Route path="/about" component={About} />

// 如果匹配，Inbox组件会被加载

<Route path="/inbox" component={Inbox} />

</App>

</HashRouter>),

document.getElementById('root')

);

当访问/About时。路由/与/About都能匹配，但因为<Route exact path="/" component={Home} />使用了exact为严格匹配，所以只有About组件被加载

## **理解和使用Route**

无论何时你需要在匹配某个路径的时候绘制一个组件，那么就可以使用Route组件。

Route组件可以使用如下的属性：

* path属性，字符串类型，它的值就是用来匹配url的。
* component属性，它的值是一个组件。在path匹配成功之后会绘制这个组件。
* exact属性，这个属性用来指明这个路由是不是排他的匹配。
* strict属性， 这个属性指明路径只匹配以斜线结尾的路径。

使用组件：

<Route exact path="/" component={HomePage} />

使用render属性实现内联绘制：

<Route path="/" render={()=><div>HomePage</div>} />

## **URL / Path / Route的参数**

通常情况下，我们都会在路径里添加参数。这样方便在不同的组件之间传递一些必要的数据。那么我们如何才能获取到这些传递的参数，并传递给组件中呢？我们只需要在路径的最后加上/:param，参数可为空则/:param？。如：

<Route path="/:param1" component={HomePage} />  
<Route path='/index/:hello?' />

export default class SideNav extends React.Component

{

constructor(props){

super(props);

this.menu = props.match.params.param1;

}

.......

}

一旦有路径可以匹配成功，那么就会穿件一个拥有如下属性的对象，并传入绘制的组件里：

* url: 匹配的url。
* path：就是path。
* isExact：如果path和当前的widnow.location的path部分完全相同的话。
* params：在URL里包含的参数。

## **理解并使用Link**

使用Link可以在React应用的不同页面之间跳转。Link只会重新加载页面里和当前url可以匹配的部分。

Link组件需要用到to属性，这个属性的值就是react router要跳转到的地址。如：

import { Link } from 'react-router-dom';

const Nav = () => (

<Link to='/'>Home</Link>

);

当被点击的时候，会跳转到路径：/。

## **<Link>和<NavLink>**

NavLink是Link的一个子类，在Link组件的基础上增加了绘制组件的样式，比如：

<NavLink to="/me" activeStyle={{SomeStyle}} activeClassName="selected">

My Profile

</NavLink>

## **理解排他路由**

排他路由是react router v3的默认实现。只有第一个匹配的路由对应的组件会被绘制。这一点也可以用react router v4的Switch组件来实现。

import { Switch, Route } from 'react-router';

<Switch>

<Route exact path="/" component={HomePage} />

<Route path="/about" component={AboutPage} />

<Route path="me" component={ProfilePage} />

<Route component={NotFound} />

</Switch>

## **使用<Redirect>组件实现重定向**

无论何时你要重定向到另外一个地址的时候，都可以使用Redirect组件：

<Redirect to="register" />

**关于默认路由**

<Switch>

<Route exact path="/ManageHome/OAuthManage" component={Client} />

<Route exact path="/ManageHome/OAuthManage/Client" component={Client} />

</Switch>

**[控制路由跳转](https://segmentfault.com/a/1190000011137828)**

### 使用 withRouter

withRouter高阶组件，提供了history让你使用~

**import** React **from** "react";

**import** {withRouter} **from** "react-router-dom";

**class** **MyComponent** **extends** **React**.**Component** {

...

myFunction() {

**this**.props.history.push("/some/Path");

}

...

}

**export** **default** withRouter(MyComponent);

这是官方推荐做法哦。