React-router4 分享 (一)

2019年6月14日 星期五　　下午2:29

常用组件库

- **React-helmet** (helmet :['hɛlmɪt] 头盔) 管理对文档头的所有更改

添加meta title 等标签，类似模块：react-document-title （只能修改标题）

```
<Helmet>
    <meta charSet='utf-8' />
    <title>{title}</title>
    <meta name='keywords' content={title} />
</Helmet>
```

- **Fragment** （ 碎片）

在 React 对象中，Fragment 可以让你聚合一个子元素列表，并且不在DOM中增加额外节点

可以理解为就是一个空标签， 例如：组件拼接的 table tr td ，dt dd li 等 不适合用div包起来

```
<React.Fragment key={item.id}>
  <dt>{item.term}</dt>
  <dd>{item.description}</dd>
```

```
</React.Fragment>
```

常用功能

- ## React-router & react-router-dom

v4版本将路由进行了拆分，分成了两个模块，拆分后 更容易理解，可以将 Route 就当做 component 组件一样使用，当path匹配时 渲染Route对应的组件内容

区别：

React-router：实现了路由的核心功能，提供了router的核心api。如Router、Route、Switch等

React-router-dom： **基于react-router**，提供了浏览器功能和dom操作相关的一些路由BrowserRouter、Route、Link等，BrowserRouter使用 HTML5 提供的 history API可以保证你的 UI 界面和 URL 保持同步

调用：

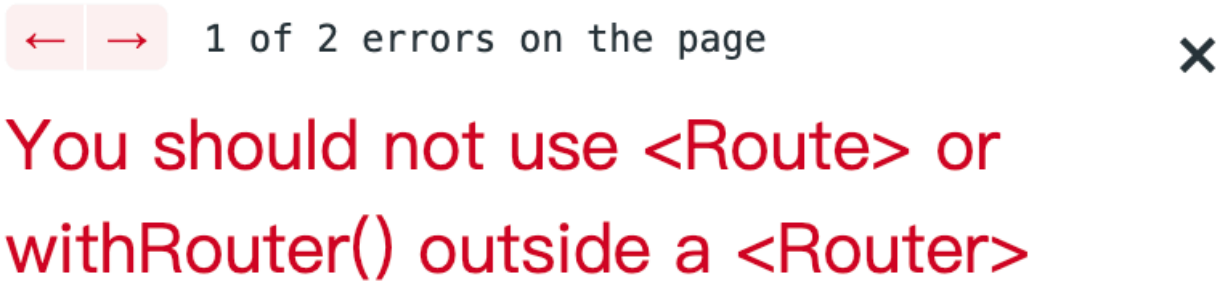
import {Swtich, Route, Router, HashHistory, Link} from 'react-router-dom';

注：react-router-dom是依赖react-router的，只需安装前者，一般只需引入react-router-dom 就可以

Router组件：V4以前Router子元素只能是路由相关的组件，V4以后各种组件都可以

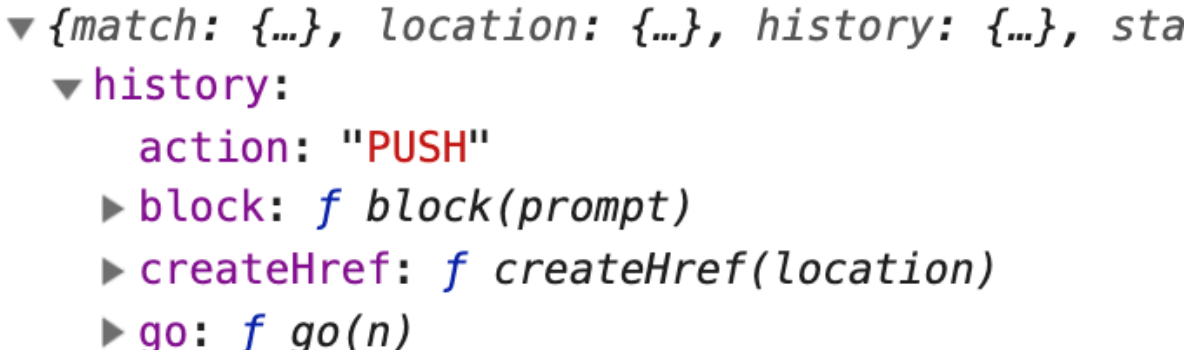
注：Router只允许存在一个子元素，存在多个会报错

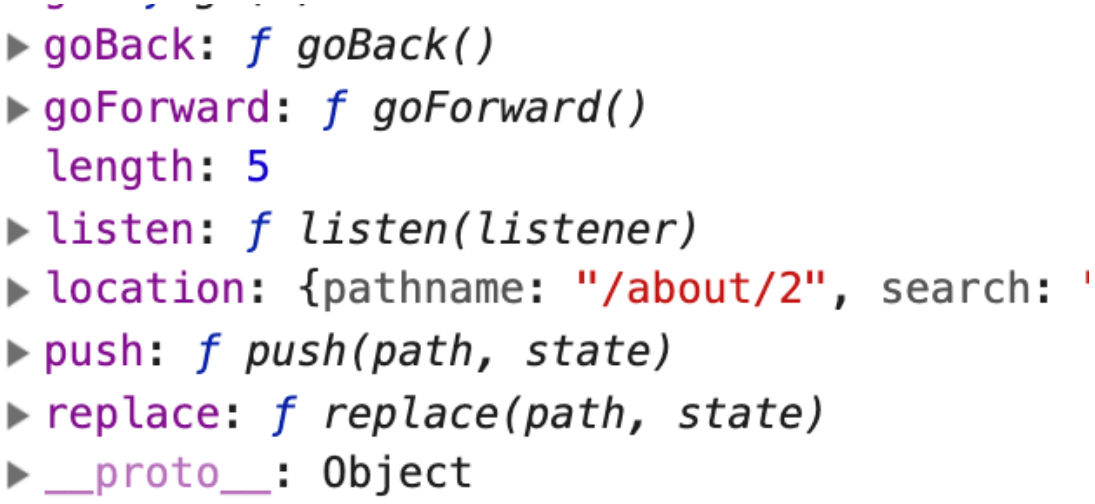
注：Route标签必须写在Router里，否则会报错



BrowserRouter

history 常用方法（参考代码实例）：





注：BrowserRouter 常用属性 basename，设置所有location的基本url，一般应用为子目录时使用（参考代码实例 index.js）

- **switch 和 exact**

V4以前：路由匹配是独一无二的；
v4以后：路由有包含关系，（例： `path="/list"` 的路由会匹配 `path="/"`的路由）

使用switch，只有一个路由会被渲染，且渲染第一个被匹配到的组件

V4 新增了exact关键字，表示只对当前的路由精确匹配

一般在路由的最后加一行 `<Redirect to='/' />` 路由匹配不到时，主动跳转配置的路由
或者 自定义跳转，不添加path `<Route component={NoMatch}></Route>`

（参考代码实例App.js ）

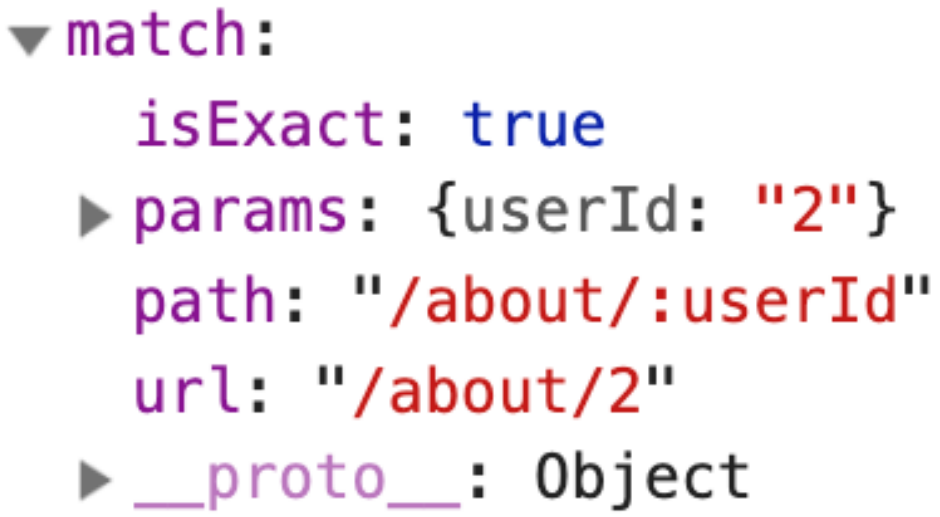
- ## Link & NavLink

  链接组件，会渲染成a标签 (参考代码实例)
  引入：`import {Link,NavLink} from 'react-router-dom';`
  区别：`NavLink`可以给当前选中的路由添加样式 `activeStyle`和 `activeClassName`

- ## Url 参数 & 查询参数 & 隐式传参

  url参数 使用方法：`<Route path="/about/:userId" component={AboutPage}/>`

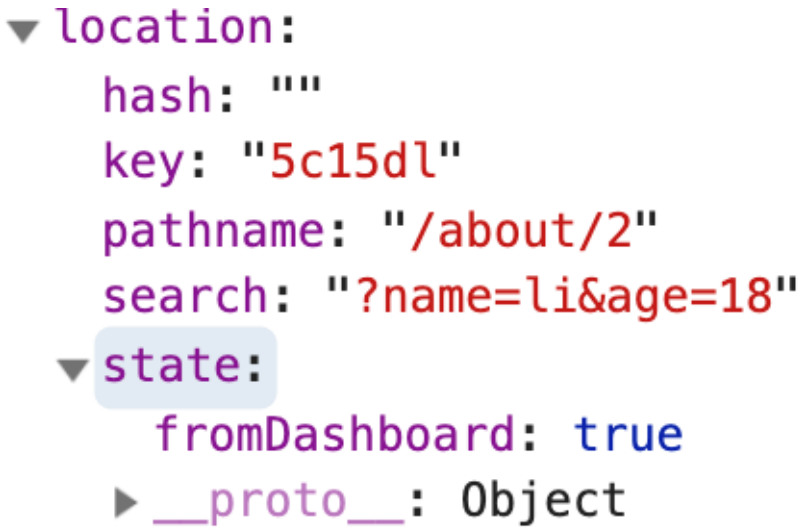
  url参数存到了match对象的 params属性中（参考代码实例 App.js定义，indexPage.js 传值）



查询参数（ search ）传值 ：

```
this.props.history.push({
        pathname: '/about/2',
        search:'?name=li&age=18',
        state: { fromDashboard: true }
})
```

查询参数（ search ）调用：this.props.history.location.search



search的格式不是json结构，需要自行处理，推荐使用 query-string ，跟Node里的是一样的。

- **编程式导航 withRouter**

默认情况下必须是经过路由匹配渲染的组件才拥有路由参数（ histroy、location、match ）this.props.history...

作用：**把不是通过路由切换过来的组件中，将react-router 的**

history、location、match 三个对象传入props对象上

使用方法：export default withRouter(App)

结合redux的使用方法：withRouter(connect(...)(App))

注：withRouter解决更新问题的时候，一定要保证withRouter在最外层

- ## 滚动恢复

V4没有提供默认的滚动管理，滚动恢复效果参考代码实例（App.js ，滚动恢复的其中一种实现，也可以通过`componentWillReceiveProps` 判断location.pathname是否一致，然后添加方法window.scrollTo(0,0)）

- ## React-loadable

实现动态加载组件的高阶组件动态加载，代码分割，并且可以添加加载动画，实现预加载等功能

用法：

```
import Loadable from 'react-loadable';
import Loading from './my-loading-component';
const LoadableComponent = Loadable({
    loader: () => import('./my-component'),
    loading: Loading,
    delay: 300,
    timeout:1000
```

```
    timeout:1000

});
export default class App extends
React.Component {
    render() {
        return <LoadableComponent/>;
    }
}
```

预加载：LoadableComponent .preload（ ）

Loading中获取:
错误状态：props.err
判断超时：props.timedOut

参考文章：https://www.jianshu.com/p/462bb9d1c982

文档：https://reacttraining.com/react-router/web/guides/quick-start

- **connected-react-router**

router结合redux，内容较多，下次分享