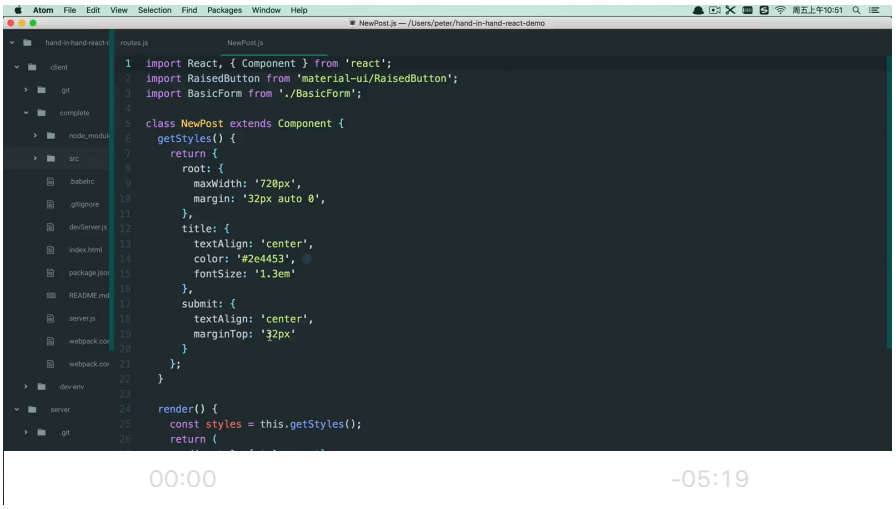

前端导航栏体现用户登录状态

前端实现用户退出登录功能

第六章：用户注册功能实现

前端实现写文章功能

前面课程中我们已经介绍了如何通过 Redux 管理用户登录状态，本节课程我们将在 Redux Store 中添加一个新的 `posts` state，用来存储案例中的所有文章，当提交添加新文章表单之后，调用异步 Action 创建函数 `newPost`，向后端请求添加新文章接口，前端等待请求响应之后，会分发一个 `ADD_POST` action，进而触发相应的 `posts` reducer 更新 store 中存储的 `posts` state，下面我们就一步步演示这个过程，再一次熟悉一下 Redux 的核心概念。

实现异步 Action 创建函数 newPost

新建 `src/redux/actions/postActions.js` 文件，

```
import axios from 'axios';
import { browserHistory } from 'react-router'
import { Settings } from '../../settings';

function handleError(error) {
  if (error.response) {
    console.log(error.response.data.error);
  } else {
    console.log(error);
  }
}

export function newPost(data) {
  return function(dispatch) {
    axios.post(`${Settings.host}/posts`, data
      headers: {'Authorization': sessionStora
    }).then(response => {
      dispatch({ type: 'ADD_POST', post: resp
      browserHistory.push('/dashboard');
      console.log(response.data.message)
    }).catch(error => {
      handleError(error);
    });
  }
}
```

当向服务器请求 `/posts` 接口的时候，需要添加 HTTP 头标识 `Authorization`，才能访问服务器端受保护的资源。当请求响应成功之后，才会分发 `ADD_POST` action，从发送请求到响应请求，需要经历一段时间，所以这个 `Add_POST` action 是一个异步 action

定义与 posts 状态相关联的 reducer

新建文件 `src/redux/reducers/posts.js`，添加代码

```
export default (state = [], action = {}) => {
  switch(action.type) {
    case 'ADD_POST':
      return [...state, action.post]
    default:
      return state
  }
}
```

上述代码定义了一个匿名的 reducer，它将负责更改 `posts` state。

合并到 root reducer

打开文件 `src/redux/rootReducer.js`，导入刚才定义的匿名 reducer 为 `posts`：

```
import posts from './reducers/posts';
```

然后把 `posts` reducer 与 `auth` reducer 合并为一个根级 reducer:

```
export default combineReducers({
  auth,
  posts
});
```

NewPost 组件中调用 newPost

首先，从 React 绑定库中导入 `connect` 方法：

```
import { connect } from 'react-redux';
```

然后，导入 `newPost` action 创建函数：

```
import { newPost } from '../../redux/actions/
```

接下来，给表单添加 `onSubmit` 事件属性，给 BasicForm 组件添加 `ref` 属性，以此获得 BasicForm 组件中各个输入框中的文字：

```
<form onSubmit={ this.handleSubmit.bind(this)
  <BasicForm ref='basic' />
  ...
</form>
```

定义表单的 `onSubmit` 事件处理函数：

```
handleSubmit(e) {
  e.preventDefault();
  const basic = this.refs.basic.getBasicForm(
  this.props.newPost(basic);
}
```

最后，使用 `connect` 方法把 `newPost` action 创建函数注入到 NewPost 组件中：

```
export default connect(null, { newPost })(New
```

修改 BasicForm 组件

打开文件 `ui/posts/BasicForm.js`，定义一个新的 `getBasicFormInputValue` 方法，获取输入框中的文字，组合成一个普通的 JS 对象作为返回值

```
getBasicFormInputValue() {
  const name = this.refs.name.getValue();
  const content = this.refs.content.getValue(
  return { name, content }
}
```