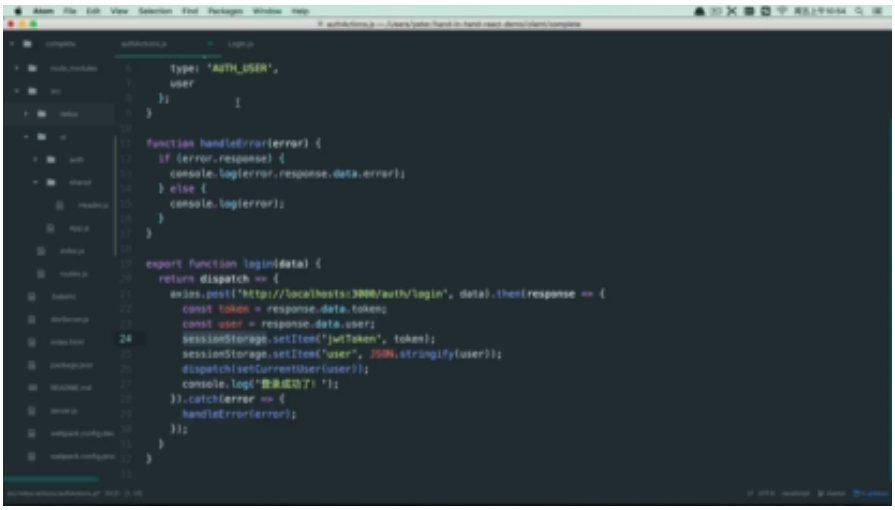

React 组件连接 React Store 分发 Redux Action

LogIn 组件中调用 login

在 `ui/auth/LogIn.js` 文件中，首先从 React 绑定库 `react-redux` 中导入 connect 方法：

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

connect 方法会检索由 React 绑定库的另一个接口 `<Provider store>` 传递给各级子组件的 `store` 属性值，然后给 React 组件注入 store 中的 state 以及 action creator。

然后，导入上节课程中定义的 action 创建函数 `login`：

```
import { login } from '../../redux/actions/a
```

接下来，检验传入的 `login` 属性的类型，确保其类型是一个函数类型：

```
LogIn.propTypes = {
  login: React.PropTypes.func.isRequired
}
```

React 类型检查

然后，调用 `connect` 方法把 action 创建函数 `login` 作为一个属性值注入到 LogIn 组件：

```
export default connect(null, { login })(Radi
```

因为 LogIn 组件不需要访问 store 中的 state，所以传给 `connect` 方法的第一个参数为 `null`。

最后，修改 `handleSubmit(event)` 事件处理方法，删除与 `axios` 相关的代码，使用：

```
handleSubmit(event) {
  ...
  this.props.login({username, password});
}
```

到谷歌浏览器中测试，若成功则在浏览器开发者工具的 Application 标签下的 Session Storage 条目中，看到上节课程中设置的 jwtToken 和 currentUser 两个键值对。

添加配置文件

新建文件 src/settings.js，添加代码：

```
export const Settings = {
  host: 'http://localhost:3000'
}
```

修改 API 地址

打开 src/redux/actions/authAction，导入配置文件：

```
import { Settings } from '../../settings';
```

修改登录接口：

```
axios.post(`${Settings.host}/api/auth`, data)
```

用户登录成功后切换到首页

打开 src/redux/actions/authAction 文件，导入 browserHistory 模块：

```
import { browserHistory } from 'react-router'
```

然后，调用 browserHistory 提供的 push 接口，切换页面到首页，添加一行代码：

```
axios.post(`${Settings.host}/auth/login`, {
  ...
  browserHistory.push(`/`);
  console.log('登录成功了！')
})
```

参考 React Router 文档 组件之外导航