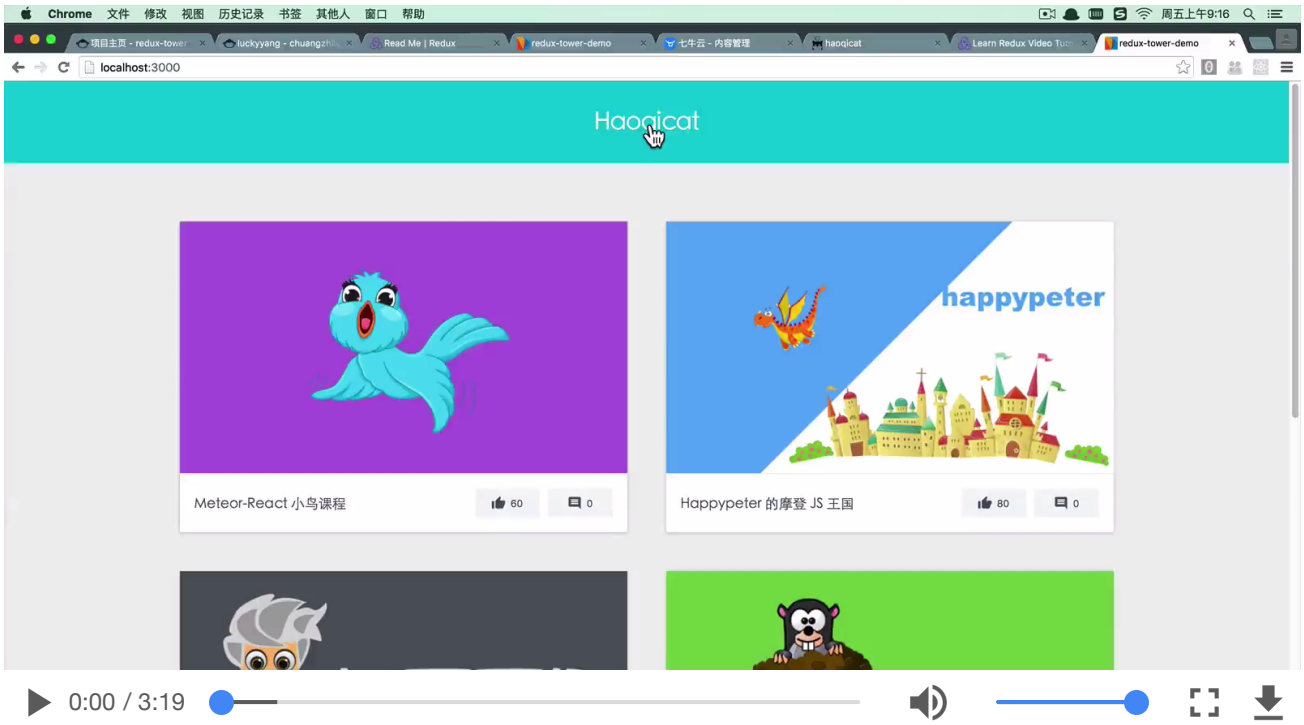

Redux 核心概念之 Store

在上节视频中，我们已经给每门课程添加了点赞功能。如果我们在应用首页为一门课程加赞之后，再访问课程详情页会发现课程赞数又恢复到最初的数目，显然这种业务逻辑是不正确的，我们期望课程详情页和首页的课程赞数一样。造成这一问题的原因是显而易见的，因为在首页为课程点赞之后，并没有更新存储在静态文件 `data/courses.js`

中的课程数据，当访问课程详情页的时候，又会重新从 `data/courses.js` 中读取数据，所以课程赞数自然不会改变。那我们该如何更新课程信息，并让其它使用相同数据的组件状态保持一致呢？那就有请大名鼎鼎的 Redux 闪亮登场吧，Redux 会为您排忧解难，小主再也不用担心了。

Redux 是天生用来管理应用 state 的，它有一个 store 容器来储存应用中的 state，任何组件都可以从 `store` 中读取 `state`。本项目中的课程信息就非常适合作为一个 `state` 储存到 `store` 中，然后各个组件之间的课程信息就能同步了。

从本节课程开始，我们就要介绍 Redux 的核心概念了，首先从 `store` 开始，下面我们就创建一个 `store`。

创建 store

新建一个文件 `src/store.js` 文件，先添加一行代码：

```
import { createStore, compose } from 'redux';
```

从 `redux` 中分别导入 createStore 模块 和 compose 模块。其中 `createStore` 这个接口就是用来创建 Redux `store` 容器的，而通过 `compose` 可以在项目中使用 redux-devtools-extension 调试工具，后续视频会介绍。

然后再添加两行代码：

```
import { syncHistoryWithStore} from 'react-router-redux';
import { browserHistory } from 'react-router';
```

react-router-redux 包并不是 Redux 项目必须的，用它则可以在 `redux-devtools-extension` 中显示页面浏览历史，不用它 Redux 和 React Router 也可以在一起完美工作。

```
import rootReducer from './reducers/index';
```

`rootReducer` 构建了一个完整的状态树（state tree），这就涉及到 Redux 的另一个核心概念 reducer，后面视频会详细讲述。现在，这个 `./reducers/index.js` 文件还没有创建。

```
import comments from './data/comments';
import courses from './data/courses';

const defaultState = {
  courses: courses,
  comments: comments
};

const store = createStore(rootReducer, defaultState);
```

导入在静态文件中存储的课程信息和课程的评论信息，然后用它们来定义一个对象常量 `defaultState`，接下来使用 `createStore` 接口创建一个 `store`，`createStore` 接口的第一个参数是 `rootReducer`，也就是构建的状态树，第二个参数就是状态树的初始值。所谓的状态树就是一个普通的 JS 对象，本项目要构建的状态树其实是这样的：

```
{
  courses: courses,
  comments: comments
}
```

自定义了两个状态变量 `courses` 和 `comments`，其初始值分别是从文件 `./data/courses` 和 `./data/courses` 中导入的数据。这样，我们就把课程信息和评论信息存储到 `store` 容器中了。

接下来，就是调用 syncHistoryWithStore 把路由状态（routing state）信息存储到新创建的 `store` 容器内。

```
export const history = syncHistoryWithStore(browserHistory, store);
```

最后再添加一行代码，把我们刚才创建的 `store` 容器导出去，供外部组件使用。

```
export default store;
```

注意：Redux 的三原则之一：单一数据源保证在一个 Redux 应用中只有唯一的一个 `store` 用于储存应用中唯一的一个状态树。