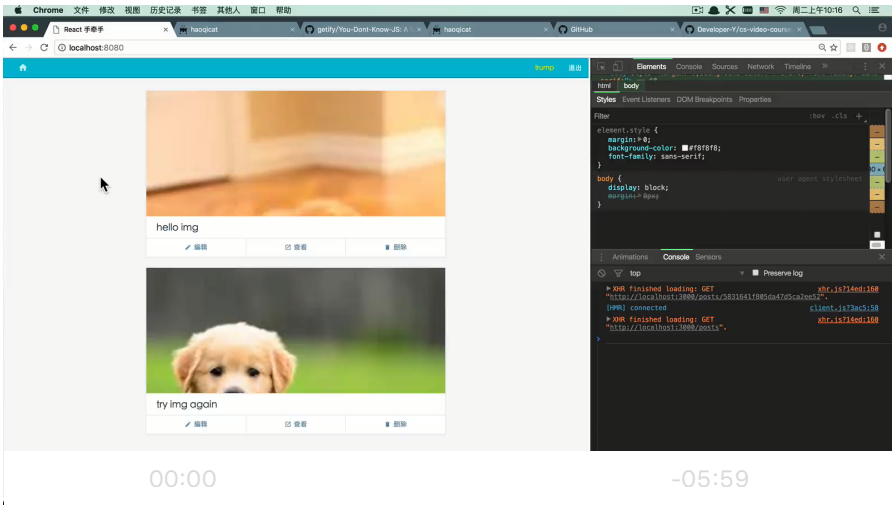

借助 Redux Thunk 实现异步 Action Creator

React 组件连接 React Store 分发 Redux Action

前端导航栏体现用户登录状态

完成更新文章功能

编辑文章的时候，表单中的数据来自 Redux store 中的 post 状态，EditPost 组件挂载之前会更新 post 状态，但是 TextField 组件的 `defaultValue` 的属性值不随着 `this.props.post` 的变化而变化，当更新下一篇文章的时候，导致表单中的数据显示的是上一篇文章的数据，而不是要更新的文章数据。因此，在 EditPost 组件和 ShowPost 组件卸载之前，应该清空 store 中的 post 状态值。

定义一个 clearPost 创建函数

修改 `redux/actions/postActions.js` 文件，定义一个 action 创建函数：

```
export function clearPost() {
  return { type: 'CLEAR_POST' };
}
```

`clearPost` 创建函数直接返回一个 action，所以说它是一个同步的 action 创建函数。

修改 post reducer

打开 `redux/reducers/post.js` 文件，添加处理 `CLEAR_POST` action 的代码：

```
switch(action.type) {
  ...
  case 'CLEAR_POST':
    return {};
  ...
}
```

EditPost 组件中调用 clearPost 创建函数

修改 EditPost 组件，导入 `clearPost` 创建函数：

```
import {... clearPost} from '../../redux/acti
```

然后，把 `clearPost` 注入到 EditPost 组件中：

```
export default connect(..., {... clearPost})(
```

当 EidtPost 组件卸载的时候，调用 `clearPost` 清空 store 中的 post 状态：

```
componentWillUnmount() {
  this.props.clearPost();
}
```

ShowPost 组件中调用 clearPost 创建函数

修改 ShowPost 组件，导入 `clearPost` 创建函数：

```
import {getPost, clearPost} from '../../redu
```

然后，把 `clearPost` 注入到 ShowPost 组件中：

```
export default connect(mapStateToProps, {getP
```

当 ShowPost 组件卸载的时候，调用 `clearPost` 清空 store 中的 post 状态：

```
componentWillUnmount() {
  this.props.clearPost();
}
```

服务器端编写更新文章接口

修改 `server/routes.js` 文件，编写 PUT `/posts/:post_id` 接口，如下：

```
app.put('/posts/:post_id', requireAuth, uploa
  Post.findById({_id: req.params.post_id}, fu
    if (err) return res.status(422).json({err
    post.name = req.body.name;
    post.content = req.body.content;
    if(req.file && req.file.filename) {
      post.cover = req.file.filename;
    }
    post.save(function(err) {
      if (err) return res.status(422).json({e
      res.json({
        post: post,
        message: '文章更新成功了！'
      });
    });
  });
})
```

提交更新文章表单

修改 EditPost 组件，首先导入异步 Action 创建函数 `editPost`，向服务器端请求更新文章接口：

```
import {getPost, clearPost, editPost} from '.
```

然后，通过 `connect` 方法，把 `editPost` 方法注入到 EditPost 组件：

```
export default connect(mapStateToProps, {getP
```

然后，提交表单，获取表单中的数据，通过

editPost 函数传送给服务器，更新文章：

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

定义 editPost 创建函数

修改 redux/actions/postActions.js 文件，添加代码：

```
export function editPost(data, id) {
  let formData = new FormData();
  formData.append('name', data.name);
  formData.append('content', data.content);
  formData.append('post', data.file);
  return function(dispatch) {
    axios.put(`${Settings.host}/posts/${id}`,
      headers: {'Authorization': sessionStora
    }).then(response => {
      dispatch({ type: 'EDIT_POST', post: res
      browserHistory.push('/dashboard');
      console.log(response.data.message)
    }).catch(error => {
      handleError(error);
    });
  }
}
```

修改 posts reducer

修改 `redux/reducers/posts.js` 文件，首先导入 `map` 方法：

```
import map from 'lodash/fp/map';
```

然后，添加响应 `EDIT_POST` action 的代码，编辑文章成功之后，同时也更新 Redux store 中的 posts 状态：

```
switch(action.type) {
  ...
  case 'EDIT_POST':
    return map((post, index) => {
      if(post._id === action.post._id) {
        return action.post
      } else {
        return post
      }
    }, state)
  default:
    return state
}
```

这样，就可以更改文章了，不过必须更改图片，才能更新文章。若只修改文字，浏览器中就报告错误，原因是 BasicForm 组件中的 state 变量 file 没有初始化。

修复错误

修改 BasicForm 组件，初始化 `file` state：

```
constructor(props) {
  super(props);
  this.state = {
    file: ''
  };
}
```