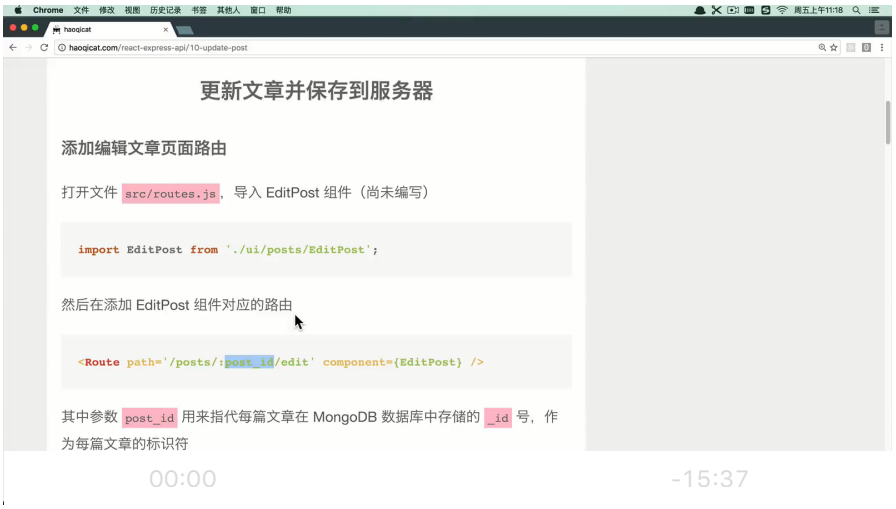

更新文章并保存到服务器

添加编辑文章页面路由

打开文件 `src/routes.js`，导入 EditPost 组件（尚未编写）

```
import EditPost from './ui/posts/EditPost';
```

然后添加 EditPost 组件对应的路由

```
<Route path='/posts/:post_id/edit' component=
```

其中参数 `post_id` 用来指代每篇文章在 MongoDB 数据库中存储的 `_id` 号，作为每篇文章的标识符

添加文章编辑入口

打开 `ui/posts/PostList.js` 文件，然后给首页文章列表中的每篇文章添加一个 编辑 链接，位于 查看 链接的下方

```
<Link to={`/posts/${post._id}/edit`} style={s
```

创建 EditPost 组件

新建文件 `src/ui/posts/EditPost.js`，添加代码：

```
import React, { Component } from 'react';
import axios from 'axios';
import Form from './Form';
import Settings from '../../settings';

class EditPost extends Component {
  publishPost(data) {
    axios.put(`${Settings.host}/posts/${this.
    .then(res => {
      console.log(res.data.message);
      this.context.router.push('/');
    })
    .catch(error => {
      if (error.response) {
        console.log(error.response.data.erro
      } else {
        console.log(error.message);
      }
    })
  }
```

```
  getStyles() {
    return {
      content: {
        width: '100%',
        maxWidth: '600px',
        margin: '30px auto',
        backgroundColor: '#fff',
        borderRadius: '10px',
        boxShadow: 'rgba(0, 0, 0, 0.12) 0px 1
      },
      title: {
        fontSize: '1.2em',
        textAlign: 'center',
        paddingTop: '20px'
      }
    };
  }

  render() {
    const styles = this.getStyles();
    return (
      <div style={styles.content}>
        <div style={styles.title}>修改文章</di
        <Form label='更新文章' publishPost={th
      </div>
    );
  }
}

EditPost.contextTypes = {
  router: React.PropTypes.object.isRequired
};

export default EditPost;
```

EditPost 组件和 NewPost 组件的模样几乎一样，使用了相同的 Form 组件，只是传递的属性值不同，两个组件的功能也就区别开来了。EditPost 组件的 `publishPost` 方法向 API 地址 `${Settings.host}/posts/:post_id` 发送 PUT 请求，更新服务器上存储的 `_id` 号为 `this.props.params.post_id` 的文章

获取所编辑文章的数据

虽然已经能更新文章了，但是用户体验很糟糕，应该把所编辑文章的当前信息在表单中显示出来，然后再对文章进行修改。接下来我们要从服务器获取单个文章的信息。

继续对 EditPost 进行修改，初始化一个 state 变量 post 存储一篇文章的数据，其类型是一个 JS 对象。

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

然后在 EditPost 组件渲染之前，调用 API `${Settings.host}/posts/:post_id` ，若请求成功，则把返回的数据赋值给 state 变量 post

```
componentWillMount() {
```

```
    axios.get(`${Settings.host}/posts/${this.pa
      this.setState({
        post: res.data.post
      })
    })
    .catch(error => {
      if (error.response) {
        console.log(error.response.data.error);
      } else {
        console.log(error.message);
      }
    });
  }
```

接下来，导入 `isEmpty` 方法:

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

最后，判断 state 变量 post 的值是否为空，若为空，则不加载 Form 组件，若不为空，则把 state 变量 post 的值作为属性值传递给 Form 组件。

```
{
  !isEmpty(this.state.post) ? <Form label='更
}
```

下面就看一下如何在 Form 表单中使用 `post` 属性值

修改表单

给每个输入框组件都添加一个 `defaultValue` 属性，

其属性值为输入框默认显示的内容。首先修改 分类 输入框组件：

```
<input style={styles.input} ref='category' de
```

若 post 属性值不为空，则输入框中显示文章的分类，若为空，输入框中什么也不显示。然后再修改 标题 输入框组件：

```
<input style={styles.input} ref='title' defau
```

然后再修改 内容 文本框组件：

```
<textarea style={[styles.input, {height: '10(
```

Form 表单修改完毕之后，就可以让 NewPost 组件和 EditPost 组件共用了。