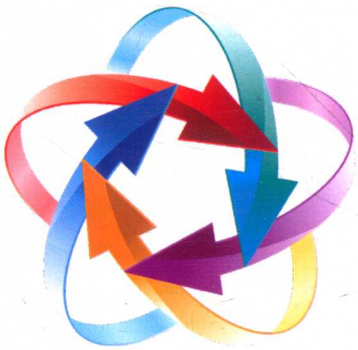
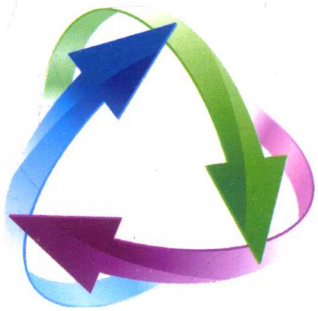
HZ BOOKS
华章IT

资深架构师十余年工程经验结晶

由浅入深介绍如何用React和Redux构建前端项目，产出高质量易维护代码

實戰





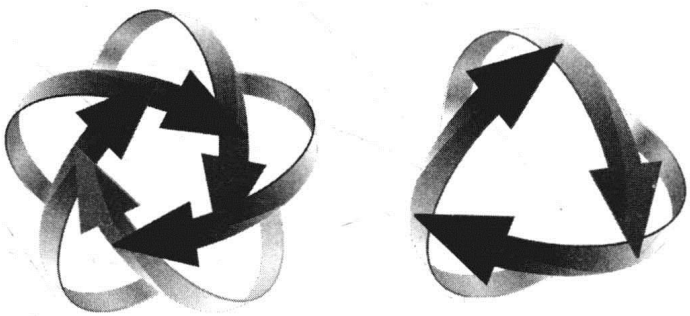
程墨 编著

Dissecting React & Redux

深入浅出 React和Redux

机械工业出版社
China Machine Press

實戰



Dissecting React & Redux

深入浅出 React和Redux

程墨 编著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

深入浅出 React 和 Redux / 程墨编著 . —北京：机械工业出版社，2017.4
（实战）

ISBN 978-7-111-56563-5

I. 深… II. 程… III. JAVA 语言 – 程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2017）第 062853 号

深入浅出 React 和 Redux

出版发行：机械工业出版社（北京市西城区百万庄大街 22 号 邮政编码：100037）
责任编辑：吴 怡 责任校对：殷 虹
印 刷：三河市宏图印务有限公司 版 次：2017 年 4 月第 1 版第 1 次印刷
开 本：186mm × 240mm 1/16 印 张：17
书 号：ISBN 978-7-111-56563-5 定 价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88379426 88361066 投稿热线：（010）88379604
购书热线：（010）68326294 88379649 68995259 读者信箱：hzit@hzbook.com

Preface 前　言

互联网技术发展一日千里，网页应用开发技术也不例外，这本书介绍的是在这一领域备受瞩目的两个工具 React 和 Redux。

自从 jQuery 问世以来，它就在网页开发领域占据统治地位，同时，还有许多 MVC 框架如雨后春笋般出现。但是业界也普遍发现，jQuery 和各种 MVC 框架在开发大型复杂应用时，依然面临很多难以克服的困难。

当 2014 年 Facebook 推出 React 时，给整个业界带来全新的看待网页应用开发的方式，和 React 一同问世的 Flux，也克服传统 MVC 框架的很多弊病。技术在不断发展，在 2015 年，Flux 的一个变体 Redux 出现，进一步优化了 Flux 的功能。

React 和 Redux 的结合，让网页开发的方式耳目一新，写这本书的初衷，是为了让国内读者能够一睹 React 和 Redux 的内在原理并深入实践。

在这里深入介绍 React 和 Redux，绝不是贬抑其他前端框架，事实上，开发者应该接触不同的开发模式，才能融会贯通，对技术有一个全面的认识，若要掌握某种技术，就要深入学习，这就是本书的目的。对 React 和 Redux 的了解不要只是停留在能用的表面功夫，重要的是理解内在的原理。

本书的内容

希望读者把阅读这本书的过程当做一个旅程，由浅入深地了解 React 和 Redux，如果你对 React 和 Redux 技术已经有一些了解，可以直接跳到感兴趣的章节。本书包括 12 章，如下所示。

第 1 章，React 新的前端思维方式。实际操作快速创建一个 React 应用，介绍和传统网页开发相比 React 应用开发的独特方式。

第 2 章，设计高质量的 React 组件。React 提倡基于组件的设计，这一章通过开发一个

ControlPanel 组件的实践，介绍了开发高质量 React 组件的原则，详细介绍 React 组件的生命周期和数据管理方式。

第 3 章，从 Flux 到 Redux。通过 Flux 介绍了单向数据流的框架模式，由此引出比 Flux 更优秀的 Redux 框架，通过用不同框架实现 ControlPanel 应用可以比较框架的优劣。

第 4 章，模块化 React 和 Redux 应用。这一章通过开发一个 Todo 应用介绍将 React 和 Redux 结合的方法。

第 5 章，React 组件的性能优化。通过对 Todo 应用的性能优化，介绍提高 React 组件渲染性的方法，以及提高从 Store 获取数据性能的方法。

第 6 章，React 高级组件。介绍高阶组件和“以函数为子组件”的模式。

第 7 章，Redux 和服务器通信。通过开发一个天气信息应用的实践，介绍应如何在 React 和 Redux 的环境中实现与服务器的通信。

第 8 章，单元测试。介绍针对 React 和 Redux 的单元测试技巧。

第 9 章，扩展 Redux。介绍创建中间件和 Store Enhancer 的技巧。

第 10 章，动画。介绍在 React 中通过 ReactTransitionGroup 和 React-Motion 库实现动画的技巧。

第 11 章，多页面应用。介绍如何创建多页面路由，以及为了提高网页装载性能的代码分片技巧。

第 12 章，同构。创建让 React 组件能够在服务器端和浏览器端渲染的技术。

本书的目标读者

阅读这本书只需要一些基本的 JavaScript、HTML 和 CSS 知识，了解网页应用的工作原理，就足够具备体验 React 和 Redux 这种全新的开发方式。

如果你熟悉传统的 jQuery 应用开发，那么通过阅读本书会让你发现不一样的应用构建模式；如果你之前学习过 Angular.js 或者 Vue.js，那么对理解 React 和 Redux 的工作机理很有帮助，同时有机会体验同样一种思想的不同实现之道。

即使你对 React 和 Redux 已经有了一定认识，相信阅读此书也不会让你觉得是浪费时间，因为书中不只是介绍“如何去做”，更多地还解释了“为什么这么做”，相信阅读此书会让你对 React 和 Redux 会有更多更深的认识。

源代码

本书每章都附带大量的实际代码例子，因为篇幅所限，在书中不可能包含所有代码，

读者可以在 Github（网址 https://github.com/mocheng/react-and-redux）上找到所有代码，代码按照所属章节内容组织。

如果读者发现代码或者书中的错误，可以直接在上面网址对应的代码库中提交问题，请不吝斧正。

致谢

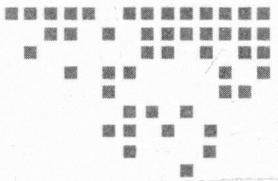
首先要感谢我的家人，没有他们的帮助和理解，这本书不可能完成。

感谢 Hulu 公司，本书中的很多内容都是和 Hulu 的研发团队协同合作中得到的体会。

感谢机械工业出版社的吴怡编辑，因为她的鼓励和帮助，这本书才得以问世。

最后要感谢 React 和 Redux 社区，因为千千万万开发者以开放的心态贡献代码和积极讨论，前端开发技术才获得巨大的飞跃，这个世界才变得更加美好。

目　录 *Contents*



第 1 章 Chapter 1

React 新的前端思维方式

我们先来直观认识 React，对任何一种工具，只有使用才能够熟练掌握，React 也不例外。通过对 React 快速入手，我们会解析 React 的工作原理，并通过与功能相同的 jQuery 程序对比，从而看出 React 的特点。

在这一章中，我们会介绍：

- 如何初始化一个 React 项目；
- 如何创建一个 React 组件；
- React 的工作方式。

让我们开始旅程吧！

1.1 初始化一个 React 项目

为了开发 React 应用，你的电脑是运行微软 Windows 操作系统，还是苹果 Mac，或者是 Linux，并不重要，只需要保证具备以下条件：

- 安装了浏览器，如果是 Windows 操作系统，请保证微软 IE 浏览器版本不低于 8.0 版，因为 React 不支持比 IE 8 更低版本的浏览器；
- 有一个命令行环境，在 Windows 操作系统中有命令行界面，在苹果 Mac 电脑中可以使用 Terminal 应用，对于 Linux 用户，命令行环境我想不用过多解释；
- 一个你最喜欢的代码编辑器，用于编辑 React 应用的代码，本书内容注重实践，只有实际编码才能有深入体会。

作为开发者，推荐使用谷歌 Chrome 浏览器，因为 Chrome 浏览器自带的开发辅助工具非常友好，而且还可以安装辅助 React 和 Redux 的扩展工具，具体的开发工具在第 4 章 4.2 节中有详细介绍。

React 是一个 JavaScript 语言的工具库，在这个 JavaScript 工具铺天盖地的时代，没有意外，你需要安装 Node.js，React 本身并不依赖于 Node.js，但是我们开发中用到的诸多工具需要 Node.js 的支持。

在 Node.js 的官网 (https://nodejs.org/) 可以找到合适的安装方式，安装 Node.js 的同时也就安装了 npm，npm 是 Node.js 的安装包管理工具，因为我们不可能自己开发所有功能，会大量使用现有的安装包，就需要 npm 的帮助。

1.1.1 create-react-app 工具

React 技术依赖于一个很庞大的技术栈，比如，转译 JavaScript 代码需要使用 Babel，模块打包工具又要使用 Webpack，定制 build 过程需要 grunt 或者 gulp……这些技术栈都需要各自的配置文件，还没有开始写一行 React 相关代码，开发人员就已经被各种技术名词淹没。

针对这种情况，React 的创建者 Facebook 提供了一个快速开发 React 应用的工具，名叫 create-react-app，这个工具的目的是将开发人员从配置工作中解脱出来，无需过早关注这些技术栈细节，通过创建一个已经完成基本配置的应用，让开发者快速开始 React 应用的开发。

本书中所有应用实例都由 create-react-app 创建，我们用这种最简单的方式创建可运行的应用，必要的时候才会介绍底层技术栈的细节，毕竟，没有什么比一个能运行的应用更加增强开发者的信心。

create-react-app 是一个通过 npm 发布的安装包，在确认 Node.js 和 npm 安装好之后，命令行中执行下面的命令安装 create-react-app：

```
npm install --global create-react-app
```

安装过程结束之后，你的电脑中就会有 create-react-app 这样一个可以执行的命令，这个命令会在当前目录下创建指定参数名的应用目录。

我们在命令行中执行下面的命令：

```
create-react-app first_react_app
```

这个命令会在当前目录下创建一个名为 first_react_app 的目录，在这个目录中会自动添加一个应用的框架，随后我们只需要在这个框架的基础上修改文件就可以开发 React

应用，避免了大量的手工配置工作：

在 create-react-app 命令一大段文字输出之后，根据提示，输入下面的命令：

```
cd first_react_app
npm start
```

这个命令会启动一个开发模式的服务器，同时也会让你的浏览器自动打开了一个网页，指向本机地址 http://localhost:3000/，显示界面如图 1-1 所示。

注意　本书中的截图是根据 create-react-app 1.0.0 版本所得，其他版本产生的页面可能略有不同。

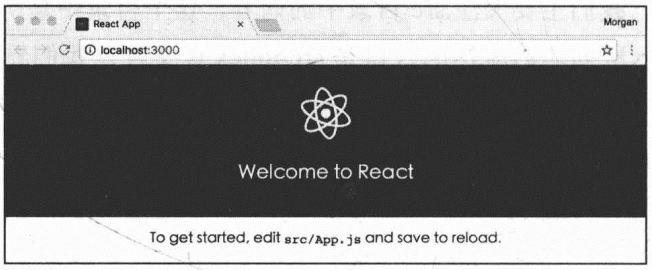


图 1-1　由 create-react-app 创造的 React 应用界面

恭喜你，你的第一个 React 应用诞生了！

接下来，我们会用 React 开发一个简单的功能，让我们继续吧。

1.2　增加一个新的 React 组件

React 的首要思想是通过组件（Component）来开发应用。所谓组件，简单说，指的是能完成某个特定功能的独立的、可重用的代码。

基于组件的应用开发是广泛使用的软件开发模式，用分而治之的方法，把一个大的应用分解成若干小的组件，每个组件只关注于某个小范围的特定功能，但是把组件组合起来，就能够构成一个功能庞大的应用。如果分解功能的过程足够巧妙，那么每个组件可以在不同场景下重用，那样不光可以构建庞大的应用，还可以构建出灵活的应用。打个比方，每个组件是一块砖，而一个应用是一座楼，想要一次锻造就创建一座楼是不现实的。实际上，总是先锻造很多砖，通过排列组合这些砖，才能构建伟大的建筑。

React 非常适合构建用户交互组件，让我们从创建一个 React 组件开始。

学习任何一门语言或者任何一门框架，往往是从写 Hello World 程序开始，不过只是

展示一句 Hello World 并不足以体现 React 的神奇能力，所以，我们要做一个不那么简单的组件，为了体现 React 对交互功能的支持，我们做一个显示点击次数的组件。

我们先看一看 create-react-app 给我们自动产生的代码，在 first-react-app 目录下包含如下文件和目录：

```
src/
public/
README.md

package.json

node_modules/
```

在开发过程中，我们主要关注 src 目录中的内容，这个目录中是所有的源代码。

create-react-app 所创建的应用的入口是 src/index.js 文件，我们看看中间的内容，代码如下：

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

这个应用所做的事情，只是渲染一个名叫 App 的组件，App 组件在同目录下的 App.js 文件中定义，渲染出来的效果就是在图 1-1 中看到的界面。

我们要定义一个新的能够计算点击数组件，名叫 ClickCounter，所以我们修改 index.js 文件如下：

```
import React from 'react';
import ReactDOM from 'react-dom';
import ClickCounter from './ClickCounter';
import './index.css';

ReactDOM.render(
  <ClickCounter />,
  document.getElementById('root')
);
```

我们接下来会介绍代码的含义。现在我们先来看看如何添加一个新组件，在 src 目录下增加一个新的代码文件 ClickCounter.js，代码如下：

```
import React, { Component } from 'react';
class ClickCounter extends Component {
```

```
  constructor(props) {
    super(props);
    this.onClickButton = this.onClickButton.bind(this);
    this.state = {count: 0};
  }

  onClickButton() {
    this.setState({count: this.state.count + 1});
  }

  render() {
    return (
      <div>
        <button onClick={this.onClickButton}>Click Me</button>
        <div>
        Click Count: {this.state.count}
        </div>
      </div>
    );
  }
}
export default ClickCounter;
```

如果你是从上一节不停顿直接读到这一节，而且没有关闭命令行中的 npm start 命令，当你保存完这个文件之后，不需要主动做刷新网页的动作，就会发现网页中的内容已经发生改变，如图 1-2 所示。

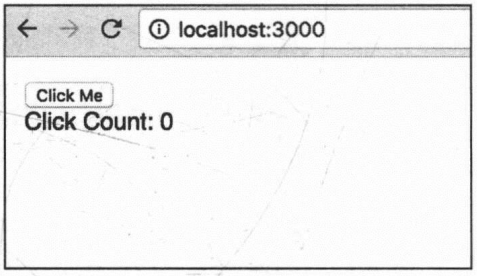


图 1-2　ClickCounter 组件界面效果

去点击那个“Click Me”按钮，可以看到“Click Count”后面的数字会随之增加，每点击一次加一。

恭喜你，现在你已经构建了一个有交互性的组件！

现在让我们来逐步详细解释代码中各部分的要义。

在 index.js 文件中，使用 import 导入了 ClickCounter 组件，代替了之前的 App 组件。

```
import ClickCounter from './ClickCounter';
```

import 是 ES6（EcmaScript 6）语法中导入文件模块的方式，ES6 语法是一个大集合，大部分功能都被最新浏览器支持。不过这个 import 方法却不在广泛支持之列，这没有关系，ES6 语法的 JavaScript 代码会被 webpack 和 babel 转译成所有浏览器都支持的 ES5 语法，而这一切都无需开发人员做配置，create-react-app 已经替我们完成了这些工作。

在 ClickCounter.js 文件的第一行，我们从 react 库中引入了 React 和 Component，如下所示：

```
import React, { Component } from 'react';
```

Component 作为所有组件的基类，提供了很多组件共有的功能，下面这行代码，使用的是 ES6 语法来创建一个叫 ClickCounter 的组件类，ClickCounter 的父类就是 Component：

```
class ClickCounter extends Component {
```

在 React 出现之初，使用的是 React.createClass 方式来创造组件类，这种方法已经被废弃了，但是在互联网上依然存在大量的文章基于 React.createClass 来讲解 React，这些文章中依然有很多真知灼见的部分，但是读者要意识到，使用 React.createClass 是一种过时的方法。在本书中，我们只使用 ES6 的语法来构建组件类。

细心的读者会发现，虽然我们导入的 Component 类在 ClickCounter 组件定义中使用了，可是导入的 React 却没有被使用，难道在这里引入 React 没有必要吗？

事实上，引入 React 非常必要，你可以尝试删掉第一行中的 React，在网页中立刻会出现错误信息，如图 1-3 所示。

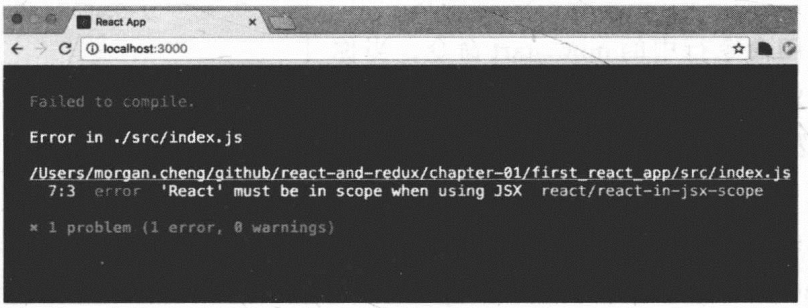


图 1-3 缺失 React 的错误

这个错误信息的含义是：“在使用 JSX 的范围内必须要有 React。”

也就是说，在使用 JSX 的代码文件中，即使代码中并没有直接使用 React，也一定要导入这个 React，这是因为 JSX 最终会被转译成依赖于 React 的表达式。

接下来，我们就要认识什么是 JSX。

1.2.1 JSX

所谓 JSX，是 JavaScript 的语法扩展（eXtension），让我们在 JavaScript 中可以编写像 HTML 一样的代码。在 ClickCounter.js 的 render 函数中，就出现了类似这样的 HTML 代码，在 index.js 中，ReactDOM.render 的第一个参数 <App /> 也是一段 JSX 代码。

JSX 中的这几段代码看起来和 HTML 几乎一模一样，都可以使用 <button> 之类的元素，所以只要熟悉 HTML，学习 JSX 完全不成问题，但是，我们一定要明白两者的不同之处。

首先，在 JSX 中使用的“元素”不局限于 HTML 中的元素，可以是任何一个 React 组件，在 App.js 中可以看到，我们创建的 ClickCounter 组件被直接应用在 JSX 中，使用方法和其他元素一样，这一点是传统的 HTML 做不到的。

React 判断一个元素是 HTML 元素还是 React 组件的原则就是看第一个字母是否大写，如果在 JSX 中我们不用 ClickCounter 而是用 clickCounter，那就得不到我们想要的结果。

其次，在 JSX 中可以通过 onClick 这样的方式给一个元素添加一个事件处理函数，当然，在 HTML 中也可以用 onclick（注意和 onClick 拼写有区别），但在 HTML 中直接书写 onclick 一直就是为人诟病的写法，网页应用开发界一直倡导的是用 jQuery 的方法添加事件处理函数，直接写 onclick 会带来代码混乱的问题。

这就带来一个问题，既然长期以来一直不倡导在 HTML 中使用 onclick，为什么在 React 的 JSX 中我们却要使用 onClick 这样的方式来添加事件处理函数呢？

1.2.2　JSX 是进步还是倒退

在 React 出现之初，很多人对 React 这样的设计非常反感，因为 React 把类似 HTML 的标记语言和 JavaScript 混在一起了，但是，随着时间的推移，业界逐渐认可了这种方式，因为大家都发现，以前用 HTML 来代表内容，用 CSS 代表样式，用 JavaScript 来定义交互行为，这三种语言分在三种不同的文件里面，实际上是把不同技术分开管理了，而不是逻辑上的“分而治之”。

根据做同一件事的代码应该有高耦合性的设计原则，既然我们要实现一个 ClickCounter，那为什么不把实现这个功能的所有代码集中在一个文件里呢？

这点对于初学者可能有点难以接受，但是相信当你读完此书后，观点会随之改变。

那么，在 JSX 中使用 onClick 添加事件处理函数，是否代表网页应用开发兜了一个大圈，最终回到了起点了呢？

不是这样，JSX 的 onClick 事件处理方式和 HTML 的 onclick 有很大不同。

即使现在，我们还是要说在 HTML 中直接使用 onclick 很不专业，原因如下：

- onclick 添加的事件处理函数是在全局环境下执行的，这污染了全局环境，很容易产生意料不到的后果；
- 给很多 DOM 元素添加 onclick 事件，可能会影响网页的性能，毕竟，网页需要