概述

前面一章就是全书的最后一章节。我希望你享受阅读本书的过程，也希望这本书对你学习 React 有所帮助。如果你喜欢这本书，请将其作为学习 React 的一种方法推荐给你的朋友们。授人玫瑰，手有余香。此外，若不介意的话请花上五分钟在亚马逊上写个短评。

但在阅读本书之后，你又将去向何处呢？你可以自行扩展这个应用，也可以尝试构建属于自己的 React 项目。在你深入另一本书、课程或教程之前，你应该动手创建一个属于自己的 React 项目。持续做上一个星期，把它上线部署到某个地方，然后可以通过 Twitter 联系我把它展示出来。我很好奇你在看到本书之后会创造出点什么，我也很乐于跟我的粉丝们一起分享你的作品。你也可以在 GitHub 找到我并分享你的代码库。

本书中我们只是操练了纯粹的 React，如果你需要进一步扩展你的应用程序，我可以再推荐几个学习途径：

- **状态管理：**相信你已经使用过 React 中的 `this.setState()` 和 `this.state` ，用于管理和存取组件内部状态。千里之行，始于足下。然而在更大型的项目中，你就会切身体会到 React 组件内部 state 的局限性。因此你可以使用一个第三方的状态管理库，比如说 Redux 或 MobX。你会发现在 Road to React 授课平台上，《驾驭 React 中的状态》（"Taming the State in React"）这门课程将会传授 Redux、MobX 与 React 本地 state 的进阶内容。这门课程也会包含一本电子书，但我推荐大家可以深入到源代码和录屏教学中去。如果你喜欢这本书，那么毫无疑问你也应该看一看《驾驭 React 中的状态》。
- **项目实例：**在学习了纯 React 内容过后，要将所学内容运用到自己的项目上，而不是急于学习其他新东西，这对你总是有好处的。你可以使用 React 编写自己的井字游戏（tic-tac-toe）或是一个简单的计算器。有很多不错的教程会教你，仅仅使用 React 就能打造出一些有意思的玩意儿。看看我所做的诸如 分页和无限滚动列表，在 Twitter 墙上展示推文或是 为 React 应用集成 Stripe 支付功能（译者注：Stripe 类似于支付宝）。
- **代码结构：**在阅读本书的过程中，你应该对提到组织代码结构那一章有印象。如果还没来得及实践的话，你现在就可以实际运用起来，可以把你的组件们放到结构化的文件和目录（模块）当中。此外，这也能帮助

你理解和学习关于代码拆分、可复用性、可维护性和模块 API 设计的原则。

- **连接到数据库和/或认证：**在一个不断增长的 React 应用程序中，你可能希望最终能够持久化存储数据。数据应该存储在数据库中，以便于浏览器会话结束后仍可继续使用，并能在应用程序的不同用户之间进行共享。引入数据库最简单的方法就是使用 Firebase。 在这个全面的教程当中，你会找到一份如何在 React 中使用 Firebase 进行身份验证（注册、登录、注销、…）且循序渐进的指南。除此之外，你还将使用 Firebase 的实时数据库来存储用户实体。在此之后，你可以为你的应用程序存储更多的数据。
- **测试：**这本书对测试的涉及尚浅。如果你对测试这个大话题还不太熟悉的话，你应该尝试进一步了解单元测和集成测试的相关概念，特别是在 React 应用的上下文里面。从实现层面上来说，我会强烈推荐 Enzyme 和 Jest，通过单元测试和快照测试来改善你的 React 测试手法。
- **异步请求：**你可以使用执行异步请求的第三方库来替代原生的 fetch API：superagent 或 axios。发送异步请求并没有完美的解决方案。但是通过更换 React 外围的组成部分，你可以切身体会到在 React 当中拥有这种灵活性是多么的强大。在某些框架中通常你只能使用某一种方案，但是在诸如 React 这样的灵活的生态系统当中，你可以任意更换解决方案。
- **路由：**你可以使用 react-router 为你的应用程序实现路由功能。到目前为止，你的应用程序还只有一个页面。React Router 则能够让你跨多个 URL 创建多个页面。在将路由引入你的应用之后，你不需要发送任何请求到服务端去获取下个页面。路由器（Router）将会帮你在客户端搞定一切。
- **类型检查：**在某个章节，你已经使用过 React PropTypes 来定义组件接口。这是预防 bugs 的一种良好实践，但是 PropTypes 只能在运行时执行检查。你可以更进一步地，在编译时就引入静态类型检查。TypeScript 就是备受欢迎的手段之一。但在 React 生态当中，通常情况下大家会使用 Flow。如果你想要让你的应用程序更加健壮的话，我会推荐你去尝试一下 Flow。
- **Webpack 和 Babel 相关工具：**在本书中你已经使用过 *create-react-app* 来创建应用程序。到了某个节点，当你已经对 React 足够了解的时候，你可能就想要学习跟 React 相关的一些工具。这可以让你不用 *create-react-app* 也能初始化自己的项目。我会推荐你先了解如何使用 Webpack 和 Babel 完成最少量的配置，然后你可以根据自己的情况去实践更多的工具。例如，你可以在应用程序中使用 ESLint 来统一代码风格。

- **React Native：**React Native 可以将你的应用程序带到移动设备上。React Native 使你能够把在 React 中所学到的知识应用到 iOS 和 Android 应用当中去。一旦你学会了 React，React Native 的学习曲线就应该不会那么陡峭，两者都是相同的原则和理念。你只是会在移动设备上碰到一些跟 Web 应用有所不同的布局组件。

总之，我希望你能来我的网站看一看，你会发现更多关于 Web 开发和软件工程的有趣内容。你也可以订阅我的更新，大概每月一次就会送达你的收件箱，同时你可以通过 Patron 众筹来支持我的这些内容。此外，在通向 React 之路（Road to React）授课平台上会提供更多学习 React 生态圈的进阶课程。你不应该错过！

请允许我再啰嗦一次，如果你喜欢这本书，我希望你能花点时间想一想谁比较适合学习 React。找到他们然后把这本书分享给他。这对我真的非常重要，这本书就旨在与他人分享。随着时间的推移，越来越多阅读本书的人会与我分享他们的看法和意见，这也会帮助我改进这本书。

由衷地感谢你阅读这本书，《React 学习之道》。

Robin