React Design Patterns and Best Practices

React设计模式与最佳实践

[意] 米凯莱·贝尔托利 著
林昊 译

- Facebook前端工程师15年一线开发经验凝结
- 深入探讨React核心模式与组件，创建可复用的代码和可扩展的设计

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

React设计模式与最佳实践

TURING 图灵程序设计丛书

React Design Patterns and Best Practices

React设计模式与最佳实践

[意] 米凯莱·贝尔托利 著
林昊 译

人民邮电出版社
北京

图书在版编目（CIP）数据

React设计模式与最佳实践 / （意）米凯莱·贝尔托利（Michele Bertoli）著 ; 林昊 译. -- 北京 : 人民邮电出版社, 2018.8
（图灵程序设计丛书）
ISBN 978-7-115-48875-6

Ⅰ. ①R… Ⅱ. ①米… ②林… Ⅲ. ①移动终端－应用程序－程序设计 Ⅳ. ①TN929.53

中国版本图书馆CIP数据核字(2018)第157759号

内 容 提 要

本书共分为12章，通过介绍React中最有价值的设计模式，展示如何将设计模式和最佳实践应用于现实的新项目和已有项目中。主要内容包括帮助读者理解React的基本概念，学习编写整洁、可维护的代码；优化React组件，使应用拥有更快的速度和响应性；介绍如何有效地编写测试，避免反模式，开源组件并对React生态系统做贡献。

本书适合想要深入理解React，希望提高相关编程技能的前端开发人员阅读。

◆ 著 [意] 米凯莱·贝尔托利
译 林 昊
责任编辑 杨 琳
责任印制 周昇亮
◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
三河市君旺印务有限公司印刷
◆ 开本：800×1000 1/16
印张：14.75
字数：357千字 2018年8月第1版
印数：1－3 000册 2018年8月河北第1次印刷
著作权合同登记号 图字：01-2017-5047号

定价：59.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字20170147号

站在巨人的肩上

Standing on Shoulders of Giants

iTuring.cn

版权声明

致　谢

我要感谢我的妻子和儿子，他们用微笑让我的生活变得更美好。Dante，我将一些时间用来撰写这本书而没有陪你玩耍，希望等你长大后能理解并因为我而感到骄傲。另外，还要感谢Packt出版社给予我这次机会。

前　言

本书将带你全面了解 React 中最有价值的设计模式，并展示如何在全新或已有的真实项目中应用设计模式与最佳实践。本书将帮助你让应用变得更加灵活、运行更流畅并且更容易维护——在不降低质量的情况下极大地提升工作流的速度。

我们将首先理解 React 的内部原理，接着逐步编写整洁且可维护的代码。我们将开发能够在整个应用中复用的组件，搭建应用架构，并创建真正可用的表单。

接下来，我们会为 React 组件编写样式并优化组件，从而使应用运行得更快且更具响应性。最后，我们将有效地编写测试代码，还会学到如何为 React 及其生态系统做贡献。

学完本书后，你会从大量的试错以及开发难题中解脱出来，也将踏上成为 React 专家的道路。

本书内容

第 1 章，React 基础。这一章从高级角度介绍了 React 的基本概念。

第 2 章，整理代码。这一章讲解了编写可维护代码中最重要的一个方面，即保持代码整洁并遵循编程风格指南。了解函数式编程的基础知识对于使用 React 也很重要。

第 3 章，开发真正可复用的组件。这一章阐述了构建应用的一个关键因素在于使用组件，而要想保持代码库整洁且可维护，最重要的是开发真正可复用的组件。

第 4 章，组合一切。这一章阐述了真实应用由不同的组件构成，重要的是让组件之间可以高效地通信，并按照正确的形式组织和搭建层次结构。

第 5 章，恰当地获取数据。这一章指明了任何客户端应用在某些时刻都必须处理数据，并且介绍了不同的技巧和方法，让你能够以 React 的方式获取数据。

第 6 章，为浏览器编写代码。这一章阐述了如何正确使用在浏览器中运行的应用，还讲解了一些高级概念，如事件、动画以及如何与 DOM 交互。

第 7 章，美化组件。这一章说明了开发美观的 UI 组件是前端工程中很重要的一部分内容。

React 可以通过多种方式实现这个目的，每种方式从不同角度解决该问题。了解可用的库及其工作原理，对于做出正确的选择至关重要。

第 8 章，服务端渲染的乐趣与益处。这一章指明了服务端渲染是 React 众多优秀特性之一。虽然该特性开箱即用，但学习其正确用法很重要，因为这样才能充分加以利用。

第 9 章，提升应用性能。这一章阐述了性能是 Web 平台吸引用户的重要因素之一。React 提供了一系列工具和技术来创建快如闪电的应用，这一章将全面介绍这些内容。

第 10 章，测试与调试。这一章会让你意识到，我们都希望自己的应用保持稳定，并且能够应对一切极端情况，而测试有助于实现这个目的。编写全面的测试集对于创建稳定且可维护的代码至关重要。从另一方面来看，bug 总会出现，而知道如何调试并尽早发现问题很关键。

第 11 章，需要避免的反模式。这一章阐明了开发人员经常尝试采取捷径和创意方案这一事实，但在某些情况下这种做法对应用来说是很危险的，尤其是团队以及代码库规模很大时。这一章将带你了解使用 React 时，应该避免的常见反模式。

第 12 章，未来的行动。这是本书的最后一章，至此我们已经介绍完所有主题。我认为探讨如何开源组件（以回馈社区）以及如何为 React 及其生态系统做贡献也很重要。

阅读须知

我们需要一台计算机，并配有终端程序、Node.js/npm 环境以及浏览器。

目标读者

如果想要深入理解 React 并将其应用到真实应用的开发中，那么本书很适合你。

排版约定

本书采用不同的文本样式来区分不同类别的信息。以下展示了部分样式示例及其相应的含义。

正文中的代码、数据库表名、用户输入等采用以下样式：“循环内部包含一些条件逻辑，用于检查`#first` 和`#link` 属性是否存在，并根据它们的值渲染不同的 HTML 片段。变量位于双花括号中。”

代码块的样式如下所示：

```
const toLowerCase = input => {
  const output = []
  for (let i = 0; i < input.length; i++) {
    output.push(input[i].toLowerCase())
  }
  return output
}
```

命令行中的输入或输出内容采用以下样式：

```
npm install -g create-react-app
```

新术语和**关键词**以黑体字显示。屏幕上出现的单词（如出现在菜单或对话框中）按照如下样式显示：“我们开始更新测试代码，先从**渲染文本**的那些代码着手。”

警告或重要的注意事项。

提示或小技巧。

读者反馈

我们期待读者的反馈。告诉我们你对本书的看法，喜欢什么或者不喜欢什么。读者反馈对我们很重要，因为它有助于我们策划出令读者受益最多的图书。

要想提供反馈，只需登录“图灵社区”本书页面（http://www.ituring.com.cn/book/2007）并留言。

客户支持

为了让你购买的书物有所值，我们还为你准备了以下内容。

下载示例代码

你可以从“图灵社区”本书页面（http://www.ituring.com.cn/book/2007）下载书中示例代码。下载文件后，确保使用以下工具的最新版本来解压或提取文件夹：

- WinRAR / 7-Zip（Windows）
- Zipeg / iZip / UnRarX（Mac）
- 7-Zip / PeaZip（Linux）

勘误

虽然我们竭力确保图书内容的正确性，但错误在所难免。如果你在我们出版的任何一本图书中发现了文本或代码中的错误，希望你能告知我们，我们将非常感激。你的善举足以减少其他读者在阅读出错内容时的纠结和不快，并帮助我们在后续版本中更正错误。如果你发现任何错误，请通过“图灵社区”本书页面（http://www.ituring.com.cn/book/2007）告诉我们。一旦勘误通过确认，将显示在页面上的勘误表中。

侵权行为

所有媒体在互联网上都面临着侵权问题。我们严格保护自己的版权和许可证。如果你在互联网上发现有关我们出版物的任何形式的盗版产品，请立即告知我们地址或网站名称，以便我们进行补救。

请将盗版图书的网站地址发送到 ebook@turingbook.com。

你的反盗版行动就是在保护作者和出版社，只有这样，我们才能继续以优质内容回馈像你这样的热心读者。

问题

如果对本书存有任何方面的疑问，可以通过“图灵社区”本书页面（http://www.ituring.com.cn/book/2007）联系我们，我们将尽力为你答疑解惑。

电子书

扫描如下二维码，即可购买本书电子版。

目　录

第 1 章 React 基础

你好！

本书假设你已经知道 React 是什么并且了解它能为你解决什么问题。你可能使用 React 开发过中小型应用，但希望进一步提升自己的技能并得到所有未解决问题的答案。

你应该知道，React 由 Facebook 的开发人员及 JavaScript 社区的数百名贡献者所维护。

React 是最流行的 UI 开发库之一，因高性能而为人所熟知，这得益于它操作 DOM 的方式很巧妙。

React 包含了全新的 JSX 语法，该语法用于在 JavaScript 中编写标记，这需要你重新思考关注点分离原则[①]。React 还包含了许多很棒的特性，如服务端渲染，该特性让你可以开发通用应用。

为了学习本书，你需要了解如何使用终端程序在 Node.js 环境中安装并运行 npm 包。

本书中的所有代码示例都遵循 ES2015 标准，以方便你阅读与理解。

本章会讲解你需要掌握的一些基本概念，了解这些概念有助于你高效使用 React。对于初学者来说，理解这些概念意义重大。

- 命令式编程与声明式编程的区别。
- React 组件与组件实例，以及 React 如何使用元素来控制 UI 流程。
- React 如何改变 Web 应用的开发方式，它主张的另一种全新的关注点分离概念是什么，它选择不寻常的设计理念的原因是什么。
- 为什么人们会对 JavaScript 框架感到疲劳？在 React 生态系统中，怎样避免开发人员最常犯的错误？

① 关注点分离是软件设计原则之一，前端开发中一般指文档结构、样式表现以及脚本行为的分离。——译者注

1.1　声明式编程

只要阅读过React文档或者相关博文，那么你肯定遇到过**声明式**这一术语。

其实，React如此强大的原因之一就在于它推行声明式编程范式。

因此，要想掌握 React，就需要理解声明式编程的含义，以及其与命令式编程之间的主要区别。

理解该问题的最简方式是：命令式编程描述代码如何工作，而声明式编程则表明想要实现什么目的。

与命令式世界极其相似的一个真实示例就是去酒吧喝啤酒并对服务员做出以下指示：

- 从架子上拿一个玻璃杯；
- 将杯子放到酒桶前；
- 按下酒桶开关，将杯子倒满；
- 把杯子递给我。

但在声明式世界中，你只需要说："请给我一杯啤酒。"

按声明式方式点一杯啤酒，需要假设服务员知道如何提供啤酒，这是声明式编程工作原理的一个重要方面。

我们来看一个JavaScript代码的示例。编写一个简单函数，给定包含大写字符串的数组时，该函数返回包含相同的小写字符串的数组。

```
toLowerCase(['FOO', 'BAR']) // ['foo', 'bar']
```

解决该问题的命令式函数的实现如下所示：

```
const toLowerCase = input => {
  const output = []
  for (let i = 0; i < input.length; i++) {
    output.push(input[i].toLowerCase())
  }
  return output
}
```

首先，创建一个空数组来保存结果。接着，函数循环遍历输入数组中的所有元素，并将小写值推进空数组中。最后，返回需要输出的数组。

声明式实现如下所示：

```
const toLowerCase = input => input.map(
  value => value.toLowerCase()
)
```

输入数组中的元素会传递到 map 函数，然后 map 函数会返回包含小写值的新数组。

这里需要注意几点比较重要的差别：前一个示例不够优雅，而且需要花更多精力才能理解；后者更加简洁、易读，这对注重可维护性的大型代码库来说非常重要。

另外值得一提的是，声明式编程中无须使用变量，也不用在执行过程中持续更新变量的值。事实上，声明式编程往往避免了创建和修改状态。

我们来看最后一个示例，了解一下 React 的声明式具体指什么。

我们要解决的问题是 Web 开发中常见的一个需求：展示带有标记的地图。

JavaScript 的实现（使用 Google Maps SDK）如下所示：

```
const map = new google.maps.Map(document.getElementById('map'), {
  zoom: 4,
  center: myLatLng,
})

const marker = new google.maps.Marker({
  position: myLatLng,
  title: 'Hello World!',
})
marker.setMap(map)
```

这显然是命令式的，因为代码逐条描述了创建地图、创建标记以及在地图上添加标记的指令。

改用 React 组件在页面上显示地图的方式如下所示：

```
<Gmaps zoom={4} center={myLatLng}>
  <Marker position={myLatLng} title="Hello world!" />
</Gmaps>
```

使用声明式编程方法时，开发人员只需要描述他们想要实现什么目的，无须列出实现效果的所有步骤。

声明式编程方式使得 React 很容易使用，因此最终的代码也很简单，这样产生的 bug 也更少，可维护性也更强。

1.2 React 元素

本书假设你已经熟悉组件及其实例，但要想高效地使用 React，你还需要了解另一种对象：**元素**。

无论是调用 `createClass` 方法、继承 `Component` 类还是声明一个无状态函数，其实都是在创建组件。React 管理着运行环境中的所有组件实例，在某个特定时刻，内存中可能存在同一

个组件的多个实例。

如前文所述，React 遵循声明式范式，因此无须告诉它如何与 DOM 交互。你只要声明希望在屏幕上看到的内容，React 就会完成剩下的工作。

或许你之前体会过，大部分其他 UI 库都是按相反方式工作的：它们让开发人员负责更新界面，这就需要手动管理 DOM 元素的创建与销毁。

React 使用了**元素**这种特殊类型的对象来控制 UI 流程。元素描述了屏幕上需要显示的内容。这些不可变对象比组件和组件实例要简单得多，而且只包含了展示界面所必需的信息。

以下示例展示了一个元素：

```
{
  type: Title,
  props: {
    color: 'red',
    children: 'Hello, Title!'
  }
}
```

元素最重要的属性是 `type`，另一个比较特殊的属性是 `children`，它是可选的，用于表示元素的直接后代。当然，元素还具有其他一些属性。

`type` 属性很重要，因为它告诉 React 如何处理元素本身。实际上，如果 `type` 属性是字符串，那么元素就表示 **DOM 节点**；如果 `type` 属性是函数，那么元素就是**组件**。

DOM 元素和组件可以互相嵌套，以表示整个渲染树：

```
{
  type: Title,
  props: {
    color: 'red',
    children: {
      type: 'h1',
      props: {
        children: 'Hello, H1!'
      }
    }
  }
}
```

当元素的 `type` 属性是函数时，React 会调用它，传入 `props` 来取回底层元素。React 会一直对返回结果递归地执行相同的操作，直到取回完整的 DOM 节点树，然后就可以将它渲染到屏幕。这个过程称作**一致性比较**，React DOM 和 React Native 都利用它在各自的平台上创建 UI。