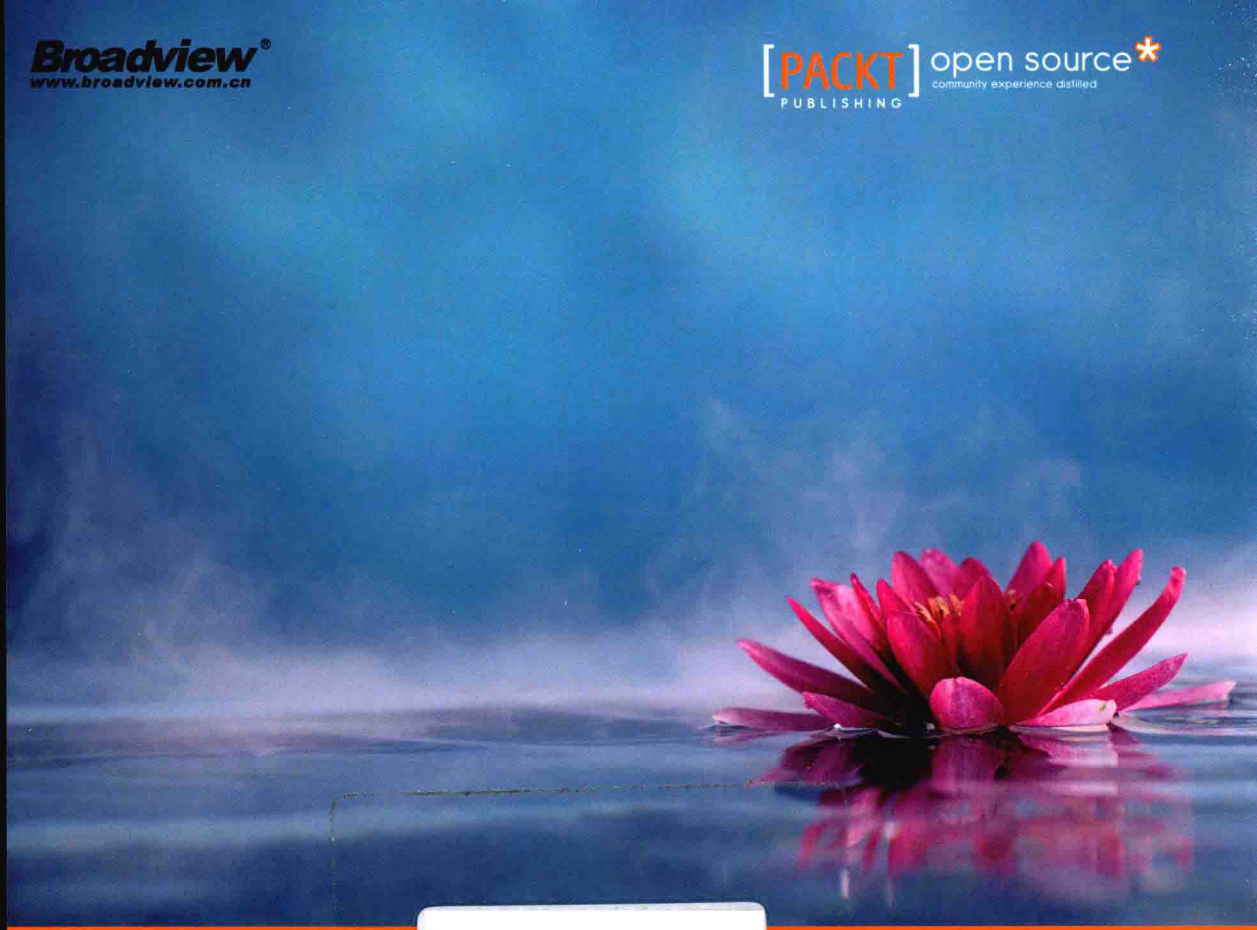
Broadview
www.broadview.com.cn

[PACKT] open source*
PUBLISHING
community experience distilled



React.js Essent

React精髓

使用React.js 构建可扩展和可维护的Web应用

[英] Artemij Fedosejev 著
奇舞团 译

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

React.js Essentials

React精髓

[英] Artemij Fedosejev 著
奇舞团 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书面向初中级前端开发者，从头到尾、由浅入深地介绍了使用 React 实现组件化 Web 应用的完整流程。作者从 React 元素、React 组件等基本的概念讲起，循序渐进地讨论了组件状态和生命周期，为开发完整的 React 应用打下了基础。与第三方 JavaScript 框架集成，以及对 React 组件进行单元测试，都是开发 React 应用的重要内容，本书也有详细讲解。最后，为进一步提升 React 应用的灵活性，作者还以实例展示了如何引入 Flux 架构，让读者的开发技能更上一层楼。

版权贸易合同登记号　图字：01-2015-8414

图书在版编目（CIP）数据

React 精髓 /（英）费多耶夫（Fedosejev,A.）著；奇舞团译. —北京：电子工业出版社，2016.5
书名原文：React.js Essentials
ISBN 978-7-121-28646-9

Ⅰ. ①R… Ⅱ. ①费… ②奇… 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(2016)第 086472 号

策划编辑：张春雨
责任编辑：徐津平
印　　刷：北京天宇星印刷厂
装　　订：北京天宇星印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编：100036
开　　本：787×980　1/16　印张：12.25　字数：244 千字
版　　次：2016 年 5 月第 1 版
印　　次：2016 年 5 月第 1 次印刷
定　　价：65.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888，88258888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式：010-51260888-819　faq@phei.com.cn。

目录

1

给项目预先安装一些有用的工具

Charles F. Kettering 曾经说过：

> 我只关心未来，因为我的余生都会在那里度过。

这位美国杰出的发明家在软件工程师这个职业诞生之前，就给我们留下了这条最重要的建议。然而，半个世纪后，我们仍在思考是什么导致了混乱的代码，或者是说混乱的思维模式。

你一定有过这种经历：你接手了之前开发者的一份代码，然而这份代码没有任何的文档，你不得不花费数周的时间来理解这份难以调试的“伪自解释代码”。更可怕的是，项目还在持续增长，代码也越来越复杂难懂。大幅修改这份代码是很危险的，并且没有人想碰这份“丑陋”的遗留代码。而重写整个项目需要很大的成本，所以目前的做法就是不停地打补丁来修复遇到的 Bug。众所周知，维护软件的成本往往比原来开发软件的成本还高。

那么今天，应该如何针对未来编写软件呢？我认为是发明一个不变的、简单的工作模式。不管你的项目规模增长到多么大，这一工作模式的复杂性总是保持不变。这个工作模式就是项目的架构，理解了这个架构，也就明白了整个软件是如何运行的。

如果你了解现代的 Web 开发，尤其是前端开发，你会发现我们正生活在一个激动人心的时代。互联网公司和开发者都在解决开发速度与成本，以及代码质量与用户体验之间的矛盾。

2013 年，Facebook 发布了 React，这是一个用于构建 UI 的开源的 JavaScript 库。可以在 `http://facebook.github.io/react/` 上深入了解 React。2015 年年初，Facebook

的 Tom Occhino 总结了 React 为什么这么“牛”：

> React 将命令式的 API 包装成为声明式的 API，React 牛在改变了我们写代码的方式。

我们知道，声明式编程代码比命令式代码要少。声明式是告诉计算机去做什么而不用管如何去做，命令式则要描述如何去做。JavaScript DOM API 的调用就是一个命令式编程的例子，jQuery 也是。

Facebook 在生产环境中使用 React 已有多年，例如在 Instagram 和其他产品线中。其实 React 同样适用于比较小的项目，比如下面这个购物清单就是用 React 编写的：`http://fedosejev.github.io/shopping-list-react`。我认为 React 是现如今对开发人员来说最好的构建 UI 的 JavaScript 库之一。

本书的目标是让大家理解 React 的基本原理。为此，我们会逐个地讲解概念，同时告诉大家怎么把这些概念落实到应用中。这样一步一步过来，最终将做出一个实时的 Web 应用，在此期间，我们会随时解决新出现的问题，并讨论 React 对这些问题的解决方案。

我们还将了解 Flux，它实现了一个单向的数据流。通过 Flux 和 React，我们能够写出可预料和可管理的应用代码，将来添加代码就可以增加它的功能，而不会增加项目的复杂度。不管你添加多少新功能，这个 Web 应用的工作模式会一直保持不变。

新技术与已往的技术总会有所不同，React 也不例外。实际上，React 的一些核心概念可能与我们的直觉相反，它们会激发我们深度思考，甚至会让我们感觉是在开历史的倒车，但不要急于下结论。正如你所想的，React 这么设计是有足够理由的，毕竟经验丰富的 Facebook 工程师在生产环境下的核心业务开发中构建和使用了 React。我建议大家保持开放的思维来学习 React，我相信在阅读完这本书后，大家会理解并认同这些新概念。

请跟随本书一起开启 React 的学习之旅，听从 Charles F. Kettering 的建议，为我们的未来好好打算打算！

了解我们的项目

我坚信学习新技术的最佳动力是一个让人兴奋的迫不及待想去构建的项目。作为一位经验丰富的开发者，你可能开发过许多成功的商业项目，这些项目共享了一些类似的功能

和设计模式，甚至还包括目标用户。本书将带大家开发一个令人耳目一新的项目，一个你在日常工作中不太可能会做的项目。这个项目不仅是一个有趣的尝试，让你能通过它学到新知识，而且也会满足你的好奇心，并拓展你的想象空间。当然，考虑到你也很忙，所以这个项目不会耗费太长的时间。

这个项目的名字是 **Snapterest**，是一个发现并收集发布在 Twitter 上的公开照片的 Web 应用。可以认为它是一个照片仅来源于 Twitter 的 Pinterest（`www.pinterest.com`）。换句话说，我们将实现一个有以下核心功能的网站：

- 实时收集和展示推文。
- 在集合中添加和删除推文。
- 查看收集的推文。
- 将推文信息导出为一段 HTML 代码以供分享。

当开始开发一个新项目时，要做的第一件事就是准备好所使用的工具。对于这个项目来说，你可能不熟悉我们将要使用的工具，所以让我们先来介绍一下它们，以及如何对其进行安装和配置。

如果在安装和配置这些工具的过程中有任何的问题，可以上 `https://github.com/fedosejev/react-essentials` 提交一个 issue，描述清楚你是怎么做的，得到的错误信息是什么，我会尽力帮助你解决这些问题。

在本书中，我将假定你使用 Mac 或者 Windows 来做开发，如果你是一个 Unix 的开发者，那么你应该非常了解自己的包管理工具，使用它来安装本章中需要的工具应该没什么问题。

让我们从安装 Node.js 开始。

安装 Node.js 和 npm

Node.js（`https://nodejs.org`）是一个允许我们使用熟悉的客户端语言——JavaScript 来编写服务端代码的运行环境。它提供了事件驱动和非阻塞的 I/O 模型，这使得 Node.js 适用于数据密集型和实时的应用程序。这意味着，当在我们的项目中使用 Node.js，一旦在推文消息流到达服务器，我们就可以对消息流进行处理——而这真是我们想要的。

开始安装 Node.js，我们安装的版本是 v0.10.40，因为在写本书的时候 Jest 支持的最高版本的 Node.js 就是 v0.10.40（Jest 是一个 Facebook 出品的测试框架，第 8 章会介绍它）。

访问 `http://nodejs.org/dist/v0.10.40/`并下载适合你的开发系统的安装包。

- OS X：`http://nodejs.org/dist/v0.10.40/node-v0.10.40.pkg`。
- Windows 64-bit：`http://nodejs.org/dist/v0.10.40/x64/node-v0.10.40-x64.msi`。
- Windows 32-bit：`http://nodejs.org/dist/v0.10.40/x64/node-v0.10.40-x86.msi`。

运行并按步骤安装 Node.js，安装完成后，打开终端/命令提示符，输入下面的命令来查看是否安装成功了：

```
node -v
```

如果输出以下结果就表示安装成功：

```
v0.10.40
```

Node.js 有着丰富的模块生态系统来供我们使用，一个模块就是一个可以在应用中反复使用的 Node.js 应用。在本书编写的时候 Node.js 已经有 120 000 多个模块了。Node.js 怎么管理这么多的模块呢？npm 可以满足这个要求，**npm** 是一个管理 Node.js 模块的包管理器，实际上，npm 和 Node.js 会一起被安装进来，所以现在你的环境已经安装了 npm，在终端/命令提示符输入下面的命令来检测：

```
npm -v
```

你应该会看到这样的输出：

```
1.4.28
```

可以在 `www.npmjs.com` 上了解更多 npm 的知识。现在我们已经为 Node.js 的应用安装好了环境。

安装 Git

在本书中，我们通过 Git 来安装 Node.js 的模块，如果你没有安装 Git，请参考 `https://`

git-scm.com/book/en/v2/Getting-Started-Installing-Gitand 来将 Git 安装到你的系统中。

从 Twitter Streaming API 中获取数据

我们的 React 应用将从 Twitter 中获取数据，Twitter 提供了 **Streaming API**，使得任何人都可以从公开的推文信息流中获取 JSON 格式的数据。

要开始使用 Twitter 的 Streaming API 前，你需要先完成以下两步。

1. 创建 Twitter 账号。访问 `https://twitter.com` 并且注册账号，如果你之前注册过，直接登录就可以。
2. 创建 Twitter 应用。访问 `https://apps.twitter.com` 并点击 **Create New App**（创建新应用程序）。你需要填写 **Application Details**（应用详情）这个表单，点击同意 **Developer Agreement**（开发者协议），然后点击 **Create your Twitter application**（创建 Twitter 应用），你将看到自己的应用页面。最后切换到 **Keys and Access Tokens**（密钥和访问令牌）标签页。

在这个页面的 **Application Settings**（应用设置）区域，找到下面两个必要的数据。

- **Consumer Key**（API Key），例如：
  `jqRDrAlKQCbCbu2o4iclpnvem`
- **Consumer Secret**（API Secret），例如：
  `wJcdogJih7uLpjzcs2JtAvdSyCVlqHIRUWI70aHOAf7E3wWIgD`

记录下来这两个数据，本章后面需要它们。

现在我们希望生成 access token。在同样的页面上可以看到 **Your Access Token** 区域是空的，点击 **Create my access token** 会创建下面两部分信息。

- **Access Token**，例如：
  `12736172-R017ah2pE2OCtmi46IAE2n0z3u2DV6IqsEcPa0THR`
- **Access Token Secret**，例如：
  `4RTJJWIezIDcs5VX1PMVZolXGZG7L3Ez7Iz1gMdZucDaM`

同样记录下来这两个值。access token 对于你来说是唯一的，你需要保证它的私密性，不能和任何人分享。

现在我们做好使用 Twitter Streaming API 的准备了。

使用 Snapkite 引擎来过滤数据

从 Twitter Streaming API 获取到的推文数据量会比你需要的数据量大很多，所以我们需要找到一种方法将其中有意义的推文数据过滤并显示出来。我建议你查阅一下 Twitter Streaming API 的文档：`https://dev.twitter.com/streaming`，尤其是这个页面中介绍的方法：`https://dev.twitter.com/streaming/reference/post/statuses/filter`。你会看到 Twitter 只提供了少量的过滤器供我们使用，所以我们需要找到另一种方法来进一步过滤数据。

幸运的是，有一个叫作 **Snapkite 引擎**的 Node.js 应用提供了这个功能。它连接着 Twitter Streaming API，使用内置的或自定义规则的过滤器就可以将过滤后的推文数据输出到一个 web socket 连接上。我们即将开发的 React 应用会直接监听这个 socket 连接上的事件，并在推文到达时直接处理。

我们先安装 Snapkite 引擎。

首先需要克隆 Snapkite 引擎的仓库。克隆就是将程序源代码从 GitHub 上复制到本地目录。在这本书中，我们假设本地目录是 Home 主目录。打开终端/命令提示符，输入下面命令：

```
cd ~
git clone https://github.com/snapkite/snapkite-engine.git
```

这个命令将创建一个 `~/snapkite-engine/` 目录。接下来我们安装 `snapkite-engine` 依赖的其他 Node.js 模块。其中一个是 `node-gyp` 模块。所使用的系统环境（Windows 或 UNIX）不同，需要安装的工具也会不同，工具列表在这个 Web 页面上：`https://github.com/Too TallNate/node-gyp#installation`。

安装完这些工具后，你就可以安装 `node-gyp` 模块了：

```
npm install --global node-gyp
```

现在进入到~/snapkite-engine 目录：

```
cd snapkite-engine/
```

然后执行下面的命令：

```
npm install
```

使用这个命令可以安装 Snapkite 引擎依赖的 Node.js 模块。现在让我们来配置 Snapkite 引擎，假设你在~/snapkite-engine/目录，通过下面的命令将./example.config.json 文件复制到./config.json：

```
cp example.config.json config.json
```

如果你使用的是 Windows，则执行下面的命令：

```
copy example.config.json config.json
```

打开 config.json，编辑配置项，从 trackKeywords 开始。这个配置项设置了需要抓取的关键字，例如，如果需要抓取关键字"my"，我们可以这么更改：

```
"trackKeywords": "my"
```

下一步，我们需要设置 Twitter Streaming API 相关的 key。把 consumerKey、consumerSecret、accessTokenKey 和 accessTokenSecret 4 个值设置为之前创建 Twitter 应用时保存的值，其他属性保持默认值就可以了，如果你想了解这些值分别代表什么含义，请查看与 Snapkite 引擎相关的文档：https://github.com/snapkite/snapkite-engine。

接下来安装 **Snapkite Filters**。Snapkite Filters 是一个根据定义的规则来过滤推文的 Node.js 模块。它有许多种不同的过滤器，我们可以任意组合它们来过滤推文，过滤器列表详见 https://github.com/snapkite/snapkite-filters。

在我们的应用中，需要用到以下 4 个过滤器。

- **Is Possibly Sensitive**：https://github.com/snapkite/snapkite-filter-is-possibly-sensitive
- **Has Mobile Photo**：https://github.com/snapkite/snapkite- filter-has-mobile-photo

- **Is Retweet**: https://github.com/snapkite/snapkite-filter-is-retweet
- **Has Text**: https://github.com/snapkite/snapkite-filter-has-text

我们来安装这些过滤器。首先跳转到~/snapkite-engine/filters/目录：

```
cd ~/snapkite-engine/filters/
```

接着使用下面的命令来克隆这些过滤器：

```
git clone https://github.com/snapkite/snapkite-filter-is-possibly-
sensitive.git
git clone https://github.com/snapkite/snapkite-filter-has-mobile-photo.git
git clone https://github.com/snapkite/snapkite-filter-is-retweet.git
git clone https://github.com/snapkite/snapkite-filter-has-text.git
```

然后设置这些过滤器。为此，我们需要给每个过滤器创建一个 **JSON** 格式的配置文件，并在文件里面定义一些属性。幸运的是，每个过滤器都提供了示例的配置文件，我们可以复制出来，然后根据需要做出相应的更改就可以了。确认你在~/snapkite-engine/filters/目录下，然后执行下面的命令（在 Windows 环境下使用 copy 命令来复制，并且将路径中的正斜线替换成反斜线）：

```
cp snapkite-filter-is-possibly-sensitive/example.config.json snapkite-
filter-is-possibly-sensitive/config.json
cp snapkite-filter-has-mobile-photo/example.config.json snapkite-filter-
has-mobile-photo/config.json
cp snapkite-filter-is-retweet/example.config.json snapkite-filter-is-
retweet/config.json
cp snapkite-filter-has-text/example.config.json snapkite-filter-has-
text/config.json
```

以上这些配置文件不需要我们进行更改，因为这些配置已经符合我们的要求了。

最后，我们需要告诉 Snapkite 引擎哪些过滤器是我们需要用到的，用编辑器打开~/snapkite-engine/config.json 文件，找到下面的内容：

```
"filters": []
```

将这部分修改为：

```
    "filters": [
    "snapkite-filter-is-possibly-sensitive",
    "snapkite-filter-has-mobile-photo",
    "snapkite-filter-is-retweet",
    "snapkite-filter-has-text"
]
```

现在你已经成功安装了 Snapkite 引擎和一些 Snapkite 过滤器。现在让我们检查一下是否可以成功运行。跳转到~/snapkite-engine/然后执行下面的命令：

```
npm start
```

应该不会出现任何错误信息。如果你收到一些错误信息，并且不知道如何修复，请访问 https://github.com/fedosejev/react-essentials/issues，创建一个 issue，将收到的错误信息复制粘贴进去。

接下来，我们来创建项目结构。

创建项目结构

现在开始创建我们的项目结构。组织项目结构听起来像是一个简单的任务，其实不然，一个深思熟虑的项目结构会有助于我们理解应用程序的底层架构。稍后在本书介绍 Flux 应用程序架构时，会有相关示例。首先，在用户根目录下创建项目目录：~/snapterest/。

紧接着，我们在这个目录下创建下面两个子目录。

- ~/snapterest/source/：我们将把 JavaScript 的源文件放在这个目录。
- ~/snapterest/build/：我们将把编译后的 JavaScript 文件和 HTML 文件放在这个目录。

现在，在~/snapterest/source/目录下创建 components/子目录，此时你的项目目录如下：

- ~/snapterest/source/components/
- ~/snapterest/build/

项目的基本结构准备好之后，开始创建应用程序文件。首先，我们在~/snapterest/source/目录下创建最重要的文件 app.js。~/snapterest/source/app.js 文件将是我们的应用程序的入口点。

现在~/snapterest/source/app.js 文件还是空的，我们把它放一边，先来讨论另一个更重要的问题。

创建 package.json

你以前听说过 **DRY** 法则吗？它代表 Don't Repeat Yourself，这个法则促进了软件开发中的一个核心原则——代码复用。最好的代码是不需要你亲自去实现的。事实上，我们这个项目的目标之一就是写尽可能少的代码。你可能没有意识到这一点，但 React 可以帮助我们去实现这一目标。它不仅可以节省现在编写代码的时间，而且在今后维护和改进代码时，它将为我们节省更多的时间。

我们可以运用以下方式来减少代码的书写量：

- 通过声明式的编程风格来书写我们的代码。
- 重用其他人编写的代码。

在这个项目中，我们将使用这两种方式来减少代码的书写量。第一种方式是由 React 本身提供的。React 让我们别无他选，只能使用声明式风格来编写 JavaScript 代码。这意味着 React 会替我们告诉浏览器怎么去做（如同我们使用 jQuery 时的情况），我们只需告诉 React 我们想要什么，如何去做是 React 的事情。这对我们是有利的。

Node.js 和 npm 包含了第二种方式。我在这一章前面提到过，有数十万不同的 Node.js 应用程序可供我们使用。这意味着我们的应用程序所依赖的功能或许已经有人实现过了。

问题是我们怎么知道从哪里可以获得想要复用的 Node.js 应用程序。我们可以通过 npm install <package-name>命令来对其进行安装。在 npm 环境下，一个 Node.js 应用程序被称为一个**包**，每个 **npm 包**都有一个 package.json 文件，这个文件描述了与这个包相关联的元数据。可以在 https://docs.npmjs.com/files/package.json 上了解更多关于 package.json 文件中的字段含义。