Broadview®
www.broadview.com.cn

揭秘Vue2生态结构、实际编程技巧
以组件化编程思想为指导，以前端工程化方法为手段来实践Vue2

Vue2 实践揭秘

梁睿坤 著

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

Vue2 实践揭秘

梁睿坤 著

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

内 容 简 介

本书以 Vue2 的实践应用为根基，从实际示例入手，详细讲解 Vue2 的基础理论应用及高级组件开发，通过简明易懂的实例代码，生动地让读者快速、全方位地掌握 Vue2 的各种入门技巧以及一些在实际项目中的宝贵经验。

本书除了全面、细致地讲述 Vue2 的生态结构、实际编程技巧和一些从实践中得到的经验，还重点介绍如何以组件化编程思想为指导，以前端工程化方法为实现手段来实践 Vue2，通过组件的单元测试和 E2E 测试来保证工程质量。

图书在版编目（CIP）数据

Vue2 实践揭秘 / 梁睿坤著. —北京：电子工业出版社，2017.4
ISBN 978-7-121-31068-3

Ⅰ. ①V… Ⅱ. ①梁… Ⅲ. ①JAVA 语言－程序设计 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字（2017）第 047565 号

责任编辑：陈晓猛
印　　刷：三河市双峰印刷装订有限公司
装　　订：三河市双峰印刷装订有限公司
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：19　字数：400 千字
版　　次：2017 年 4 月第 1 版
印　　次：2017 年 4 月第 1 次印刷
定　　价：79.00 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。

前　言

写作背景

我从事软件开发至今接近 18 年了，在我进入这个行业之前，只有软件工程师与硬件工程师之分，并没有什么前端工程师或者后端工程师之说。前端后端都得懂，否则根本找不到工作。当然，现在对于前端工程师与后端工程师的划分是由于软件技术发展实在太快了，两个方向已经发展成各自独立的体系，前端开发由于技术的革新、移动端的崛起，其地位显得越来越重要。

我是从 jQuery 1.0 开始真正地接受前端工程化开发概念的，也是从那时对 JavaScript 产生了一发而不可收的兴趣。因为前端工程化能使项目的体系结构更加合理，那些在后端实现起来极为繁杂的交互操作以一种最“轻巧”的方式给取代了。当第一次遇到 Angular 之时我更是兴奋不已，它简直就是为传统工程师或者说是后端工程师配备的最强大的前端武器！虽然 Angular 的入门曲线非常陡峭，很多内容都极为晦涩难懂，但它与 jQuery 一样，可以算得上是前端架构发展史上的一座丰碑。

软件领域中后者永远具有更大的吸引力，在前端开发领域，React 可以说是继 Angular 之后又一震撼整个前端开发圈子的重磅炸弹。与 Angular 相比，它大大降低了学习的成本，同时拥有极高效的运行效能，使之一下子盖过了 Angular 的风头。Angular 与 React 两套前端框架的崛起也掀起了整个前端开发圈的一股革命，实际上我们都清楚这是 Google 与 Facebook 之间对开发者的一种争夺手段。对于一直从事实战领域应用的开发者而言，虽然有更多的选择是好事，但“谁更好用？”，“谁更强大？”这类选择困难症也将伴随而来。

Angular 与 React 各有优劣，很难从综合性上来评判谁比谁更好，加上 Angular2 的诞生，使得我们更难以从中选择最合心意的框架了，可能最熟悉的就自然成为最好的了吧。

2016 年我和我的团队所从事的 Web 项目由于需要有大量界面交互功能，因此我引入了 Angular2 + Flask 的搭配方式作为项目的基础语言架构。然而，我的团队大多数是由从事多年后端开发和系统开发的工程师所组成，他们对当下前沿的前端技术涉猎并不算深入，我只能不断地进行内部培训以快速提升团队的前端开发实力。Angular2 一直处于 Beta 状态，

而且相关的官方开发文档一直缺失，开发与测试工具的发展也相对滞后，在实际使用过程中，TypeScript 这个将弱类型化的 JS 强制变成强类型语言的怪胎在不断地给我们制造麻烦，除了让团队接受 Angular2 对 Angular 的**优化模式**，还得不断地在各种**大坑**中求生存，这毫无疑问对于我和我的团队是一种极大的挑战。当时我非常担心由于选择了 Angular2 而导致项目失败，中途曾想过用 React 对之加以取代。但从实际出发，这只是一种换汤不换药的方案而已，直至我们偶然间遇到了 Vue，Vue 可以说给予我们项目生的希望！选中了 Vue 是因为我和我的团队只是付出了极小的代价，甚至可以说是毫无障碍地将 Angular2 上开发的代码切换到 Vue 上面来，Vue 的开发工具链虽说没有 Angular.js 完备，但有 vue-cli 的辅助，也基本能应付项目开发的需要，架构理论上几乎就是对 Angular.js 的简化。更吸引我们的是，这是一个由我们中国人开发的前端框架！而且适合我们项目使用的社区资源也非常丰富，性能、工具链、学习曲线、极小的运行库这些优点一下就完全弥补了 Angular 的不足，也成为了我们项目最后能守住的最坚实的防线。

编撰此书出于一次巧合，我们在升级到 Vue2 之后我一直想找一本能系统化、全面地讲述 Vue2 开发的书籍作为我团队的培训教材，但很可惜一直无法找到。出于一时的心血来潮，突然间想将我们在实践中应用 Vue2 的一些技巧和方法记录下来编撰成书，此时也得到了本书的策划编辑陈晓猛先生给予我的鼓励与支持才得以成书。

此书从构思到成书用了接近 4 个月，实际上花在编撰上的时间估计也只是一个月左右，其他的时间都用在了准备素材与写代码上。本书中的素材都取自我参与过的项目，在此过程中我对 Vue2 的实践应用也有了很大的提高与深化。期望此书能为正在奋斗于前端开发工作的同行们带来帮助，同时也作为我对 Vue 团队的一种支持。Vue 是一款能与世界级的 Angular 与 React 比肩的前端框架，更重要的是它是由我们中国人“智造”的！

内容介绍

本书以 Vue2 的理论为中心，以实战示例为基础，通过示例应用展开覆盖 Vue 的各个理论知识点。本书从实践应用出发，对 Vue 官方未曾进行详尽说明甚至不曾提及的实用内容进行揭秘，试图使此书能成为你在 Vue 前端工程化开发实战中的参考手册。本书主要从多个示例由浅入深地讲述 Vue 的使用知识，除此之外，还重点介绍了 Vue 工程化开发中必备的源码库、第三方开发工具以及如何对 Vue 的各种模块进行全方位的测试。

第 1 章 从一个经典的“待办事项”（TODOs）示例入手，从零开始介绍 Vue 的入门知识，包括插值、数据绑定、属性与样式绑定和组件的基本概念与用法。

第 2 章　讲述如何为 Vue 建立一个真实的工程化开发的环境，以及工程化环境下第三方工具的基本使用与配置，其中包括：vue-cli、webpack、Karma、Phantom、Mocha、Sinon、Chai 和 Nightwatch。

第 3 章　介绍 Vue 的路由机制和 Vue 生态系统中最重要的一个成员——vue-router 的基本使用方法。

第 4 章　通过手机书店示例来介绍组件化理论与 Vue 组件的设计与实现的具体方法，包括抽象组件的基本方法，如何用 Vue 对组件进行封装，如何从界面中提取公共的数据接口，如何在没有实现服务端的情况下运行 Vue 程序以及怎样创建复杂的复合型组件。

第 5 章　全方位地讲述 Vue 的测试与调试过程中使用到技术与工具，包括 Mocha 的使用方法，如何为组件编写单元测试，如何在运行期和单元测试中进行调试，如何进行端对端测试。

第 6 章　通过一个非常普遍且实用的图书管理示例讲述 Vue 在实现一个具有复杂操作的界面时所采用的技术知识点，以及 Vue 组件的高级用法。例如视图的排序、分页、查找，多行删除的设计与实现，通过表单处理图书数据的添加、编辑和数据验证，如何用组件化的设计方法封装 Vue 组件以实现最大限度的组件重用。

第 7 章　介绍 Vue 生态结构中针对规模庞大的前端程序进行状态管理的利器 Vuex，通过实例对 Vuex 的应用原则和结构组成进行一一剖析，讲述如何将各种本来混乱的组件状态通过 Vuex 来将其进行分离，每个部分应该如何设计与编码，如何进行测试，最终使 Vue 前端工程架构变得更为合理。

本书相关源码

- 本书源码汇总——https://github.com/DotNetAge/vue-in-action；
- vue-todos——第 1 章例说 Vue.js 的示例源码 https://github.com/DotNetAge/vue-todos；
- vue-shop——第 3 章路由与页面间导航的示例源码 https://github.com/DotNetAge/vue-shop；
- vue-curd——第 6 章视图与表单的处理的示例源码 https://github.com/DotNetAge/vue-curd；
- V-UIkit——UIkit for Vue2 的组件库，构思源于第 4 章组件化的设计与实现方法的内

容 https://dotnetage.github.io/vue-ui/；

- vue-nvd3——基于 NVD3 开发的 Vue2 的组件库 https://github.com/DotNetAge/vue-nvd3；
- vue-easy-pie-chart——基于 easy pie chart 开发的环状图组件库 https://github.com/DotNetAge/vue-easy-pie-chart。

勘误和交流

本书如有勘误，会在 https://github.com/DotNetAge/vue-in-action 上发布。由于笔者能力有限，时间仓促，书中难免有错漏，欢迎读者批评指正。读者也可以到博文视点官网的本书页面进行交流（www.broadview.com.cn/31068）。注册成为博文视点社区用户，可享受以下服务：

- 下载资源：本书所提供的示例代码及资源文件均可在【下载资源】处下载。
- 提交勘误：您对书中内容的修改意见可在【提交勘误】处提交，若被采纳，将获赠博文视点社区积分（在您购买电子书时，积分可用来抵扣相应金额）。
- 与我交流：在页面下方【读者评论】处留下您的疑问或观点，与我和其他读者一同交流。
- 页面入口：

您也可以直接联系我：

- 博客园：http://www.cnblogs.com/ray-liang
- 简书：http://www.jianshu.com/u/5c81d3d72b56
- 邮箱：csharp2002@hotmail.com
- 微博：广州亚睿
- GitHub：https://github.com/dotnetage

致谢

首先，感谢电子工业出版社博文视点公司的陈晓猛编辑，是您鼓励我将本书付诸成册，并在我写作过程中审阅了大量稿件，给予我很多指导和帮助。感谢工作在幕后的电子工业出版社评审团队对于本书在校对、排版、审核、封面设计、错误改进方面所给予的帮助，使本书得以顺利出版发行。其次，感谢在我十几年求学生涯中教育过我的所有老师，是你们将知识和学习方法传递给了我。感谢我曾经工作过的公司和单位，感谢和我一起共事过的同事和战友，你们的优秀一直是我追逐的目标，你们所给予的压力正是我不断改进自己的动力。

感谢我的父母和儿子。由于撰写本书，牺牲了很多陪伴家人的时间。感谢你们对我工作的理解和支持。

2017 年 2 月 16 日梁睿坤于广州

目　　录

第 1 章　例说 Vue.js

本章将通过极具代表性的 Todo 的示例作为引领读者进入 Vue.js 大门的引子。我会以实践为第一出发点，从零开始一步一步地构造一个单页式的 Todo 应用，在这个过程中会将 Vue.js 相关的知识点融入其中，在实际应用中展现这个“小”而“强”的界面框架。

我们先来看看最终希望构造出一个什么样的 App：

Vue.js 与 Angular2 和 React 相比，让我感觉最舒适的是它在一开始就为我们铺平了入门的道路，这就是它的脚手架 vue-cli。因为它的存在，省去了手工配置开发环境、运行环境和测试环境的步骤，开发者可以直接步入 Vue.js 开发的殿堂。然而，现在我并不打算详细地介绍这个脚手架工具，先让我们一起从使用体验来感性地认识它，在后面的章节中我会详细地介绍这个工具。

在开始动手之前，必须先得在机器上安装好 npm，然后输入以下指令将 vue-cli 安装到机器的全局环境中：

```
$ npm i vue-cli -g
```

然后，我们就可以开始建立工程了，键入以下的指令：

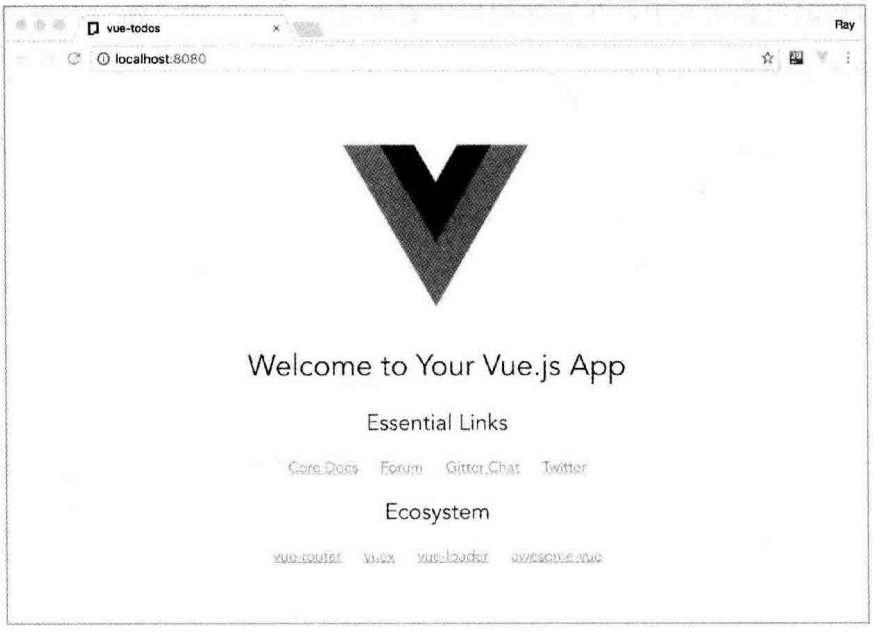
```
$ vue init webpack-simple vue-todos
```

此时控制台会提出一些关于这个新建项目的基本问题，直接“回车”跳过就行了。然后进入 vue-todo 目录，安装脚手架项目的基本支持包：

```
$ npm i
```

安装完支持包后键入以下指令就可以运行一个由脚手架构建的基本 Vue.js 程序了：

```
$ npm run dev
```



是不是很简单？进入代码中看看 vue-cli 到底为我们构造了一个什么样的代码结构：

```
├── README.md
├── index.html              #默认启动页面
├── package.json            # npm 包配置文件
├── src
│   ├── App.vue             # 启动组件
│   ├── assets
│   │   └── logo.png
│   └── main.js             # Vue 实例启动入口
└── webpack.config.js       # webpack 配置文件
```

Vue2 与 Vue1.x 相比有了很大的区别，从最小化的运行程序开始了解 Vue 是一种绝佳的途径，先从 main.js 文件入手：

```
import Vue from 'vue'
import App from './App.vue'

new Vue({
  el: '#app',
  render: h => h(App)
})
```

这里就运用了 Vue2 新增的特色 Render 方法，如果你曾用过 React，是不是有一种似曾相识之感？确实，Vue2 甚至连渲染机制都与 React 一样了。为了得到更好的运行速度，Vue2 也采用了 Virtual DOM。如果你还没有接触过 Virtual DOM，并不要紧，现在只需要知道它是一种比浏览器原生的 DOM 具有更好性能的虚拟组件模型就行了，我们会在稍后的章节中再来讨论它。

我们需要知道的是，通过 import 将一个 Vue.js 的组件文件引入，并创建一个 Vue 对象的实例，在 Vue 实例中用 Render 方法来绘制这个 Vue 组件（App）就完成了初始化。

然后，将 Vue 实例绑定到一个页面上，真实存在的元素 App Vue 程序就引导成功了。

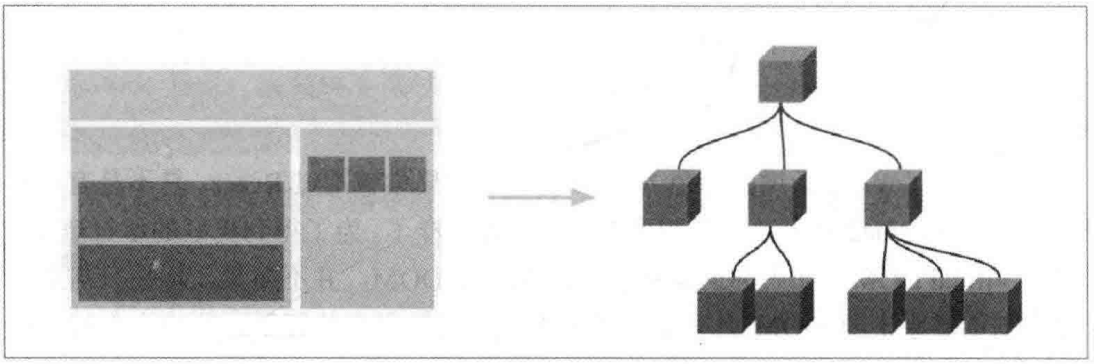
打开 index.html 文件就能看到 Vue 实例与页面的对应关系：

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
</head>
<body>
    <!-- Vue 实例所对应的页面元素 -->
    <div id="app"></div>
    <!-- 由 Webpack 编译后的运行文件 -->
    <script src="/dist/build.js"></script>
</body>
</html>
```

也就是说，一个 Vue 实例必须与一个页面元素绑定。Vue 实例一般用作 Vue 的全局配置来使用，例如向实例安装路由、资源插件，配置应用于全局的自定义过滤器、自定义指令等。在本章示例中，我们只需要知道它的作用就可以了。

我们需要了解的是 App.vue 这个文件，*.vue 是 Vue.js 特有的文件格式，表示的就是一个 Vue 组件，它也是 Vue.js 的最大特色，被称为单页式组件。“*.vue”文件可以同时承载“视图模板”、“样式定义”和组件代码，它使得组件的文件组织更加清晰与统一。

Vue.js 的组件系统提供了一种抽象，让我们可以用独立可复用的小组件来构建大型应用。如果我们考虑到这一点，几乎任意类型应用的界面都可以抽象为一个组件树：



Vue2 具有很高的兼容性，我们也可以用“.js”文件来单纯地定义组件的逻辑，甚至可以使用 React 的 JSX 格式的组件（需要 babel-plugin-transform-vue-jsx 支持）。

脚手架为我们创建的这个 App 组件内加入了不少介绍性的文字，将这个文件“净化”后就可以得到一个最简单的 Vue 组件定义模板：

```
<template>
  <div id="app">
  </div>
</template>

<style></style>

<script>
export default {
  name: 'app'
}
</script>
```

由以上的代码我们可以了解到，单页组件由以下三个部分组成：

- <template>——视图模板；
- <style>——组件样式表；
- <script>——组件定义。

接下来我们就从这个示例开始，一步步学习 Vue 的基本组成部分，在实践中理解它们的作用。

1.1　插值

Vue 的视图模板是基于 DOM 实现的。这意味着所有的 Vue 模板都是可解析的有效的 HTML，而且它对一些特殊的特性做了增强。接下来，我们就在模板上定义一个网页标题，并通过数据绑定语法将 App 组件上定义的数据模型绑定到模板上。

首先，在组件脚本定义中使用 data 定义用于内部访问的数据模型：

```
export default {
  ...
  data () {
    return {
       title: "vue-todos"
    }
  }
}
```

data 可以是一个返回 Object 对象的函数，也可以是一个对象属性，也就是说，可以写成以下的方式：

```
export default {
  ...
  data : {
    title: "vue-todos"
  }
}
```

使用函数返回是为了可以具有更高的灵活性，例如对内部数据进行一些初始化的处理，官方推荐的用法是采用返回 Object 对象的函数。

在模板中引用 data.title 数据时我们并不需要写上 data，这只是 Vue 定义时的一个内部数据容器，通过 Vue 模块的插值方式直接写上 title 即可：

```
<h1>{{ title }}</h1>
```

用双大括号{{ }}引住的内容被称为“Mustache”语法，Mustache 标签会被相应数据对象的 title 属性的值替换。每当这个属性变化时它也会更新。

插值是 Vue 模板语言的最基础用法，很多的变量输出都会采用插值的方式，而且插值还可以支持 JavaScript 表达式运算和过滤器（下文将会提及）。{{}}引用的内容都会被编码，如果要输出未被编码的文本，可以使用{{{}}}对变量进行引用。

完整代码如下所示。

```
<template>
  <div id="app">
    <h1>{{ title }}</h1>
  </div>
</template>

<style></style>

<script>
export default {
  name: 'app',
  data () {
      return {
          title: "vue-todos"
      }
  }
}
</script>
```

> 从 Vue2 开始，组件模板必须且只能有一个顶层元素，如果在组件模块内设置多个顶层元素将会引发编译异常。

请注意，在上述代码中 template 属性是 V，也就是视图，title 属性是 M，也就是模型，这个概念是必须要了解的。

1.2 数据绑定

我们需要一个稍微复杂一点的数据模型来表述 Todo，它的结构应该是这样的：

```
{
    value: '事项 1', // 待办事项的文字内容
    done: false    // 标记该事项是否已完成
}
```

由于是多个事项，那么这个数据模型应该是一个数组，为了能先显示这些待办事项，我们需要先设定一些样本数据。在 Vue 实例定义中的 data 属性中加入以下代码：

```
export default {
    data () {
        return {
            title: 'vue-todos',
            todos: [
```