Web开发经典丛书

Vue.js in Action

Vue.js 开发实战

Chris Fritz作序推荐

[美] 埃里克•汉切特(Erik Hanchett)
本•利斯顿(Benjamin Listwon) 著
任 强 邓龚达 译

MANNING

清华大学出版社

Web 开发经典丛书

Vue.js 开发实战

[美] 埃里克·汉切特(Erik Hanchett)
本·利斯顿(Benjamin Listwon) 著
任　强　邓奎达 译

清华大学出版社
北　京

Erik Hanchett, Benjamin Listwon
Vue.js in Action
EISBN: 978-1-61729-462-4

图书在版编目(CIP)数据

Vue.js 开发实战 / (美)埃里克·汉切特(Erik Hanchett)，(美)本·利斯顿(Benjamin Listwon)著；任强，邓龚达 译. —北京：清华大学出版社，2019
(Web 开发经典丛书)
书名原文：Vue.js in Action
ISBN 978-7-302-53607-9

Ⅰ. ①V…　Ⅱ. ①埃…　②本…　③任…　④邓…　Ⅲ. ①网页制作工具－程序设计　Ⅳ. ①TP393.092.2

中国版本图书馆 CIP 数据核字(2019)第 174350 号

责任编辑：王　军
封面设计：孔祥峰
版式设计：思创景点
责任校对：牛艳敏
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：涿州市京南印刷厂
经　　销：全国新华书店
开　　本：170mm×240mm　　**印　　张**：18　　**字　　数**：384 千字
版　　次：2019 年 9 月第 1 版　　**印　　次**：2019 年 9 月第 1 次印刷
定　　价：79.80 元

产品编号：082441-01

译 者 序

随着互联网的蓬勃发展，前端技术也百花齐放，出现了许多优秀的、应用广泛的前端框架。选择哪个前端框架，是入门者到专家所有层次开发人员都需要思考的问题。本书虽然并不对众多前端框架做比较，也不帮你做决策；但如果你想初探甚至深入学习 Vue.js，本书会提供你需要的所有帮助。本书不仅深入浅出地讲解 Vue.js 的相关概念和技术，还准备了一系列小而全的示例来帮助你融会贯通。

非常荣幸清华大学出版社能给我翻译本书的机会，李阳编辑是指引我进入翻译领域的老师。编辑老师们一如既往、不厌其烦地帮助并指导我们，他们为本书的翻译付出了很多心血，没有他们，本书不可能顺利付梓。感谢同事邓龚达欣然接受我的邀请，与我合作翻译本书，没有他，我无法在这么短时间内完成本书的翻译。另外感谢我的同事叶盛飞和周骅，他们校对了部分重要章节，并给予我许多宝贵的指导建议。

最后，希望你能通过阅读本书提升 Vue.js 技能，创建出更多丰富多彩的应用程序。

任　强

2019 年 3 月 31 日

序　言

前端 Web 开发已经变得异常复杂。如果你从未使用过现代的 JavaScript 框架，构建你的第一个只显示“Hello”的应用程序可能需要一个星期！这听起来可能很荒谬，我也认为如此。问题是，大多数框架都假定你已对终端、JavaScript 高级技术以及诸如 Node 包管理器(Node Package Manager，NPM)、Babel、Webpack 等工具有一定的了解，甚至知道更多。

令人振奋的是，Vue 并没有这样的束缚。我们称之为“渐进式”JavaScript 框架，因为它既可以向下扩展，也可以向上扩展。如果你的应用程序很简单，你可以像使用 jQuery 那样使用 Vue：通过放入一个<script>标签。但是，随着技能的提升和需求的增多，Vue 也会随着你的成长而变得更加强大和高效。

Vue 不仅是由计算机科学家建立的，而且是由设计师、教育工作者和更多以人为中心的行业的其他人建立的。因此，我们的文档、指南和开发工具都是世界一流的。Vue 的使用经验与其性能、可靠性和多用途对我们同样重要。

Erik 把以人为本的精神注入本书。首先，本书非常直观。许多详细的插图和带注释的屏幕截图在真实开发人员的工作流程中牢固地奠定了示例的基础。因此，实际上你可以学习如何使用浏览器和 Vue 的开发工具来确认正在学习的内容，更重要的是，在出现问题时进行故障排除。

对于那些在前端开发、JavaScript 甚至编程方面没有很强背景的人，Erik 还仔细解释了理解 Vue 在做什么以及为什么这样做的基本概念。这与他以项目为中心引入新特性的方法相结合，意味着这本书对于初出茅庐的开发人员来说是理想的，他们希望通过 Vue 作为自己的第一个现代前端框架来扩展技能。

——Chris Fritz，Vue 核心团队成员和文档管理员

作者简介

Erik Hanchett 是一位拥有超过 10 年开发经验的 Web 开发人员。他是 *Emberjs Cookbook* (Packt Publishing，2016)的作者，也是 YouTuber(http://erik.video)和博主(http://programwitherik.com)。他运行了一个邮件列表，通过 https://goo.gl/UmemSS 为 JavaScript 开发人员提供提示和技巧。在工作或编码之余，他会和妻子 Susan 以及两个孩子 Wyatt 和 Vivian 共度时光。

致　谢

首先，同时也是最重要的，我要感谢我的妻子 Susan，因为没有她的帮助，这本书永远不会完成。我要感谢我的儿子 Wyatt 和女儿 Vivian。他们是我如此努力工作的动力源泉。我还要感谢所有的评论家、Vue.js in Action 论坛的成员，以及任何有助于对这本书给予反馈的人。你们提供的帮助使得这本书远比我一个人所能做到的要好得多。另外，Chris Fritz，感谢你写了一篇精彩的序言。最后，我衷心感谢 Vue.js 社区、Evan You 以及所有让 Vue.js 成为如此卓越框架的人。

—Erik Hanchett

首先，我要向我的妻子 Kiffen 表示最诚挚的感谢，感谢她对我的支持和鼓励，不仅感谢让我参与这项工作，还感谢她对我们生活各个方面的支持和鼓励。对我们的儿子 Leo，我们家族宇宙中心的明星，我要感谢你无限的微笑、拥抱和欢呼。感谢他们的鼓励、理解和支持。我衷心感谢 Manning 的编辑团队。关于 Erik，没有他这本书就不会有生命，我深表感谢；我祝福你一切顺利。最后，感谢 Evan You 和所有为 Vue.js 做出贡献的人，感谢他们汇集大量的软件并组成一个伟大的社区。我很荣幸能成为这个社区的一员。

—Benjamin Listwon

我们都应该感谢我们的技术校对员 Jay Kelkar，以及一直以来提供反馈的所有审稿人，包括 Alex Miller、Alexey Galiullin、Chris Coppenbarger、Clive Harber、Darko Bozhinovski、Ferit Topcu、Harro Lissenberg、Jan Pieter Herweijer、Jesper Petersen、Laura Steadman、Marko Letic、Paulo Nuin、Philippe Charriere、Rohit Sharma、Ronald Borman、Ryan Harvey、Ryan Huber、Sander Zegveld、Ubaldo Pescatore 和 Vittorio Marino。

前　言

2017 年年初，在 Benjamin Listwon 因个人原因退出之后，我有机会撰写这本书。我最近刚从雷诺内华达大学获得了工商管理硕士学位，自从我出版上一本书 *Ember.js Cookbook* (Pact Publishing，2016)以来已经整整一年了。我已经开始了我的 YouTube 频道，和 Erik 合作，我花了大部分时间试图找出如何更好地为我的小规模但人数不断增长的观众录制节目教程。大约在这段时间，我开始在 Vue.js 上制作一个电影系列，并从我的观众那里得到了积极反馈。这让我更想探索 Vue.js。

首先，我会倾听 Evan You(Vue.js 的创建者)及其框架路线图。然后我看了无数的 YouTube 教程和其他创建者的视频。我访问了在线论坛和 Facebook 群组，看看人们在谈论什么。无论我到哪里，人们都对 Vue.js 框架的可能性感到兴奋。这使我想探索写这本书的可能性。

经过深思熟虑，并且与我妻子谈话后，我决定去做这件事。幸运的是，Benjamin 为我打下很好的基础，所以我可以在此基础上进行创作。在接下来的 10 个月里，我花了无数的夜晚和周末研究、测试和写作。

我希望我能告诉你写这本书多么容易，或者说我没有遇到任何问题。但事情并没有按计划进行。我遇到了个人挫折，错过了最后期限，遇到了写作障碍，如果这还不够的话，我最终不得不在 Vue.js 做了一次更新之后进行重大修改。

尽管如此，我还是为这本书感到骄傲。每次挫折，我都被激励加倍努力。我决心以最高质量完成这本书。

谢谢读者，非常感谢你购买了本书。我真的希望它能帮助你学习 Vue.js。如果真的帮到了你，请告诉我。你可以在@ErikCH 上发推特，在 erik@programwitherik.com 上发邮件给我，或者在 https://goo.gl/UmemSS 上加入我的邮件列表！再次感谢！

关 于 本 书

在你学习如何编写 Vue.js 应用程序之前，让我们先介绍一下你应该了解的一些事项。

在本书中，我们将介绍你所需了解的所有知识来让你熟悉 Vue.js。本书的目标是为你提供所需的知识，因此你可以毫不犹豫地跳到任何 Vue.js 应用程序中。

在为本书做研究时，我无数次听到官方的 Vue.js 指南是学习 Vue.js 的最佳资源。虽然官方指南很棒，但我强烈建议你在学习 Vue.js 时将它们作为附加参考，但它们并不涵盖所有内容，并且它们并不完美。本书涵盖的内容超越官方指南，并且书中的示例更容易理解、可关联，因此你可以更轻松地将这些概念应用到你自己的项目中。我认为某个主题超出了本书的讨论范围，或者说不够重要，因此我添加了一份参考资料，你可以在官方指南中了解更多相关内容。

本书可以有几种不同的阅读方式。你可以从前到后阅读。在这种情况下，你将获得 Vue.js 所提供技能的全部广度。或者你也可以使用本书作为参考手册来查找你需要掌握更多信息的那些概念。无论哪种方式都是可以接受的。

在本书的后面部分，我们将转换为使用构建系统创建 Vue.js 应用程序。别担心，附录 A 中包含如何使用 Vue.js 构建工具 Vue-CLI。Vue-CLI 最重要的好处之一是，它可以帮助我们创建更复杂的 Vue.js 应用程序，而不必担心构建或转换我们的代码。

贯穿本书，我们将创建一个 Vue.js 宠物商店应用程序。某些章节比其他章节更多地使用这个宠物商店示例应用程序。我刻意这么做，这样你可以很容易地掌握一个概念，而无须了解它如何与这个宠物商店示例应用程序共存。但是，那些喜欢用真实应用程序学习的人仍然有选择的权利。

读者对象

本书适合任何有兴趣学习 Vue.js 并具有 JavaScript、HTML 和 CSS 经验的人。我不希望你对此有太多了解，但了解基础知识，如数组、变量、循环和 HTML 标签，将会有所帮助。至于 CSS，我们将使用 Bootstrap 3，它是一个 CSS 库。但是，你不需要了解 Bootstrap 的任何内容以跟随示例。它只是用于帮助构建样式。

在本书的前面部分，我使用 ECMAScript 2015(也称为 ES6)介绍了示例代码。在开始阅读本书之前，最好先查看一下。在大多数情况下，我只使用一些 ES6 功能，例

如箭头函数和 ES6 导入。当我们进行这种转变时，我会在书中警告你。

学习路线图

本书内容分为三部分，每部分都建立在前一部分的基础之上。第 I 部分是了解 Vue.js 的关键。在第 1 章和第 2 章中，我们将创建第一个 Vue.js 应用程序。我们将看看 Vue.js 实例是什么及其与应用程序的关系。

在第 II 部分(第 3～9 章)，我们将深入了解 Vue.js 的几个最重要的部分。第 I 部分更多是 Vue.js 的开胃菜，而第 II 部分是主菜。你将学习如何创建 Vue.js 应用程序的复杂性。我们将首先学习反应模型，然后将创建一个宠物商店应用程序，我们将在本书的其余部分使用它。

我们将添加表单和输入框，还将介绍如何使用 Vue.js 强大的指令绑定信息，然后查看条件、循环和表单。

第 6 章和第 7 章非常重要。我们将学习如何使用组件将 Vue.js 应用程序分成几个逻辑部分，我们将先看一下创建 Vue.js 应用程序所需的构建工具。

第 7 章还将介绍路由。在前面的章节中，我们使用简单的条件来导航应用程序。通过添加路由，我们可以正确地在应用程序中移动并在路由之间传递信息。

第 8 章介绍使用 Vue.js 可以执行的强大动画和转场。这些功能都融入了语言，是你应该掌握的很好功能。

在第 9 章中，我们将学习如何使用 Mixin 和自定义指令轻松扩展 Vue 而不用重复自己。

第III部分是关于建模数据、使用 API 和测试的全部内容。在第 10 章和第 11 章中，我们首先深入探讨 Vue 的状态管理系统 Vuex。然后我们将看看如何开始与后端服务器进行通信，我们将了解有关服务器端呈现框架 Nuxt.js 的更多信息。

第 12 章专门用于测试。在任何专业环境中，你都需要了解测试，我们将介绍你必须了解的基本要素。

书籍论坛

购买本书后就可以免费访问由 Manning Publications 运营的私人网络论坛，可以在该论坛上对本书发表评论，提出技术问题，并从作者和其他用户那里获得帮助。要访问该论坛，可访问 https://forums.manning.com/forums/vue-js-in-action。还可以访问 https://forums.manning.com/forums/about，了解有关 Manning 论坛和行为规则的更多信息。

源代码

本书包含许多源代码示例，包括编号列表和内联普通文本。在这两种情况下，源代码都以这样的固定等宽字体格式化，以与普通文本分开。有时使用粗体突出显示已从本章前面的步骤中更改的代码，例如当新功能被添加到现有代码行时。

在许多情况下，源代码已经重新格式化；我们添加了换行符并重写缩进以适应书中可用的页面空间。在极少数情况下，如果这还不够，列表将包括行继续标记(➥)。此外，当使用文本描述代码时，源代码中的注释通常已从代码清单中删除。代码注释伴随着许多代码清单，从而突出了重要的概念。

本书的源代码可从出版商的网站(www.manning.com/books/vue-js-in-action)和我的个人 GitHub 存储库(https://github.com/ErikCH/VuejsInActionCode)下载，也可通过扫描封底的二维码下载。你还可以在附录 A 中找到有关下载代码和设置编程环境的更多说明。

在阅读本书时，你会注意到我经常将源代码拆分为单独的文件。我已将完成的文件和每章中的分隔文件都包含在源代码中，因此你可以按照这些步骤进行操作。如果你在代码中发现错误，请随意通过 pull request 发送到我的 GitHub 存储库。我会维护该存储库，并在自述(README)文件中发表任何更新的评论。

软件要求

为简单起见，本书中的所有代码都适用于任何现代浏览器。我已经在 Firefox 58、Chrome 65 和 Microsoft Edge 15 上亲自测试过。我不建议尝试在旧浏览器上运行我的任何应用程序，因为你肯定会遇到问题。Vue.js 本身不支持 IE 8 及以下版本。你必须具有符合 ECMAScript 5 标准的浏览器。

在本书的前几章中，我使用了一些 ES6 特性。你需要使用现代 Web 浏览器来运行这些示例。

我们将在本书中创建的宠物商店应用程序将在移动浏览器上运行。然而，宠物商店应用程序并未针对移动设备进行优化，因此我建议你在台式计算机上运行这些示例。

你不必担心操作系统。如果 Web 浏览器运行了，就应该没问题。实际上没有其他要求。

在线资源

正如我之前提到的，当你使用本书中的示例时，Vue.js 官方指南非常适合用作参考资料。你可以在 https://vuejs.org/v2/guide/上找到指南。它们在不断更新。

在 GitHub 页面 https://github.com/vuejs/awesome-vue 上有一个与 Vue.js 相关的精彩内容的精选列表。在这里，你可以找到 Vue.js 播客(Podcast)、其他 Vue.js 资源、第三方库，甚至是使用 Vue.js 的公司的链接。强烈建议你查看一下。

Vue.js 社区规模庞大且不断发展。与其他 Vue.js 开发人员交流的最佳地点之一是 https://forum.vuejs.org/上的官方 Vue.js 论坛。在这里可以讨论或获得任何关于 Vue 的帮助。

如果你正在寻找更多视频教程，我的频道——YouTube 上的 http://erik.video——涵盖了大量有关 Vue.js 和 JavaScript 的信息。

更多信息

这本书中覆盖了大量的材料。如果你遇到了困难，或者你需要帮助，请不要犹豫，与我联系。如果我无法帮助你，至少我会指出正确的方向。别害羞。你会发现 Vue.js 社区中的人们对初学者是很友好的。

此外，在阅读本书时，请尝试采用你学到的几个概念并自行实施。最好的学习方法之一就是实践。例如，尝试创建自己的电子商务网站，而不是参照宠物商店应用程序。使用本书作为指导，以确保你不会陷入困境。

最后一件事：学得开心。要有创意，做一些很炫酷的事情。如果你这样做了，一定要在 Twitter 上通过@ErikCH 联络我！

目　录

第 I 部分　初识 Vue.js

第 II 部分　视图与视图模型

第 I 部分

初识Vue.js

在开始学习 Vue 的炫酷功能之前，我们需要先了解 Vue。在前两章中，将研究 Vue.js 背后的原理、MVVM 模式及其与其他框架的关联方式。

在理解了 Vue 的来源后，我们将继续深入研究 Vue 实例。Vue 根实例是应用程序的核心，我们将探索它是如何构成的。稍后，还将研究如何在 Vue 应用程序中绑定数据。

这些章将为你的学习提供一个良好开端。你将学习如何创建一个简单的 Vue 应用程序并了解 Vue 是如何工作的。

第 1 章

Vue.js介绍

本章涵盖：

- 探讨 MVC 和 MVVM 设计模式
- 介绍反应式应用程序
- 介绍 Vue 的生命周期
- 评价 Vue.js 的设计

交互式网站已经存在了很长时间。21 世纪中期，当 Web 2.0 还处于早期阶段时，人们更关注的是交互性和如何吸引用户使用。推特(Twitter)、脸书(Facebook)和优兔(YouTube)等公司都是在这个时期诞生的。社交媒体和用户原创内容(User-Generated Content，UGC)的兴起改变了互联网的发展方向，使其变得更丰富多彩。

开发人员必须跟上这些变化，为最终用户提供更多的交互性。在早期，各种库和框架使构建交互式网站变得更容易。2006 年，John Resig 发布了 jQuery，极大地简化了 HTML 中的客户端脚本。随着时间的推移，客户端框架和库出现了。

起初这些框架和库都很庞大而笨重，没有模块化且不灵活通用。现在这些库正在向更小、更轻量化的方向转变，可以很容易地集成到任何项目中。Vue.js 就是在这时出现的。

有了 Vue.js，就可以在任何 JavaScript 能运行的地方增加交互行为和功能。Vue 既可以用于完成个别页面的简单任务，也可以为企业级应用提供基础框架。

提示 术语 Vue 和 Vue.js 通常可以互换。在本书中，多数情况下我会使用更通俗的 Vue，而当引用代码和库时将使用 Vue.js。

从提供用户交互的界面到为我们的应用提供数据的数据库，我们将探讨 Vue 以及相关支持库是如何让我们能够构建完整复杂的 Web 应用的。

在此过程中，我们将研究每一章的代码如何适应于全局，哪些行业最佳实践是可以借鉴的，以及如何将这些结合到你自己现有的或新的项目中。

本书主要面向对 JavaScript 有一定了解并熟悉 HTML 和 CSS 的 Web 开发人员。因为 API 的多样性，Vue 将会使你和你的项目得到长足发展。任何想要构建原型或个人项目的人都可以把本书当作指南。

1.1 站在巨人的肩膀上

在我们为第一个应用程序编写第一行代码之前，甚至是在深入探究 Vue 之前，很有必要先了解一些软件相关的历史。如果不了解 Web 应用程序在过去曾经面临过的问题和挑战，就很难真正领会到 Vue 为我们带来了什么以及 Vue 的优势。

1.1.1 MVC 模式

客户端的模型-视图-控制器(Model-View-Controller，MVC)模式提供的架构蓝图，已经被众多的现代 Web 应用开发架构所采用，这充分证明了它的实用性(如果你熟悉 MVC，可以跳过本节内容)。

值得一提的是，原始的 MVC 设计模式多年来一直在改变。有时我们称之为经典 MVC(Classic MVC)，它涉及一组单独的视图、控制器和模型之间如何交互的规则集。为简单起见，我们将讨论客户端 MVC 模式的简化版本。这种模式是对 Web 应用程序的一种更现代的解释。

如图 1.1 所示，该模式用于分离应用程序的层级关系。视图负责向用户显示信息，体现为图形用户界面(Graphical User Interface，GUI)。控制器在中间，它有助于将事件从视图转换为模型，将数据从模型转换为视图。最后，模型保持业务逻辑并且包含数据存储。

信息　如果你有兴趣了解关于 MVC 模式的更多信息，请访问 Martin Fowler 的关于 MVC 演变的网页，网址为 https://martinfowler.com/eaaDev/uiArchs.html。

许多 Web 框架的作者之所以使用这种 MVC 模式的变体，正是因为它的坚实且经受过时间考验。如果你想了解更多关于现代 Web 框架是如何设计和架构的，请查看 Emmitt A.Scott Jr.撰写的 *SPA Design and Architecture* (Manning，2015)。

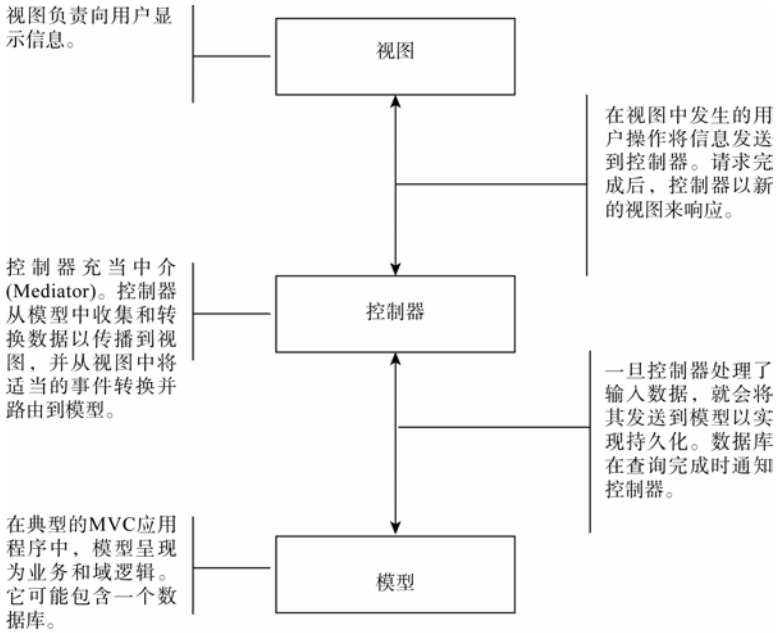

图 1.1　MVC 模式中模型、视图和控制器的角色

在现代软件开发中，MVC 模式通常作为单应用程序的组成部分，并且为分离应用程序代码的不同角色提供了良好的机制。对于使用 MVC 模式的网站，每个请求都会启动从客户端到服务器、再到数据库的信息流，最后依次返回。该过程耗时且资源密集，并且不提供响应式用户体验。

多年以来，开发人员通过使用异步 Web 请求和客户端 MVC 来提升 Web 应用程序的交互性，因此发送给服务器的请求是非阻塞性的，能持续操作而无须等待返回。但是随着 Web 应用程序像桌面应用程序一样工作，任何对客户端/服务器交互的等待都会让人感觉应用程序迟缓或断续。这就需要下一个设计模式来拯救。

关于业务逻辑

当你考虑应该在哪里实现业务逻辑时，你会发现客户端 MVC 模式有相当大的灵活性。在图 1.1 中，为简化起见，我们在模型中合并了业务逻辑，但是它也可能存在于应用程序的其他层中，包括控制器在内。自从 1979 年 Trygve Reenskaug 在 Smalltalk-76 中引入 MVC 模式以来，MVC 模式已经发生了许多变化。

下面考虑对用户输入的邮政编码进行验证：

- 视图可能需要包含 JavaScript，在输入时或提交之前对邮政编码进行验证。
- 当创建用于保存输入数据的地址对象时，模型可能也需要验证邮政编码。
- 对于邮政编码字段上的数据库约束，可能意味着模型还会强制执行业务逻辑，尽管这可能会被认为是不好的实践。

很难定义什么构成了实际的业务逻辑，并且在许多情况下，所有之前的约束都可能会最终在某个请求中产生作用。

当我们在本书中构建应用程序时，将研究如何以及在何处组织业务逻辑，还将研究 Vue 及其支持库是如何帮助我们避免令人讨厌的功能跨界的。

1.1.2 MVVM 模式

当 JavaScript 框架开始支持异步编程技术时，Web 应用程序不再需要请求完整的网页。网站和应用程序可以通过部分更新视图来更快地响应，但这样做需要一定程度的重复劳动。用于呈现的代码逻辑通常会重复包含业务逻辑。

模型-视图-视图模型(MVVM)模式是对 MVC 模式的重新定义，最主要的区别是引入了视图模型(view-model)及其数据绑定(统称为绑定)。MVVM 模式为我们构建客户端应用程序提供了更多的反应式用户交互和反馈，同时在整体架构上避免了代价高昂的重复代码和重复劳动。单元测试也变得更容易。尽管如此，对于非常简单的 UI 来说，使用 MVVM 可能会有些大材小用，这点也需要充分考虑到。

对于 Web 应用程序，MVVM 的设计让我们编写的软件可以即时响应用户交互，并且允许用户自由地从一个任务移动到下一个任务。通过图 1.2 可以看出，视图模型(view-model)也充当着不同角色。这种职责的整合使得应用程序的视图的职责变得单一而重要：当视图模型中的数据发生改变时，任何与其绑定的视图都要自动刷新。数据绑定器负责提供数据，并且当数据发生变化时确保数据能反映到视图中。

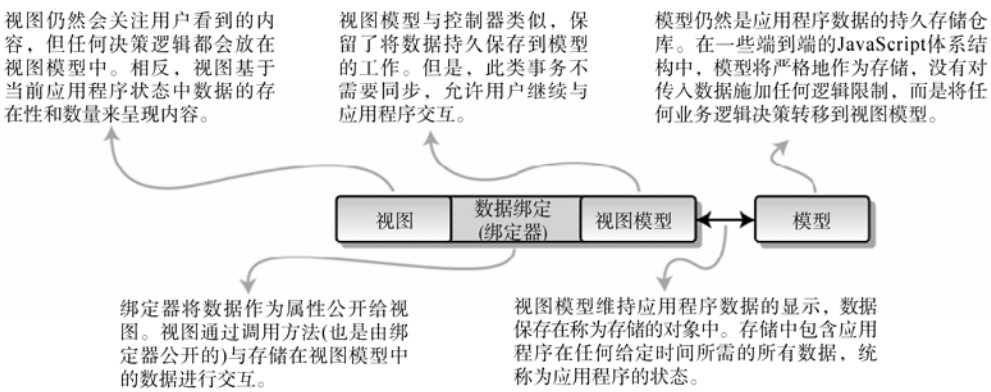

图 1.2 模型-视图-视图模型(MVVM)模式中的组件

信息　你可以访问 Martin Fowler 的关于 Presentation 模型的网页，了解关于 MVVM 模型的更多信息，网址为 https://martinfowler.com/eaaDev/PresentationModel.html。

1.1.3　什么是反应式应用程序

反应式编程范例并不是一个新概念。但它在 Web 应用程序上的应用相对较新，并且很大程度上归功于 JavaScript 框架(诸如 Vue、React 和 Angular 等)的可用性。

网上有许多关于反应式理论的优秀资源，但我们的需求可能更加集中。对于一个具备响应性的 Web 应用程序，应该：

- 监听应用程序状态的变化
- 在整个应用程序内广播变化通知
- 根据状态的变化自动渲染视图
- 为用户交互提供及时反馈

反应式 Web 应用程序通过采用 MVVM 设计原则，使用异步技术避免阻塞持续性交互，尽可能地运用函数式编程风格，从而实现了以上目标。

虽然 MVVM 模式并不意味着反应式应用程序，反之亦然，但它们具有共同的意图：为用户提供响应更快、更可靠的体验。

信息　如果你想了解关于 Vue 反应式编程范例的更多信息，请查看 *Reactivity in Depth* 指南，网址为 https://vuejs.org/v2/guide/reactivity.html。

1.1.4　JavaScript 计算器

为了更好地理解数据绑定和反应式的概念，我们将首先在简单的、Vanilla JavaScript(Vanilla JavaScript 是指浏览器提供的原始 JavaScript 功能)中实现一个计算器，如代码清单 1.1 所示。

代码清单 1.1　JavaScript 计算器：chapter-01/calculator.html

```
<!DOCTYPE>
<html>
  <head>
    <title>A JavaScript Calculator</title>
    <style>
     p, input { font-family: monospace; }
     p, { white-space: pre; }
    </style>
  </head>
  <!-- Bind to the init function -->
  <body>
    <div id="myCalc">
```

借助表单输入采集 x 和 y，绑定 runCalc 函数

```
    <p>x <input class="calc-x-input" value="0"></p>
    <p>y <input class="calc-y-input" value="0"></p>
    <p>--------------------</p>
       <p>= <span class="calc-result"></span></p>
</div>
<script type="text/javascript">
 (function(){

   function Calc(xInput, yInput, output) {
     this.xInput = xInput;
     this.yInput = yInput;
     this.output = output;
   }

   Calc.xName = 'xInput';
   Calc.yName = 'yInput';

   Calc.prototype = {
     render: function (result) {
       this.output.innerText = String(result);
     }
   };
   function CalcValue(calc, x, y) {
     this.calc = calc;
     this.x = x;
     this.y = y;
     this.result = x + y;
   }

   CalcValue.prototype = {
     copyWith: function(name, value) {
       var number = parseFloat(value);

       if (isNaN(number) || !isFinite(number))
         return this;

       if (name === Calc.xName)
         return new CalcValue(this.calc, number, this.y);

       if (name === Calc.yName)
         return new CalcValue(this.calc, this.x, number);

       return this;
     },
     render: function() {
       this.calc.render(this.result);
     }
   };

   function initCalc(elem) {
```

显示 x 和 y 的结果

创建 calc 实例的构造函数

为 calc 实例赋值的构造函数

初始化 calc 组件

```
    var calc =
      new Calc(
        elem.querySelector('input.calc-x-input'),
        elem.querySelector('input.calc-y-input'),
        elem.querySelector('span.calc-result')
      );
    var lastValues =
      new CalcValue(
        calc,
        parseFloat(calc.xInput.value),
        parseFloat(calc.yInput.value)
      );

    var handleCalcEvent =                          事件处理器
      function handleCalcEvent(e) {
        var newValues = lastValues,
          elem = e.target;

        switch(elem) {
          case calc.xInput:
            newValues =
             lastValues.copyWith(
                Calc.xName,
                elem.value
             );
            break;
          case calc.yInput:
            newValues =
              lastValues.copyWith(
                Calc.yName,
                elem.value
          );
        break;
        }

        if(newValues !== lastValues){
          lastValues = newValues;
          lastValues.render();
        }
      };
     elem.addEventListener('keyup', handleCalcEvent, false);
return lastValues;
}
window.addEventListener(
'load',
function() {
        var cv = initCalc(document.getElementById('myCalc'));
        cv.render();
```

设置 keyup 事件侦听器

```
            },
            false
        );

    }());
  </script>
 </body>
</html>
```

此计算器使用的是 ES5 JavaScript(本书后面将使用更加现代化的版本 JavaScript ES6/2015)。我们使用一个即时调用的函数表达式来启动 JavaScript。构造函数用于保持值以及处理所有 keyup 事件的 handleCalcEvent 函数。

1.1.5 Vue 计算器

不要过分担心 Vue 示例的语法，因为我们的目标不是要理解代码中发生的一切，而是要比较这两个实现。也就是说，如果你对 JavaScript 示例(如下面的代码清单 1.2 所示)的工作原理有很好的了解，那么大多数 Vue 代码至少应该在理论层面上是有意义的。

代码清单 1.2 Vue 计算器：chapter-01/calculatorvue.html

```
<!DOCTYPE html>
<html>
<head>
  <title>A Vue.js Calculator</title>
  <style>
    p, input { font-family: monospace; }
    p { white-space: pre; }
  </style>
</head>
<body>
  <div id="app">
    <p>x <input v-model="x"></p>
    <p>y <input v-model="y"></p>
    <p>----------------------</p>
    <p>= <span v-text="result"></span></p>
  </div>
  <script src="https://unpkg.com/vue/dist/vue.js"></script>
  <script type="text/javascript">
  function isNotNumericValue(value) {
    return isNaN(value) || !isFinite(value);
}
    var calc = new Vue({
      el: '#app',
      data: { x: 0, y: 0, lastResult: 0 },
```

应用程序的 DOM 锚

应用程序的表单输入框

结果将会显示在这个 span 标签内

添加 Vue.js 库的 script 标签

初始化应用程序

连接到 DOM

添加到应用程序的变量

```
    computed: {                                  <-- 这里用 computed 属性来计算结果
      result: function() {
          let x = parseFloat(this.x);
          if(isNotNumericValue(x))
            return this.lastResult;
          let y = parseFloat(this.y);
          if(isNotNumericValue(y))
            return this.lastResult;
          this.lastResult = x + y;
          return this.lastResult;
      }
    }
  });
</script>
</body>
</html>
```

1.1.6　JavaScript 和 Vue 的差别

两种计算器实现的代码在很大程度上是不同的。图 1.3 中显示的每个示例都可以在本章对应的代码仓库中找到，因此你可以运行每个示例并比较它们的运行方式。

```
<script type="text/javascript">
 (function(){

   function Calc(xInput, yInput, output) {
     this.xInput = xInput;
     this.yInput = yInput;
     this.output = output;
   }

   Calc.xName = 'xInput';
   Calc.yName = 'yInput';

   Calc.prototype = {
     render: function (result) {
       this.output.innerText = String(result);
     }
   };

   function CalcValue(calc, x, y) {
     this.calc = calc;
     this.x = x;
     this.y = y;
     this.result = x + y;
   }
```

```
<script src="https://unpkg.com/vue/dist/vue.js"></script>
<script type="text/javascript">
function isNotNumericValue(value) {
  return isNaN(value) || !isFinite(value);
}
  var calc = new Vue({
    el: '#app',
    data: { x: 0, y: 0, lastResult: 0 },
    computed: {
      result: function() {
        let x = parseFloat(this.x);
        if(isNotNumericValue(x))
        return this.lastResult;

        let y = parseFloat(this.y);
        if(isNotNumericValue(y))
        return this.lastResult;

        this.lastResult = x + y;

        return this.lastResult;
      }
    }
  });
</script>
```

图 1.3　使用原生 JavaScript(左图)和 Vue(右图)编写的反应式计算器的并排对比

两个应用程序之间的关键区别在于如何触发最终计算结果的更新以及计算结果如何找到返回页面的方式。在我们的 Vue 示例中，通过一个 v-model 绑定来负责页面上的所有更新和计算。当我们使用 new Vue({...})实例化我们的应用程序时，Vue 会检查我们的 JavaScript 代码和 HTML 标签，然后创建运行应用程序所需的所有数据和事件绑定。

1.1.7 Vue 如何促进 MVVM 和响应性

Vue 有时被称为渐进式框架(progressive framework)，渐进式框架广义地意味着它可以被整合到某个现有的网页中用于简单的任务，或者可以完全用作大规模 Web 应用程序的基础。

无论你选择如何将 Vue 集成到项目中，每个 Vue 应用程序将具有至少一个 Vue 实例(instance)。最基本的应用程序将具有一个实例，该实例在特定标签与视图模型中存储的数据之间提供绑定(参见图 1.4)。

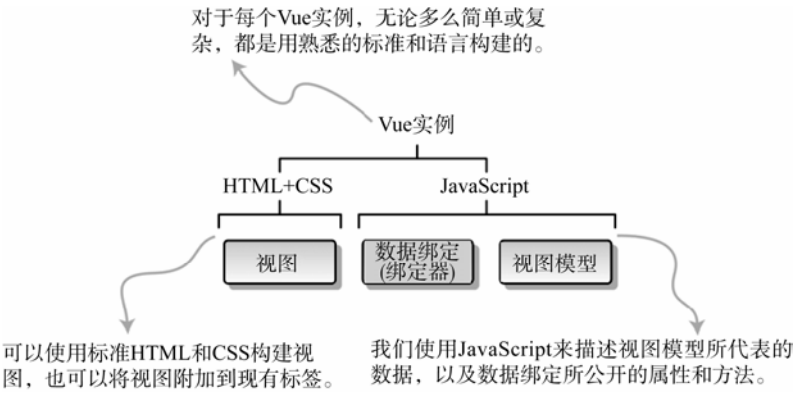

图 1.4 典型的 Vue 实例通过在 HTML 标签和视图模型之间创建数据绑定将它们绑定在一起

由于完全由 Web 技术构建，单个 Vue 实例完全存在于 Web 浏览器中。关键是，这意味着我们不需要依赖基于服务器的页面重载来刷新视图，也不需要执行业务逻辑、视图或视图模型域的任何其他任务。让我们回顾一下表单提交示例。

或许关于客户端 MVC 架构的最显著变化是，浏览器页面在用户的整个会话期间很少需要(如果还有的话)重新加载。由于视图、视图模型和数据绑定都是用 HTML 和 JavaScript 实现的，因此我们的应用程序可以异步将任务委派给模型，让用户可以执行其他任务。当新数据从模型返回时，Vue 建立的绑定将触发视图中需要发生的任何更新。

可以说，Vue 的主要作用是通过创建和维护视图与视图模型中数据之间的绑定来促进用户交互。在这方面，正如你在第一个应用程序中见到的，Vue 为所有反应式应用程序提供了坚实基础。

1.2 使用 Vue.js 的理由

在开始一个新项目时，有很多决定要做。其中最重要的是应该使用的框架或库。如果你是代理商甚至是独立开发人员，那么选择正确的工具非常重要。幸运的是，Vue.js 是多功能的，可以应对许多不同的情况。

以下是你作为独立开发人员或代理商启动新项目时可能遇到的几个最常见的问题，

以及 Vue 如何直接或作为更大的反应式 Web 应用程序的一部分来解决这些问题的说明。

- 我们的团队不善于使用 Web 框架。在项目中使用 Vue 的最大优势之一是不需要什么高深的专业知识。每个 Vue 应用程序都使用 HTML、CSS 和 JavaScript 等熟悉的工具构建，让你可以从一开始就提高工作效率。即使是那些几乎没有任何前端开发经验的团队，也会在 MVVM 模式中找到一个舒适的立足点，因为他们熟悉其他环境中的 MVC 模式。
- 我们已经有一些成果了，想要在原有的基础上继续。别担心，不用废弃呕心沥血设计的 CSS 或酷炫的图片轮播。无论是将 Vue 放入具有多个依赖项的现有项目中，还是开始一个新项目并希望利用你们已熟悉的其他库，Vue 都不会受到影响。你可以继续使用 Bootstrap 或 Bulma 等工具作为 CSS 框架，保留 jQuery 或 Backbone 组件，用你喜欢的库发出 HTTP 请求，处理 Promise 或其他扩展功能。
- 我们需要快速制作原型来评估用户的反应。正如我们在第一个 Vue 应用程序中看到的，要开始使用 Vue 来帮助构建，我们要做的就是在任何独立网页中引用 Vue.js。无需复杂的构建工具！从开始开发到在用户面前显示原型只需要一到两周时间，让你可以尽早收集反馈并频繁迭代。
- 我们的产品只用于移动设备。缩小并压缩的 Vue.js 文件大小约为 24KB，对于前端框架来说非常紧凑。所需文件可通过蜂窝数据连接轻松传递。Vue 2 有个新功能：服务器端渲染(SSR)。这样的策略意味着应用程序的初始负载可以是最小的，能让你仅在需要时引入新的视图和资源。将 SSR 与有效的组件缓存相结合，可进一步降低带宽消耗。
- 我们的产品具有独特的自定义功能。Vue 应用程序是从头开始构建的，是模块化的且具有可扩展性，使用可重用的组件。Vue 还支持通过继承扩展组件，将功能与 Mixin 相结合，并通过插件和自定义指令扩展 Vue 的功能。
- 我们拥有庞大的用户群，担心会有性能问题。Vue 最近基于可靠性、性能和速度进行了重写，现在使用了虚拟 DOM。这意味着 Vue 首先对尚未加载到浏览器的 DOM 执行渲染操作，然后再将这些更改“复制”到我们能看到的视图中。因此，Vue 通常优于其他前端库。由于通用测试通常过于抽象，我总是鼓励客户选择几个典型用例和几个极端用例，开发测试场景并自己测量结果。可以访问 https://vuejs.org/v2/guide/comparison.html，以了解更多有关虚拟 DOM 以及与竞争对手对比的信息。
- 我们有现行的构建、测试和(或)部署过程。在本书的后面章节中，我们将深入探讨这些主题，但最重要的是 Vue 很容易集成到许多流行的构建框架(Webpack、Browserify 等)和测试框架(Karma、Jasmine 等)中。如果已经为现行框架编写了单元测试，在多数情况下是可以直接移植的。如果刚开始但同

时想要使用这些工具，Vue 会提供集成这些工具的项目模板。用最简单的术语来说，将 Vue 添加到现有项目中是很容易的。

- 如果在集成期间或集成之后需要帮助，我们该怎么办？Vue 的两个不可估量的好处是它的社区和支持生态系统。Vue 的在线文档和代码本身都有详细记录，并且核心团队很活跃、反应很快。也许更重要的是，Vue 的开发人员社区也同样强大。Gitter 和 Vue 论坛等平台也有许多乐于助人的人，并且几乎每天都会有越来越多的流行代码被放到平台上，包括插件、集成和库扩展。

经过在我自己的项目上询问了很多这些问题之后，我现在向几乎所有的项目都推荐 Vue。当你通过本书对 Vue 的掌握充满信心时，我希望你能在下一个项目中推行 Vue。

1.3 展望未来

作为开篇，在本章中我们已经涵盖了许多方面。如果你刚开始接触 Web 应用程序开发，这可能是第一次接触 MVVM 架构或反应式编程，但我们已经看到构建反应式应用程序并不像听到这些术语时那么可怕。

也许本章最多的内容并不是关于 Vue 本身，而是反应式应用程序如何越来越容易使用、容易编写。另一个好处是，我们可以编写更少的样板化接口代码。不必编写所有的用户交互脚本，我们就可以专注于如何建模数据和设计接口。把它们连接在一起对 Vue 来说毫不费力。

如果你像我一样，那么肯定已经在考虑使用许多方法来改善我们的应用程序。这是一件好事，你绝对应该实验和运行代码。当我审视应用程序时，会考虑以下几点：

- 如何才能避免需要在这么多地方复制文本字符串？
- 当用户聚焦输入框时，如何才能清空默认值？当用户离开空的输入框时又如何再次恢复默认值？
- 是否有办法避免手写每个输入框？

在本书第 II 部分，我们会找到这些问题的答案以及更多问题和答案。Vue 旨在让我们开发人员一起成长，与我们的代码一起成长，因此我们将始终确保采用不同的策略，比较它们的优缺点，并学习如何确定哪种是针对特定情况的最佳实践。

好吧，让我们看看如何改进我们编写的一些内容！

1.4 本章小结

- 简要介绍模型、视图和控制器的工作原理，以及它们如何与 Vue.js 联系起来。
- Vue.js 如何让你在创建应用程序时节省时间？
- 为什么应该在下一个项目中考虑使用 Vue.js？

第2章

Vue实例

本章涵盖：

- 创建 Vue 实例
- 研究 Vue 生命周期
- 向 Vue 实例增加数据
- 绑定数据到标签
- 格式化输出数据

通过本书，我们将构建一个完整的 Web 应用程序：一个包含商品列表、结账流程和管理界面等功能的网络商店。要完成一个完整的网络商店似乎还有很长的路要走，尤其是在刚开始开发 Web 应用程序时，但是 Vue 能让你基于所学的内容从小处着手构建，在一个平稳的进程中交付复杂的产品。

Vue 实例，就是在应用程序发展的每个阶段保证 Vue 的一致性的关键所在。一个 Vue 应用程序就是一个 Vue 实例，所有 Vue 组件也都是 Vue 实例，甚至可通过创建具有自定义属性的实例来扩展 Vue。