前 言

随着移动互联网技术的迅速发展，基于互联网的应用与日俱增，PC 端和移动端商业应用需求层出不穷，网站重构、用户体验提升、前端交互的需求越来越高，商业应用功能越来越复杂，原有的前端框架已经不能满足新的中大型商业应用的需要。在传统的 MVC 下，前端和后端发生数据交互后会刷新整个页面，从而导致比较差的用户体验。特别是在移动端，当前端对数据进行操作时，刷新页面的代价太昂贵。目前，AngularJS、React 和 Vue.js 三大主流渐进式前端框架能够很好地解决这一问题。相比 AngularJS 和 React，Vue.js 这个渐进式的 MVVM 框架具有更轻量、渲染速度更快、打包体积更小、学习曲线比较平稳、用户体验更佳等特点，深受全球用户欢迎，中国用户特别喜爱。所以笔者以“Vue.js 前端框架技术与实战”为主题创作新编教材，重点阐述 Vue.js 的基础、指令、组件、Vue Router、Vuex 及周边生态技术和应用，以帮助读者掌握 Vue 前端项目的开发流程和开发方法，从而满足当前商业应用的需求。

Vue.js 是一套用于构建用户界面的渐进式框架。Vue 采用自底向上增量开发的设计，这与其他重量级框架有所不同，Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。最初它不过是个人项目，发展至今，已成为全世界三大主流前端框架之一，领先于 AngularJS 和 React，在国内更是首选。它的设计思想、编码技巧也被众多的框架借鉴、模仿。近几年，Vue.js 前端框架越来越受欢迎，越来越多的网站前端开始采用 Vue.js 开发。为了方便 Web 前端开发人员、编程爱好者以及广大读者用户熟悉和使用 Vue.js，笔者编写了本书。

编写思路

以 Vue 工程项目开发的生命周期为导向，从项目脚手架、构建、插件化、组件化，到编辑器工具、浏览器插件、Vue 周边生态等方面构建教材知识体系，从基础应用、项目开发环境配置、开发工具使用、编译打包工具、项目实战等方面构建内容结构。基础部分为了方便读者理解和消化，不使用任何工具，只引入 Vue.js 和浏览器就可以运行 Vue 应用；提高部分及实战部分需要使用脚手架、插件、组件等开发 Vue 应用程序。整本书以“基础 Vue 单页应用→Vue 前端项目→Vue 前后端分离项目”为开发线路，逐层深入、递进式培养读者的工程能力和工程素养。每章配置学习目标、本章小结和适量练习与实训，帮助读者消化和理解所学知识，运用所学知识和技能解决实训中的技术难题，从而提高自己的编程能力和水平。

编写特色

- **理论教学与技能实训一体化设计。**本书填补了 Vue.js 图书市场上一直缺乏“理论与实训一体化设计教材”的空白。在构建章节结构时，设置了本章学习目标、教学内容（含基础语法、语法说明、示范案例、代码解释、注意提醒等）、本章小结、练习与实训（实训要求和实训步骤）。
- **知识传授与能力培养一体化实施。**本书在传授知识的同时，将工程项目中常用环境配置、开发工具的使用及项目工程化工具一并传授，融会贯通，以期培养学生的工程能力和工程素养。
- **知识更新与技术发展同步。**紧跟 Vue.js 技术的发展，及时将 Vue.js 3.0 新特性和新应用写入书中，进一步完善本书的知识体系结构，引入新技术、新特性、新应用，提高项目开发速度、项目执行速度，降低开发成本。

教学资源

为了方便各类高校选用教材和读者选书自学，本书提供了大量的配套资源，包括教学大纲、教学课件、程序源码、素材、习题答案和 700 分钟的微课视频。书中教学案例以统一格式进行命名，如 vue_2_1.html 表示第 2 章第 1 个案例。每章资源以子目录形式存放，如 chapter5 存放第 5 章的各类资源。

资源下载提示

课件等资源：扫描封底的“课件下载”二维码，在公众号“书圈”下载。

素材（源码）等资源：扫描目录上方的二维码下载。

在线作业：扫描封底作业系统二维码，登录网站在线做题及查看答案。

视频等资源：扫描封底刮刮卡中的二维码，再扫描书中相应章节中的二维码，可以在线学习。

全书由储久良独立编写、修订和统稿。本书出版得到清华大学出版社相关人员的大力支持，在此谨表示衷心感谢。作者参阅了 GitHub 和其他网络资源，对这些资源的贡献者深表感谢。由于互联网技术发展迅速，前端技术持续改进与优化，加上作者水平有限，书中的疏漏之处在所难免，恳请各位专家和读者批评指正。

本书是 2019 年江苏省高等教育教改立项研究课题“‘Web 前端开发技术’数字课程与优质教学资源共建共享研究与实践”（项目编号：2019JSJG596）的成果。

作　者

2021 年 8 月

目 录

源码下载

第 1 章 Vue.js 概述 …… 1
1.1 Vue.js 简介 …… 1
1.2 Vue.js 生产环境配置 …… 2
1.2.1 Vue.js引入方法 …… 2
1.2.2 安装Vue Devtools …… 3
1.3 Vue 页面基本结构 …… 5
1.3.1 <template>标记 …… 5
1.3.2 <script>标记 …… 6
1.3.3 <style>标记 …… 8
1.4 Vue.js 开发工具 …… 8
1.4.1 Visual Studio Code …… 9
1.4.2 Sublime Text …… 9
1.4.3 WebStorm …… 10
1.4.4 HBuilderX …… 10
本章小结 …… 11
练习 1 …… 11
实训 1 …… 12
第 2 章 Vue.js 基础 …… 15
2.1 MVC 与 MVVM 模式 …… 15
2.1.1 MVC模式 …… 15
2.1.2 MVVM模式 …… 16
2.1.3 MVVM模式的前端框架发展趋势 …… 17
2.1.4 MVVM模式的应用 …… 17
2.2 数据绑定与插值 …… 20
2.2.1 文本绑定 …… 20
2.2.2 HTML代码绑定 …… 20
2.2.3 属性绑定 …… 21
2.2.4 JavaScript表达式绑定 …… 21
2.3 计算属性与方法 …… 22
2.3.1 计算属性基础应用 …… 23

2.3.2 计算属性缓存与方法的比较 …… 24
2.3.3 计算属性的setter和getter …… 26
2.4 侦听属性 …… 29
2.4.1 侦听属性基本用法 …… 29
2.4.2 侦听属性高级用法 …… 31
2.5 生命周期钩子函数 …… 34
2.5.1 生命周期钩子函数的作用 …… 34
2.5.2 生命周期钩子函数的应用 …… 37
2.6 控制台对象 …… 39
2.6.1 显示信息的命令 …… 40
2.6.2 占位符 …… 40
2.6.3 分组显示 …… 41
2.6.4 查看对象的信息 …… 42
2.6.5 显示某个节点的内容 …… 42
2.6.6 判断变量是否为真 …… 42
2.6.7 追踪函数的调用轨迹 …… 43
2.6.8 计时功能 …… 43
2.6.9 性能分析 …… 44
2.6.10 表格形式输出数组和对象 …… 45
2.7 数据与方法 …… 48
2.7.1 数据对象的定义与使用 …… 49
2.7.2 Vue实例属性与方法 …… 50
2.8 Vue 中的数组变动更新检测 …… 53
2.8.1 变异方法 …… 53
2.8.2 非变异方法 …… 54
2.9 Vue 中的过滤器 …… 57
本章小结 …… 60
练习 2 …… 61
实训 2 …… 62
第 3 章 Vue.js 指令 …… 66
3.1 Vue.js 内置指令 …… 67
3.1.1 条件渲染 …… 67
3.1.2 用key管理可复用的元素 …… 69
3.1.3 根据条件展示元素 …… 71
3.1.4 列表渲染 …… 73
3.1.5 绑定属性 …… 81
3.1.6 事件处理 …… 83
3.1.7 事件修饰符 …… 85
3.1.8 按键修饰符 …… 90
3.1.9 表单输入绑定 …… 92
3.1.10 表单元素值绑定 …… 95
3.1.11 v-model修饰符 …… 97

3.1.12 v-text与v-html指令 …… 98
3.1.13 v-pre、v-once和v-cloak指令 …… 100
3.2 Vue.js 自定义指令 …… 101
3.2.1 自定义指令注册 …… 101
3.2.2 对象字面量 …… 105
3.2.3 动态指令参数 …… 105
3.2.4 自定义指令实际应用 …… 107
本章小结 …… 108
练习 3 …… 109
实训 3 …… 110
第 4 章 Vue.js 基础项目实战 …… 114
4.1 简易图书管理 …… 114
4.1.1 项目需求 …… 114
4.1.2 项目实现 …… 115
4.2 我的待办事项 …… 119
4.2.1 项目需求 …… 119
4.2.2 项目实现 …… 120
本章小结 …… 125
实训 4 …… 125
第 5 章 Vue.js 组件开发 …… 126
5.1 组件基础 …… 126
5.1.1 组件命名 …… 127
5.1.2 组件注册 …… 128
5.2 组件间通信 …… 131
5.2.1 父组件向子组件传值 …… 131
5.2.2 子组件向父组件传值 …… 139
5.2.3 兄弟组件之间的通信 …… 143
5.2.4 父链与子组件索引 …… 145
5.3 单文件组件 …… 146
5.4 插槽 …… 149
5.4.1 匿名插槽 …… 149
5.4.2 具名插槽 …… 151
5.4.3 作用域插槽 …… 153
5.4.4 动态插槽名 …… 155
本章小结 …… 157
练习 5 …… 158
实训 5 …… 158
第 6 章 Vue.js 过渡与动画 …… 163
6.1 单元素/单组件的过渡 …… 163
6.1.1 过渡的类名 …… 165
6.1.2 CSS 过渡 …… 166

6.1.3 CSS动画 ……167
6.1.4 自定义过渡的类名 ……168
6.1.5 同时使用过渡和动画 ……170
6.1.6 显性的过渡持续时间 ……170
6.1.7 JavaScript 钩子 ……170
6.2 初始渲染的过渡 ……171
6.3 多个元素的过渡 ……172
6.4 多个组件的过渡 ……177
6.5 列表过渡 ……178
6.5.1 列表的进入/离开过渡 ……179
6.5.2 列表的排序过渡 ……181
6.5.3 列表的交错过渡 ……183
本章小结 ……186
练习 6 ……186
实训 6 ……187
第 7 章 Vue 项目开发环境与辅助工具部署 …… 192
7.1 部署 Node.js ……192
7.1.1 Node.js简介 ……192
7.1.2 Node.js部署 ……194
7.1.3 Node.js模块系统 ……194
7.1.4 Node.js 创建第1个应用 ……202
7.2 Node 包管理器 npm ……203
7.2.1 npm简介 ……203
7.2.2 npm常用命令 ……204
7.3 Node.js 环境配置 ……206
7.4 webpack 打包工具 ……208
7.4.1 webpack简介 ……208
7.4.2 webpack使用与基本配置 ……209
7.4.3 webpack配置加载器 ……215
7.4.4 webpack配置插件 ……222
7.4.5 webpack配置开发服务器 ……228
7.5 Vue CLI ……235
7.5.1 Vue CLI安装 ……236
7.5.2 Vue CLI创建Vue项目 ……236
7.5.3 Vue CLI可视化创建Vue项目 ……238
本章小结 ……240
练习 7 ……240
实训 7 ……241
第 8 章 前端路由 Vue Router …… 246
8.1 Vue Router 概述 ……246
8.1.1 Vue Router的安装与使用 ……247
8.1.2 Vue Router基础应用 ……247

8.2 Vue Router 高级应用 252
8.2.1 动态路由匹配 252
8.2.2 嵌套路由 254
8.2.3 编程式导航 260
8.2.4 命名路由 262
8.2.5 命名视图 263
8.2.6 重定向和别名 264
8.2.7 路由组件传参 265
8.2.8 HTML5 History模式 267
本章小结 267
练习 8 268
实训 8 269
第 9 章 状态管理模式 Vuex 275
9.1 Vuex 概述 275
9.1.1 Vuex定义 276
9.1.2 简单状态管理——store模式 277
9.2 Vuex 基本应用 280
9.3 Vuex 核心概念 281
9.3.1 一个完整的store结构 281
9.3.2 最简单的store 282
9.3.3 Vuex中的state 283
9.3.4 Vuex中的getters 287
9.3.5 Vuex中的mutations 291
9.3.6 Vuex中的actions 293
9.3.7 Vuex中的modules 301
9.4 Vuex 多模块实战案例 307
本章小结 313
练习 9 313
实训 9 314
第 10 章 Vue UI 组件库 318
10.1 Vue PC 端组件库 318
10.1.1 Element UI 319
10.1.2 iView UI 331
10.1.3 其他PC端UI组件库 336
10.2 Vue 移动端 UI 组件库 336
10.2.1 Mint UI 336
10.2.2 Vant 340
10.2.3 其他移动端组件库 344
本章小结 345
练习 10 345
实训 10 346

第 11 章　Vue 高级项目实战 …… 351
11.1　友联通讯录 …… 351
11.1.1　项目需求 …… 351
11.1.2　实现技术 …… 352
11.1.3　环境配置 …… 352
11.1.4　项目实现 …… 353
11.2　通用登录/注册管理系统 …… 379
11.2.1　项目需求 …… 379
11.2.2　实现技术 …… 380
11.2.3　环境配置 …… 381
11.2.4　项目实现 …… 383
本章小结 …… 415
练习 11 …… 415
实训 11 …… 415
第 12 章　Vue 3.0 基础应用 …… 416
12.1　Vue 3.0 新特性 …… 416
12.1.1　新特性简介 …… 417
12.1.2　下一阶段工作 …… 418
12.1.3　Vue 3.0学习参考 …… 418
12.2　Vue 3.0 初步体验 …… 418
12.2.1　Vue 3.0下载与引用 …… 418
12.2.2　Vue 3.0创建简易应用 …… 419
12.2.3　Vue 3.0发布文档的使用 …… 422
12.3　Vue 3.0 新特性应用 …… 423
12.3.1　使用脚手架创建项目 …… 423
12.3.2　组件选项 …… 425
12.3.3　ref()、reactive() 和toRefs() 函数 …… 428
12.3.4　computed、watch和watchEffect …… 431
12.3.5　ref引用DOM元素和组件实例 …… 434
12.3.6　Vue Router和Vuex …… 436
12.3.7　Vue 3.0生命周期 …… 441
12.3.8　provide() 和inject() 函数 …… 442
12.3.9　组合式API …… 447
12.3.10　模板refs …… 449
12.4　Vue 3.0 购物车实战 …… 453
12.4.1　项目设计要求 …… 453
12.4.2　项目实现 …… 455
本章小结 …… 464
练习 12 …… 464
实训 12 …… 465
参考文献 …… 470

第 1 章

Vue.js 概述

本章学习目标

通过本章的学习，能够了解 Vue.js 的发展简史和生产环境要求，学会配置生产环境，掌握 Vue.js 页面的基本组成，学会使用 Vue.js 开发工具编写简易的 Vue.js 项目。

Web 前端开发工程师应知应会以下内容。

- 掌握 Vue.js 页面的基本组成。
- 学会配置 Vue.js 生产环境。
- 掌握常用的 Vue.js 开发工具。
- 编写最基本的 Vue.js 页面。

1.1 Vue.js 简介

Vue.js 是一套用于构建用户界面的渐进式框架。与其他大型框架不同的是，Vue.js 被设计为可以自底向上逐层应用。Vue.js 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另外，当与现代化的工具链以及各种支持类库结合使用时，Vue.js 也完全能够为复杂的单页应用（Single Page Application，SPA）提供驱动。

2014 年 2 月，尤雨溪[①]（Evan You）将 Vue.js 正式发布并开源，其图标如图 1-1 所示。Vue.js

① 尤雨溪，毕业于上海复旦附中，在美国完成大学学业，本科毕业于科尔盖特大学，后在帕森斯设计学院获得设计与技术（Design & Technology）艺术硕士学位，现任职于 Google Creative Lab。尤雨溪是 Vue.js 框架的作者，HTML5 版 Clear 的打造人。

是构建 Web 界面的 JavaScript 库，是一个通过简洁的应用程序接口（Application Programming Interface，API）提供高效的数据绑定和灵活的组件系统。2016 年 4 月 27 日，发布 Vue 2.0 的 preview 版本，到 Vue 3.0 发布前稳定版本为 Vue 2.6.12。

图 1-1　Vue.js 的图标

2016 年 9 月 3 日，在南京的 JSConf（Conference for the JavaScript Community in China）上，尤雨溪正式宣布加盟阿里巴巴 Weex 团队，将以技术顾问的身份加入 Weex 团队推动 Vue 和 Weex 的 JavaScript Runtime 整合，目标是让大家能用 Vue 的语法跨三端（PC 端、手机端、平板端）。

Vue 官方团队于 2020 年 9 月 18 日晚发布了 Vue 3.0 版本，代号为 One Piece。此次版本提供了改进的性能、更小的捆绑包大小、更好的 TypeScript 集成、用于处理大规模用例的新 API，为框架的未来长期迭代奠定了坚实的基础。

Vue.js 的特点如下。

（1）易用。掌握超文本标记语言（Hyper Text Markup Language，HTML）、层叠样式表（Cascading Style Sheets，CSS）、JavaScript 即可阅读指南开始构建应用。

（2）灵活。简单小巧的核心，渐进式技术栈，足以应付任何规模的应用。

（3）高效。20Kb min+gzip 运行大小，超快虚拟文档对象模型（Document Object Model，DOM），最省心的优化。

1.2　Vue.js 生产环境配置

与其他的 JavaScript 框架类似，Vue.js 的引入方法通常分为两类：①本地化使用，通过<script>标记加载；②网络化使用，通过内容分发网络（Content Delivery Network，CDN）加载。使用 Vue.js 时，推荐在 Google Chrome 浏览器上安装 Vue Devtools，这样可以更加友好地在界面中审查和调试 Vue.js 应用。

1.2.1　Vue.js 引入方法

Vue.js 生产环境常用的配置方法有以下几种。

1. 使用<script>标记直接加载

从 Vue.js 官网（https://cn.vuejs.org/）上直接下载 vue.min.js（生产版本）或 vue.js（开发版本），并在<head>标记中使用<script>标记引入，目前版本为 2.6.12。

加载基本语法如下。

```
<script type="text/javascript" src="js/vue.min.js"></script>
```

2. 使用 CDN 加载

可以从 jsdelivr 或 cdnjs 获取（版本更新可能略滞后），也可以选择其他相对稳定的 CDN。加载语法如下。

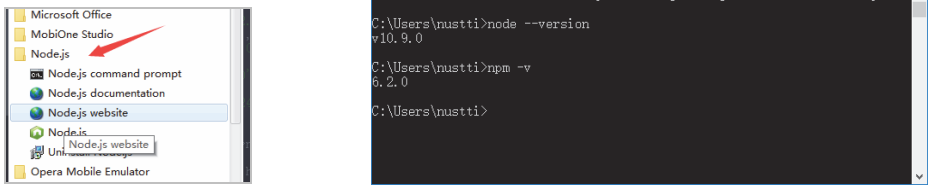
```
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
<script src="https://cdn.jsdelivr.net/npm/vue"></script>
<script src="https://cdn.staticfile.org/vue/2.2.2/vue.min.js"></script>
```

1.2.2　安装 Vue Devtools

在学习和使用 Vue 之前，需要配置好调试的环境。推荐在 Google Chrome 浏览器安装 Vue Devtools 拓展程序。通过 Vue Devtools 帮助开发者查看 Vue 组件和全局状态管理器 Vuex 中记录的数据。具体步骤如下。

1. 下载并安装 Node.js 和 npm

根据操作系统的类型选择下载相应版本的 Node.js。例如，在 Windows（x64）中安装 Node.js，可以从 https://nodejs.org/en/download/官网下载 node-v12.10.0-x64.msi 文件，然后直接安装。安装完成后在“开始”菜单中可以看到 Node.js 程序组件，如图 1-2（a）所示，然后选择第 1 个 Node.js command prompt 程序，若按组合键 Win+R，可以打开命令行窗口，如图 1-2（b）所示。由于新版的 Node.js 已经集成了 Node 包管理器（Node Package Manager，npm），所以 npm 也一并安装好了。npm 的主要功能就是管理 Node 包，包括安装、卸载、更新、查看、搜索、发布等。同样可以通过输入 npm-v 命令测试是否成功安装，如果出现版本提示，则表示安装成功。



（a）程序组件　　（b）命令行窗口

图 1-2　Node.js 安装成功程序组件和命令行窗口状态

2. 从 GitHub 上下载 Vue Devtools

（1）如图 1-3 所示，进入 https://github.com/vuejs/vue-devtools#vue-devtools 页面，单击 Clone or download 按钮，下载后解压 vue-devtools-dev.zip 文件。

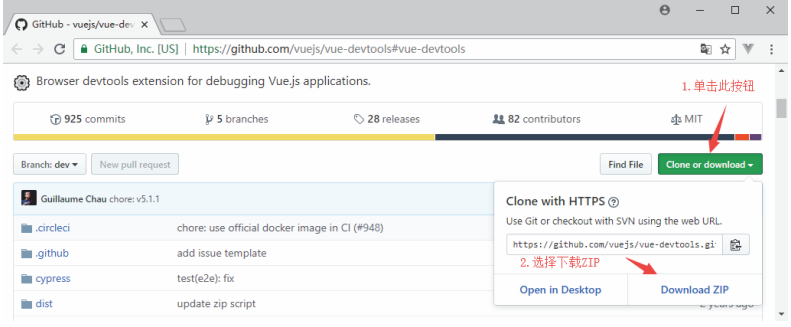


图 1-3　下载 Vue-devtools 界面

（2）然后进入 vue-devtools-dev 子目录，安装构建工具所需要的依赖，命令如下。

```
npm install
```

（3）最后完成工具构建，命令如下。

```
npm run build
```

执行上述命令后，运行结果如图 1-4 所示，说明工具构建完成。

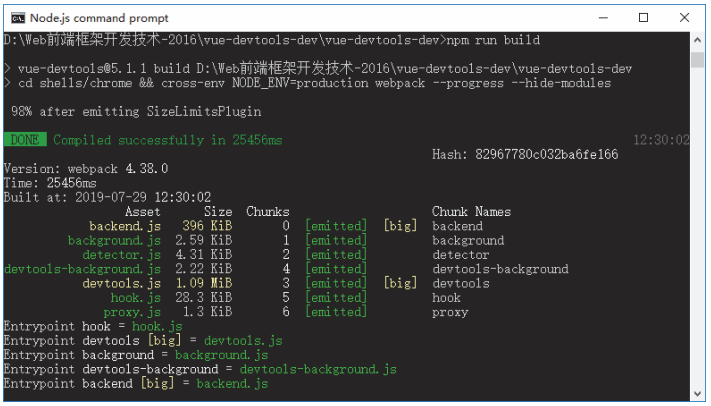


图 1-4　Vue 依赖构建界面

（4）安装 Chrome 浏览器扩展程序。打开 Chrome 浏览器，单击右上角的“⋮”图标，弹出如图 1-5 所示的菜单。从菜单中选择“更多工具”→“扩展程序”，打开“扩展程序”页面，单击“加载已解压的扩展程序”按钮，选择 shells/chrome 文件夹进行安装，显示 Vue.js devtools 加载完成，如图 1-6 所示。

图 1-5　设置 Chrome 浏览器

图 1-6　安装扩展程序加载界面

（5）打开 Vue 应用，并在 Chrome 浏览器中查看页面，按 F12 键进入调试界面。在浏览器地址栏右侧会出现 Vue.js 图标，并且在调试菜单最右侧也会出现 Vue 菜单项目。单击 Vue 菜单，会看到 Vue 模型中存储的数据。

【例 1-1】调试 Vue 应用，代码如下，页面效果如图 1-7 所示。

```
<!-- vue-1-1.html -->
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>调试 Vue 应用</title>
        <script type="text/javascript" src="../vue/js/vue.js">
        </script>
    </head>
    <body>
        <div id="vue11">
            <!-- 文本插值 -->
            {{msg}}
        </div>
```

```
15.         <script type="text/javascript">
16.             //定义 Vue 实例,绑定数据到 DOM 节点上
17.             var myViewModel = new Vue({
18.                 el: '#vue11', //挂载在指定 ID 的元素上
19.                 data: {
20.                     msg: "我喜欢学习 Vue.js! "
21.                 }
22.             })
23.         </script>
24.     </body>
25. </html>
```

上述代码中，第 13 行是文本插值表达式。第 17～22 行创建 Vue 实例，实现数据绑定和渲染 Vue 实例。页面中显示 Vue 的 data 选项中的 msg 值“我喜欢学习 Vue.js！”。在调试界面中选择 Vue 菜单，单击<Root>，可以查看模型中存放的数据，如图 1-7 所示。

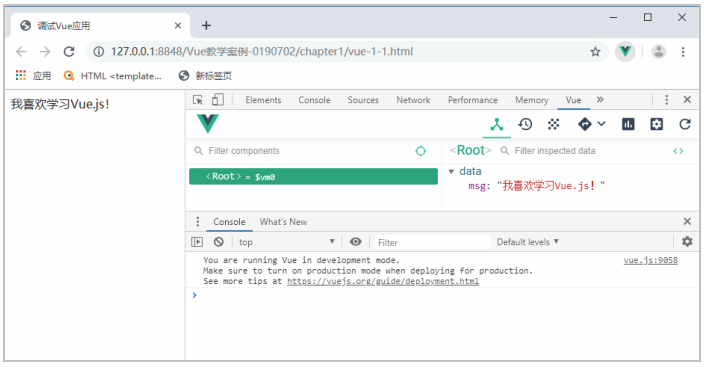


图 1-7　调试 Vue 应用界面

1.3　Vue 页面基本结构

一个 Vue 页面一般由模板、JavaScript 脚本、样式表等部分构成。在 HTML 页面代码中分别由<template>、<script>和<style>等标记组成。

创建一个 Vue 应用程序只需要 3 个步骤：引入 Vue.js 库文件、创建一个 Vue 实例、渲染 Vue 实例。

下面分别介绍 3 个标记的使用方法。

1.3.1　<template>标记

<template>标记用于声明内容模板元素，用来定义 View 视图。该标记允许声明片段 HTML 代码，这些片段 HTML 内容在加载页面时不会呈现，但随后可以在运行时使用 JavaScript 实例化，将这些片段通过脚本复制并插入文档中。

【基本语法】

```
<template  id ="myTem" >
    <!-- 模板内容，即 HTML 片段-->
    <div id="vue11">
        <p> {{content}} </p>
    </div>
</template >
```

【语法说明】

{{ content }}是 Vue 的模板语言语法，表示输出变量 content 的内容。设置 div 的 id 属性值为 vue11，方便后面的 JavaScript 脚本定位它，并将它抽象成一个对象。

【例 1-2】利用 JavaScript 脚本操作模板内容。程序功能是单击“使用模板加载内容”按钮，将<template>标记的内容复制并插入 DOM，代码如下，页面效果如图 1-8 所示。

```
1.  <!-- vue-1-2.html -->
2.  <!DOCTYPE html>
3.  <html>
4.      <head>
5.          <meta charset="utf-8">
6.          <title>template 标记与 script 标记结合使用</title>
7.      </head>
8.      <body>
9.          <!-- 模板内容 -->
10.         <template id="myVueTemp">
11.             <p>模板内容. 再次单击...</p>
12.         </template>
13.         <!-- 普通内容 -->
14.         <div id="div1">
15.             <p>普通内容</p>
16.         </div>
17.         <button onclick="useTemp();">使用模板加载内容</button>
18.         <script type="text/javascript">
19.             function $(id) {
20.                 return document.getElementById(id)
21.             }
22.             function useTemp() {
23.                 var newTemp = $("myVueTemp").content.cloneNode(true);
                    //复制模板节点
24.                 $("div1").appendChild(newTemp);//向 div 中添加模板的内容
25.             }
26.         </script>
27.     </body>
28. </html>
```

图 1-8　复制模板内容并通过 JavaScript 脚本追加到页面上

上述代码中，第 10～12 行定义内容模板，页面加载时，其内容是不显示出来的。第 22～25 行定义 useTemp() 函数，功能是利用<template>标记的 content 属性复制其内容，然后将其添加到 id 为 div1 的 div 中。

1.3.2　<script>标记

<script>标记主要用来创建 Vue 实例。Vue 是一个全局的类，使用时必须先实例化，通过它连接 View（视图）和 Model（模型）。每个 Vue 应用都是通过 Vue 构造函数创建一个 Vue 的根实例开始。

【基本语法】

```
<script type="text/javascript">
    //创建 Vue 实例,绑定数据到 DOM 节点上
    var vm = new Vue({
        //以下为选项设置
        template: '<p>{{msg}}</p>',
        el: '#vue11', //挂载在指定 id 的元素上
        data: {msg: "我是 Vue 新学者! " }
    })
</script>
```

【语法说明】

vm 为 Vue 实例化的对象变量（也称为视图模型）。Vue 是一个封装了响应式开发、模板编译等诸多特性的基础类，可以通过它提供的一些选项（配置项）或属性创建一个实例。它可以包含 data（数据）、template（模板）、el（挂载元素）、methods（方法）、created（生命周期钩子）等选项。全部选项都可以在 API 文档中查看。常用选项对象如下。

- data：声明需要响应式绑定的数据对象。data 中定义若干个键值对，键值对之间用逗号分隔，格式如下。

```
data:{key1: value1,key2:value2,…,keyn:valuen}
```

- template：可以替换挂载元素的 HTML 片段或字符串模板。当模板内容为多个元素时，一定为这些标记再包裹一个根元素（父元素），否则易发生语法错误。
- el：选择页面上已存在的 DOM 元素作为 Vue 实例的挂载目标。
- methods：定义可以通过 vm 对象访问的方法。
- created：发生在 Vue 实例初始化以及 data observer 和 event/watcher 事件被配置之后。

【例 1-3】Vue 常用选项设置。代码如下，页面效果如图 1-9 所示。

```
1.  <!-- vue-1-3.html  -->
2.  <!DOCTYPE html>
3.  <html>
4.      <head>
5.          <meta charset="utf-8">
6.          <title>Vue 选项的使用</title>
7.          <style type="text/css">
8.              .vue11 {text-align:center;border:1px solid red;width:400px;height:460px;}
9.              .vue11 img{height:350px;}
10.         </style>
11.         <script type="text/javascript" src="../vue/js/vue.js"></script>
12.     </head>
13.     <body>
14.         <div id="vue13"></div>
15.         <script type="text/javascript">
16.             var myViewModel = new Vue({
17.                 template: '<div v-bind:class="divstyle"><img v-bind:src= "image">
                    <h3>{{bookname}}荣获 2019 年度清华大学出版社畅销图书</h3></div>',
18.                 data: {
19.                     bookname: 'Web 前端开发技术-HTML5、CSS3、JavaScript',
20.                     image: 'image-1-3.png',
21.                     divstyle: 'vue11'
22.                 },
23.                 el: '#vue13'
24.             })
25.         </script>
```