循序渐进

Vue.js 3.x

前端开发实战

• 张益珲　曹艳琴　编著 •

清華大學出版社
北京

内 容 简 介

本书以一个多年前端“老司机”的视角，循序渐进地介绍流行前端框架Vue.js 3.x全家桶与周边工具在商业项目开发中的应用。全书共15章，第1~6章介绍Vue.js 3的模板、组件、交互处理等基础知识；第7章介绍Vue.js 3框架的响应式编程及组合式API；第8章介绍使用Vue.js 3框架开发前端动画效果；第9章介绍开发大型项目必备的脚手架工具Vue CLI和Vite；第10章介绍基于Vue.js 3的UI组件库Element Plus；第11~13章分别介绍网络请求框架vue-axios、路由管理框架Vue Router、状态管理框架Vuex；第14章和第15章介绍两个项目的开发——学习笔记网站和电商后台管理系统。同时，还精心设计了实践和练习，录制了45集教学视频，提供了完整源代码。

本书通俗易懂，范例丰富，原理与实践并重，适合Vue.js初学者和前端开发人员使用，也可以作为网课、培训机构与大中专院校的教学用书。

图书在版编目（CIP）数据

循序渐进Vue.js 3.x前端开发实战 / 张益珲, 曹艳琴编著. —北京：清华大学出版社，2023.7
ISBN 978-7-302-64121-6

Ⅰ. ①循… Ⅱ. ①张… ②曹… Ⅲ. ①网页制作工具－程序设计 Ⅳ. ①TP393.092.2

中国国家版本馆CIP数据核字（2023）第131434号

责任编辑：王金柱
封面设计：王　翔
责任校对：闫秀华
责任印制：沈　露
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：010-83470000　　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：三河市龙大印装有限公司
经　　销：全国新华书店
开　　本：190mm×260mm　　印　　张：21　　字　　数：567千字
版　　次：2023年8月第1版　　印　　次：2023年8月第1次印刷
定　　价：98.00元

产品编号：102409-01

前　言 PREFACE

本书说明

本书是笔者《循序渐进 Vue.js 3 前端开发实战》一书的修订升级版，上一版在 2022 年 1 月出版后，受到了读者的广泛赞誉，一年多时间连续加印了 8 次，这期间收到了读者的很多来信，一些初学者使用了本书练习做项目，并求职成功，但也建议本书更新为 3.x 版本，并希望对一些内容进行更细致的讲解，还有一些院校老师使用本书作为实践课教材，也发来了改进的建议。本书新版正是基于上述建议改进而来。新版书除保持上一版书的知识结构和核心内容外，主要在以下几个方面进行了改进：

（1）修正了描述不当的部分细节，重构了一些难以理解的内容的讲解方式。

（2）对书中代码新增了索引，方便读者在源文件中找到对应的代码片段。

（3）在源代码中补充了大量注释，方便读者学习理解。

（4）对原书中使用的 Element Plus、Vue Router 等库进行更新，对过时的 API 进行了清理，并替换为新接口。

（5）根据本书读者的反馈，修正了部分错误。

（6）根据一线教师反馈，对语义双关和描述不清的部分知识点进行了补充。

内容特色

本书共 15 章。从前端基础讲起，深入浅出地介绍 Vue.js 框架的功能、用法及部分实现原理。同时，几乎每一章的最后都安排了实践与练习，力求使读者边学边练，快速且扎实地掌握 Vue.js 框架的各种知识，并可以使用它开发出商业级别的应用程序。

第 1 章简单介绍了前端开发必备的基础知识，包括 HTML、CSS 和 JavaScript 这 3 种前端开发必备的技能。这些虽然不是本书的重点，但却是学习 Vue 前必须掌握的基础知识。

第 2 章介绍 Vue 模板的基本用法，包括模板插值、条件与循环渲染的相关语法。这是 Vue.js 框架提供的基础功能，使用这些基础功能能使我们在开发网页应用时事半功倍。

第 3 章介绍了 Vue 组件中属性和方法的相关概念，将使用面向对象的思路来进行前端程序开发，本章的最后介绍了一个功能简单的登录注册页面的开发。

第 4 章介绍前端应用中用户交互的处理方法，一个网页如果不能进行用户交互，那么将如一潭死水，用户交互为应用程序带来灵魂。

第 5 章和第 6 章由浅入深地讲解 Vue.js 中组件的相关应用。组件是 Vue.js 框架的核心，在实际的应用开发中，更是离不开自定义组件技术。

第 7 章介绍 Vue.js 框架的响应性原理，以及 Vue.js 3.0 版本引入的组合式 API 的新特性。本章是对读者开发能力的一种提高，引导读者从实现功能到精致逻辑设计的进步。

第 8 章介绍通过 Vue.js 框架方便地开发前端动画效果。动画技术在前端开发中非常重要，前端是直接和用户面对面的，功能本身只是前端应用的一部分，更重要的是带来良好的用户体验。

第 9 章介绍开发大型项目必备的脚手架 Vue CLI 和 Vite 的基本用法。

第 10 章介绍样式美观且扩展性极强的基于 Vue.js 的 UI 框架 Element Plus；第 11 章介绍网络请求框架 vue-axios；第 12 章介绍一款非常好用的 Vue 应用路由管理框架 Vue Router；第 13 章介绍强大的状态管理框架 Vuex，使用该框架，开发者可以更好地管理大型 Vue 项目各个模块间的交互。这几章内容是开发商业应用程序的必备技能。

第 14 章和第 15 章将通过两个相对完整的应用项目来全面地对本书所涉及的 Vue.js 技能进行综合应用，帮助读者学以致用，更加深入地理解所学习的内容。

配书资源

为了方便读者学习本书，本书还提供了源代码、视频教学、PPT 课件。扫描下述二维码即可下载源代码和 PPT 课件，扫码书中各章节的二维码可以直接观看教学视频。

源码

PPT

如果读者在学习和下载本书的过程中遇到问题，可以发送邮件至 booksaga@126.com，邮件主题写“循序渐进 Vue.js 3.x 前端开发实战”。

最后，对于本书的出版，要感谢支持笔者的家人和朋友，还要感谢清华大学出版社的王金柱编辑的勤劳付出。

希望本书可以带给读者预期的收获。

编者

2023年5月

循序渐进 Vue.js 3.x 前端开发实战

目　录　CONTENTS

第1章 Chapter 1

从前端基础到 Vue.js 3

前端技术是互联网大技术栈中非常重要的一个分支，前端技术本身也是互联网技术发展的见证，其就像一扇窗户，展现了互联网技术的发展与变迁。

前端技术通常是指通过浏览器将信息展现给用户这一过程中涉及的互联网技术，随着目前前端设备的泛化，并非所有的前端产品都是通过浏览器来呈现的，例如微信小程序、支付宝小程序、移动端应用等被统称为前端应用，相应的，前端技术栈也越来越宽广。

讲到前端技术，虽然目前有各种各样的框架与解决方案，最基础的技术依然是前端三剑客：HTML5、CSS3 与 JavaScript。随着 HTML5 与 CSS3 的应用，现代的前端网页的美观程度与交互能力都得到了很大的提升。

本章将作为准备章节，向读者简单介绍前端技术的发展过程，以及前端三剑客的基本概念与应用，并简单介绍响应式开发框架的相关概念。本章还将通过一个简单的静态页面来向读者展示如何使用 HTML、CSS 与 JavaScript 代码来将网页展示到浏览器界面中。

本章学习内容

- 了解前端技术的发展概况。
- 对 HTML 技术有简单的了解。
- 对 CSS 技术有简单的了解。
- 对 JavaScript 技术有简单的了解。
- 认识渐进式界面开发框架 Vue，初步体验 Vue 开发框架。

1.1 前端技术演进

说起前端技术的发展历程，我们还是要从 HTML 说起。1990 年 12 月，计算机学家 Tim Berners-Lee 使用 HTML 语言在 NeXT 计算机上部署了第一套由“主机—网站—浏览器”构成的 Web 系统，我们通常认为这是世界上第一套完整的前后端应用，将其作为 Web 技术开发的开端。

1993 年，第一款正式的浏览器 Mosaic 发布，1994 年年底 W3C 组织成立，标志着互联网进入了标准化发展的阶段，互联网技术将迎来快速发展的春天。

1995 年，网景公司推出 JavaScript 语言，赋予了浏览器更强大的页面渲染与交互能力，使之前的静态网页开始真正地向动态化的方向发展，由此后端程序的复杂度大幅度提升，MVC 开发架构诞生，其中前端负责 MVC 架构中的视图层（V）的开发。

2004 年，Ajax 技术在 Web 开发中得到应用，使得网页可以灵活地使用 HTTP 异步请求来动态地更新页面，复杂的渲染逻辑由之前的后端处理逐渐更替为前端处理，开启了 Web 2.0 时代，由此，类似于 jQuery 等流行的前端 DOM 处理框架相继诞生。其中最流行的 jQuery 框架几乎成为网站开发的标配。

2008 年，HTML5 草案发布，2014 年 10 月，W3C 正式发布 HTML5 推荐标准，众多流行的浏览器也都对其进行了支持，前端网页的交互能力大幅度提高。前端网站开始由 Web Site 向 Web App 进化，2010 年开始相继出现了 AngularJS、Vue.js 等开发框架。这些框架的应用开启了互联网网站开发的 SPA 时代，即单页面应用（Single Page Application，SPA）程序时代，这也是当今互联网 Web 应用开发的主流方向。

总体来说，前端技术的发展经历了静态页面阶段、Ajax 阶段、MVC 阶段，最终发展到 SPA 阶段。

在静态页面阶段，前端代码只是后端代码的一部分，浏览器中展示给用户的页面都是静态的，这些页面的所有前端代码和数据都是后端组装完成后发送给浏览器进行展示的，页面响应速度慢，只能处理简单的用户交互，样式也不够美观。

在 Ajax 阶段，前端与后端实现了部分分离。前端的工作不再只是展示页面，还需要进行数据的管理与用户的交互。当前端发展到 Ajax 阶段时，后端更多的工作是提供数据，前端代码逐渐变得复杂。

随着前端要完成的功能越来越复杂，代码量也越来越大。应运而生的很多框架都为前端的代码工程结构管理提供了帮助，这些框架大多采用 MVC 或 MVVM 模式，将前端逻辑中的数据模型、视图展示和业务逻辑区分开来，为更复杂的前端工程提供了支持。

前端技术发展到 SPA 阶段，意味着网站不再只是用来展示数据，其是一个完整的应用程序，浏览器只需要加载一次网页，用户即可在其中完整地使用多页面交互的复杂应用程序，程序的响应速度快，用户体验也非常好。

1.2 HTML 入门

HTML 是一种描述性的网页编程语言。HTML 的全称为 Hyper Text Markup Language，我们通常也将它称为超文本标记语言。所谓超文本，是指它除了可以用来描述文本信息外，还可以描述超出基础文本范围的图片、音频、视频等信息。

虽然说 HTML 是一种编程语言，但是从编程语言的特性来看，HTML 并不是一种完整的编程语言，它并没有很强的逻辑处理能力，更确切的说法为 HTML 是一种标记语言，它定义了一套标记标签用来描述和控制网站的渲染。

标签是 HTML 语言中非常重要的一部分，标签是指由尖括号包围的关键词，例如 <h1>、<html> 等。在 HTML 文档中，大多标签都是成对出现的，例如 <h1></h1>，在一对标签中，前面的标签是开始标签，后面的标签为结束标签。例如下面就是一个非常简单的 HMTL 文档示例：

```
<html>
<body>
<h1>Hello World</h1>
<p>HelloWorld 网页 </p>
</body>
</html>
```

上面的代码中共有 4 对标签，即 html、body、h1 和 p，这些标签的排布与嵌套定义了完整的 HTML 文档，最终会由浏览器进行解析渲染。

1.2.1　准备开发工具

HTML 文档本身也是一种文本，我们可以使用任何文本编辑器进行 HTML 文档的编写，只需要其文本后缀名使用 .html 即可。使用一款强大的 HTML 编辑器可以极大地提高代码编写效率，例如很多 HTML 编辑器都会提供代码提示、标签高亮、标签自动闭合等功能，这些功能都可以帮助我们在开发中十分快速地编写代码，并且可以减少因为笔误所产生的错误。

Visual Studio Code（VSCode）是一款非常强大的编辑器，其除了提供语法检查、格式整理、代码高亮等基础编程功能外，还支持对代码进行调试和运行，以及进行版本管理。通过安装扩展，VSCode 几乎可以支持目前所有流行的编程语言。本书的示例代码的编写也将采用 VSCode 编辑器完成。用户可以在如下网站下载最新的 VSCode 编辑器：

```
https://code.visualstudio.com
```

目前 VSCode 支持的操作系统有 macOS、Windows 和 Linux，在网站中下载适合自己操作系统的 VSCode 版本进行安装即可，如图 1-1 所示。

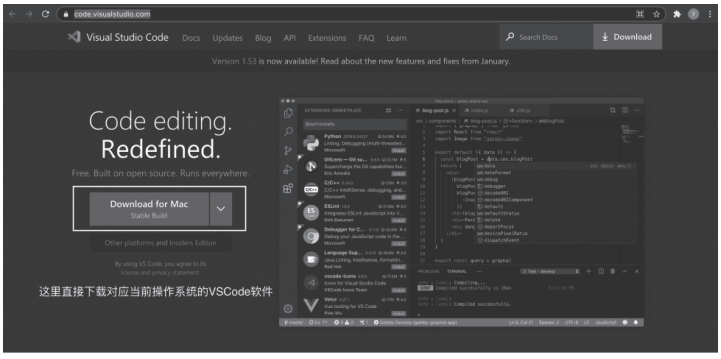

图 1-1　下载 VSCode 编辑器软件

下载并安装 VSCode 软件后，我们可以尝试使用它创建一个简单的 HTML 文档，新建一个名为 test.html 的文件，在其中编写如下测试代码：

【源码见附件代码 / 第 1 章 /1.test.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <h1>HelloWorld</h1>
</body>
</html>
```

示例代码中的标签含义当前无须深究，用户只需要知道 h1 标签用来定义标题，上面的代码可以在网页上显示一行文本为“HelloWorld”的标题即可。相信在输入代码的过程中，用户已经能够体验到使用 VSCode 编程带来的畅快体验，并且在编辑器中关键词的高亮和自动缩进也使代码结构看起来更加直观，如图 1-2 所示。

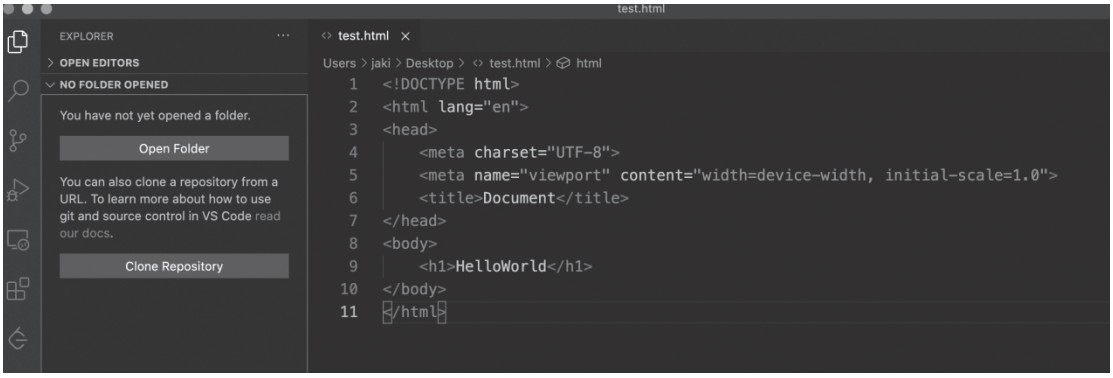

图 1-2 VSCode 的代码高亮与自动缩进功能

在 VSCode 中将代码编写完成后，我们可以直接对齐进行运行，对于 HTML 的源文件的运行，VSCode 会自动将其以浏览器的方式打开，选择 VSCode 工具栏中的 Run → Run Without Debugging 选项，如图 1-3 所示。

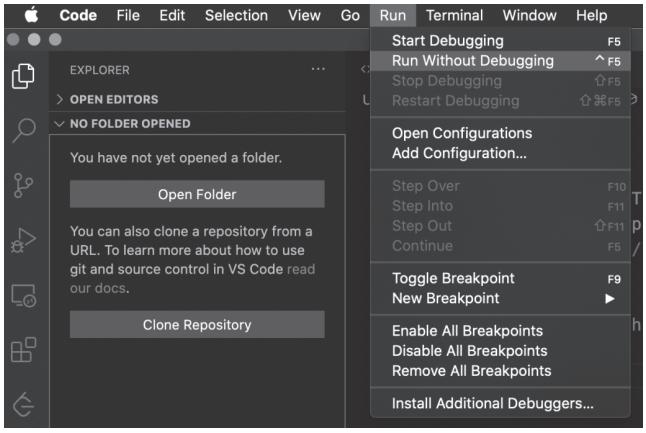

图 1-3 运行 HTML 文件

之后会弹出环境选择菜单，我们可以选择一款浏览器进行预览，如图 1-4 所示。建议安装 Google Chrome 浏览器，该浏览器有很多强大的插件可以帮助用户进行 Web 程序的调试。

预览效果如图 1-5 所示。

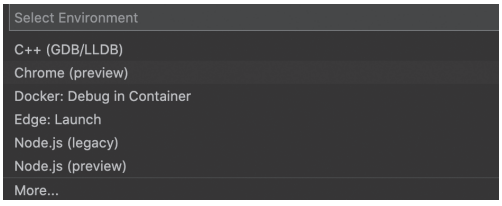

图 1-4　使用浏览器进行预览

图 1-5　使用 HTML 实现的 HelloWorld 程序

1.2.2　HTML 中的基础标签

HTML 中预定义的标签很多，本小节通过几个基础标签的应用实例来向读者介绍标签在 HTML 中的简单用法。

HTML 文档中的标题通常使用 h 标签来定义，根据标题的等级，h 标签分为 h1 ～ h6 共 6 个等级。使用 VSCode 编辑器创建一个名为 base.html 的文件，在其中编写如下代码：

【源码见附件代码 / 第 1 章 /2.base.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> 基础标签应用 </title>
</head>
<body>
    <h1>1 级标题 </h1>
    <h2>2 级标题 </h2>
    <h3>3 级标题 </h3>
    <h4>4 级标题 </h4>
    <h5>5 级标题 </h5>
    <h6>6 级标题 </h6>
</body>
</html>
```

后面的大多示例，HTML 文档的基本格式都是一样的，代码的不同之处主要在 body 标签内，后面的示例只会展示核心的 body 中的代码。

运行上面的 HTML 文件，浏览器渲染效果如图 1-6 所示。可以发现，不同等级的标题文本的字号是不同的。

图 1-6 HTML 中的 h 标签

HTML 文档的正文部分通常使用 p 标签定义，p 标签的意义是段落，正文中的每个段落的文本都可以被包裹在 p 标签内，如下：

【源码见附件代码 / 第 1 章 /2.base.html】

```
<p> 这里是一个段落 </p>
<p> 这里是一个段落 </p>
```

a 标签用来定义超链接，a 标签中的 href 属性可以指向一个新的文档路径，当用户单击超链接的时候，浏览器会跳转到超链接指向的新网页，例如：

【源码见附件代码 / 第 1 章 /2.base.html】

```
<a href="https://www.baidu.com"> 跳转到百度 </a>
```

在实际的应用开发中，我们很少使用 a 标签来处理网页的跳转逻辑，更多时候使用 JavaScript 来操作跳转逻辑。

在 HTML 文档中显示图像也很方便，我们向 base.html 文件所在的目录中添加一个图片素材（demo.png），使用 img 标签来定义图像，如下：

【源码见附件代码 / 第 1 章 /2.base.html】

```
<div><img src="demo.png" alt=" 图片 " width="400px"></div>
```

需要注意，之所以将 img 标签包裹在 div 标签中，是因为 img 标签是一个行内元素，如果我们想让图片单独另起一行展示，则需要使用 div 标签包裹，示例效果如图 1-7 所示。

图 1-7 HTML 文档效果演示

HTML 中的标签可以通过属性对其渲染，或通过交互行为进行控制，例如上面的 a 标签，href 就是一种属性，其用来定义超链接的地址。在 img 标签中，src 属性定义图片素材的地址，width 属性定义图片渲染的宽度。标签中的属性使用如下格式进行设置：

```
tagName = "value"
```

tagName 为属性的名字，不同的标签支持的属性也不同。通过设置属性，我们可以方便地对 HTML 文档中元素的布局与渲染进行控制，例如对于 h1 标签来说，将其 align 属性设置为 center 后，就会在文档中居中展示：

【源码见附件代码 / 第 1 章 /2.base.html】

```
<h1 align = "center">1 级标题 </h1>
```

效果如图 1-8 所示。

图 1-8　标题居中展示

HTML 中还定义了一种非常特殊的标签：注释标签。编程工作除了要进行代码的编写外，优雅地撰写注释也是非常重要的，注释的内容在代码中可见，但是对浏览器来说是透明的，不会对渲染产生任何影响，示例如下：

```
<!-- 这里是注释的内容 -->
```

1.3　CSS 入门

通过 1.2 节的介绍，我们了解到 HTML 文档通过标签来进行框架的搭建和布局，虽然通过标签的一些属性也可以对展示的样式进行控制，但是其能力非常有限，我们在日常生活中看到的网页往往是五彩斑斓、多姿多彩的，这都要归功于 CSS 的强大能力。

CSS 的全称为 Cascading Style Sheets，即层叠样式表。其用处就是定义如何展示 HTML 元素，通过 CSS 控制网页元素的样式极大地提高了编码效率，在实际编程开发中，我们可以先将 HTML 文档的整体框架使用标签定义出来，之后使用 CSS 来对样式细节进行调整。

1.3.1 CSS 选择器入门

CSS 代码的语法规则主要由两部分构成：选择器和声明语句。

声明语句用来定义样式，而选择器则用来指定要使用当前样式的 HTML 元素。在 CSS 中，基本的选择器有通用选择器、标签选择器、类选择器和 id 选择器。

1 通用选择器

使用 * 来定义通用选择器，通用选择器的意义是对所有元素生效。创建一个名为 selector.html 的文件，在其中编写如下示例代码：

【源码见附件代码 / 第 1 章 /3.selector.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS 选择器 </title>
    <style>
        * {
            font-size: 18px;
            font-weight: bold;
        }
    </style>
</head>
<body>
    <h1> 这里是标题 </h1>
    <p> 这里是段落 </p>
    <a> 这里是超链接 </a>
</body>
</html>
```

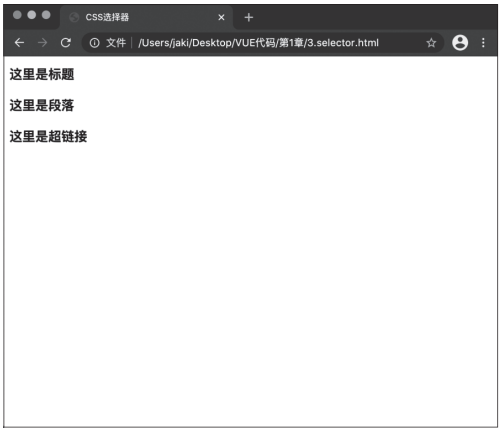

图 1-9 HTML 渲染效果

运行代码，浏览器渲染效果如图 1-9 所示。

如以上代码所示，使用通用选择器将 HTML 文档中所有的元素选中，之后将其内所有的文本字体都设置为粗体 18 号。

2 标签选择器

使用标签选择器，我们可以通过标签名对此标签对应的所有元素的样式进行设置。示例代码如下：

【源码见附件代码 / 第 1 章 /3.selector.html】

```
p {
    color:red;
}
```

上面的代码将所有 p 标签内部的文本颜色设置为红色。

3 类选择器

类选择器需要集合标签的class 属性进行使用，我们可以在标签中添加 class 属性来为其设置一个类名，类选择器会将所有设置对应类名的元素选中，类选择器的使用格式为“.className”。

4 id 选择器

id 选择器和类选择器类似，id 选择器会通过标签的 id 属性进行选择，其使用格式为“#idName”，示例如下：

【源码见附件代码 / 第 1 章 /3.selector.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS 选择器 </title>
    <style>
        * {
            font-size: 18px;
            font-weight: bold;
        }
        p {
            color:red;
        }
        .p2 {
            color: green;
        }
        #p3 {
            color:blue;
        }
    </style>
</head>
<body>
    <h1> 这里是标题 </h1>
    <p> 这里是段落一 </p>
    <p class="p2"> 这里是段落二 </p>
    <p id="p3"> 这里是段落三 </p>
    <a> 这里是超链接 </a>
</body>
</html>
```

运行上面的代码，可以看到“段落一”的文本被渲染成了红色，“段落二”的文本被渲染成了绿色，“段落三”的文本被渲染成了蓝色。

除了上面列举的 4 种 CSS 基本选择器外，CSS 选择器还支持组合和嵌套，例如我们要选中如下代码中的 p：

【源码见附件代码 / 第 1 章 /3.selector.html】

```
<div><p>div 中嵌套的 p</p></div>
```

可以使用后代选择器如下：

【源码见附件代码 / 第 1 章 /3.selector.html】

```
div p {
   color: cyan;
}
```

对于要同时选中多种元素的场景，我们也可以将各种选择器进行组合，每种选择器间使用逗号分隔即可，例如：

【源码见附件代码 / 第 1 章 /3.selector.html】

```
.p2, #p3 {
    font-style: italic;
}
```

此外，CSS 选择器还有属性选择器、伪类选择器等，有兴趣的读者可以在互联网上查找相关资料进行学习。本节只需要掌握基础的选择器的使用方法即可。

1.3.2 CSS 样式入门

掌握了 CSS 选择器的应用，要选中 HTML 文档中的任何元素都非常容易，在实际开发中最常用的选择器是类选择器，我们可以根据组件的不同样式将其定义为不同的类，通过类选择器来对组件进行样式定义。

CSS 提供了非常丰富的样式供开发者进行配置，包括元素背景的样式、文本的样式、边框与边距的样式、渲染的位置等。本节将简单介绍一些常用的样式配置方法。

1 元素背景配置

在 CSS 中，与元素背景配置相关的属性都是以 background 开头的。使用 CSS 对元素的背景样式进行设置，可以实现相当复杂的元素渲染效果。常用的背景属性配置如表 1-1 所示。

表1-1 CSS背景属性配置

属 性 名	意 义	可配置值
background-color	设置元素的背景颜色	这个属性可以接收任意合法的颜色值
background-image	设置元素的背景图片	图片素材的URL
background-repeat	设置背景图片的填充方式	repeat-x：水平方向上重复 repeat-y：垂直方向上重复 no-repeat：图片背景不进行重复平铺
background-position	设置背景图片的定位方式	可以设置为相关定位的枚举值，如top、center等，也可以设置为长度值

2 元素文本配置

元素文本配置包括对齐方式配置、缩进配置、文字间隔配置等，下面的 CSS 代码演示这些文本属性配置的方式。

【源码见附件代码 / 第 1 章 /3.selector.html】

HTML 标签：

```
<div class="text"> 文本属性配置 HelloWorld</div>
```

CSS 设置：

```
.text {
    text-indent: 100px;
    text-align: right;
    word-spacing: 20px;
    letter-spacing: 10px;
    text-transform: uppercase;
    text-decoration: underline;
}
```

效果如图 1-10 所示。

3 边框与边距配置

使用 CSS 可以对元素的边框进行设置，例如设置元素的边框样式、宽度、颜色等。示例代码如下：

【源码见附件代码 / 第 1 章 /3.selector.html】

HTML 元素：

```
<div class="border"> 设置元素的边框 </div>
```

CSS 设置：

```
.border {
    border-style: solid;
    border-width: 4px;
    border-color: red;
}
```

上面示例代码中的 border-style 属性用于设置边框的样式，例如 solid 将其设置为实线，border-width 属性用于设置边框的宽度，border-color 属性用于设置边框的颜色。上面的代码运行后的效果如图 1-11 所示。

图 1-10　使用 CSS 对文本元素进行配置

图 1-11　边框设置效果

使用 border 开头的属性默认对元素的 4 个边框都进行设置，也可以单独对元素某个方向的边框进行配置，使用 border-left、border-right、border-top、border-bottom 开头的属性进行设置即可。

元素定位是 CSS 非常重要的功能之一，我们看到的网页之所以多姿多彩，都要归功于 CSS 可以灵活地对元素进行定位。

在网页布局中，CSS 盒模型是一个非常重要的概念，其通过内外边距来控制元素间的相对位置，盒模型结构如图 1-12 所示。

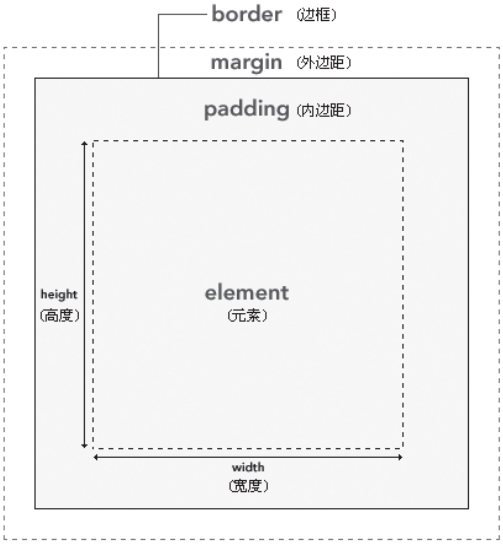

图 1-12 CSS 盒模型示意图

可以通过 CSS 的 height 和 width 属性控制元素的宽度和高度，padding 相关的属性可以设置元素内边距，可以使用 padding-left、padding-right、padding-top 和 padding-bottom 控制 4 个方向上的内边距。margin 相关的属性用来控制元素的外边距，同样的，使用 margin-left、margin-right、margin-top 和 margin-bottom 控制 4 个方向的外边距。通过 margin 和 padding 的设置可以灵活地控制元素间的相对位置。示例如下：

【源码见附件代码 / 第 1 章 /3.selector.html】

HTML 元素：

```
<span class="sp1">sp1</span>
<span class="sp2">sp2</span>
<span class="sp3">sp3</span>
<span class="sp4">sp4</span>
```

CSS 设置：

```
.sp1 {
    background-color: red;
    color: white;
    padding-right: 30px;
}
.sp2 {
    background-color: blue;
    color: white;
    padding-left: 30px;
}
.sp3 {
    background-color: green;
    color: white;
    margin-left: 30px;
}
.sp4 {
```

```
    background-color: indigo;
    color: white;
    margin-right: 30px;
}
```

页面渲染效果如图 1-13 所示。

图 1-13　控制元素内外边距

需要注意，上面的元素之所以在一行展示，是因为 span 标签定义的元素默认为行内元素，不会自动换行布局。

元素的绝对定位与浮动相关内容，这里不重点讲解，在本书后续的测试案例中，我们会逐步使用这些技术为读者演示。

1.4 JavaScript 入门

学习 Vue 开发技术，JavaScript 是基础。本书的后续章节都需要读者能够熟练使用 JavaScript。JavaScript 是一门面向对象的强大的前端脚本语言，如果要深入学习 JavaScript，可能需要一本书的厚度来介绍，这并不是本书的重点，因此，如果读者没有任何 JavaScript 基础，建议学习完本书的准备章节后，先系统地学习一下 JavaScript 语言基础，再继续学习本书后续的 Vue 章节。

本节将只介绍 JavaScript 最核心、最基础的一些概念。

1.4.1 为什么需要 JavaScript

如果将一个网页类比为一个人，HTML 构建了其骨架，CSS 为其着装打扮，而 JavaScript 则为其赋予灵魂。不夸张地说，JavaScript 就是网页应用的灵魂。通过前面的学习，我们知道，HTML 和 CSS 的主要作用是对网页的渲染进行布局和调整。要使得网页拥有强大的功能，并且可以与用户进行复杂的交互，就需要使用 JavaScript 来完成。

首先，JavaScript 能够动态改变 HTML 组件的内容。创建一个名为 js.html 的文件，在其中编写如下示例代码：

【源码见附件代码 / 第 1 章 /4.js.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
```

```
        <title>Document</title>
        <script>
            // 定义名为 count 的变量
    var count = 0
            // 定义一个函数，功能为调用时，修改 HTML 文档中 h1 标签中的内容
    function clickFunc() {
            // 每次调用都将 count 变量的值进行自增
    document.getElementById("h1").innerText = '${++count}'
            }
        </script>
    </head>
    <body>
        <div style="text-align: center;">
            <h1 id="h1" style="font-size: 40px;">数值 :0</h1>
            <button style="font-size: 30px; background-color: burlywood;"
onclick="clickFunc()">单击 </button>
        </div>
    </body>
    </html>
```

上面的代码中使用了几个核心的知识点，在 HTML 标签中可以直接内嵌 CSS 样式表，为其设置 style 属性即可，内嵌的样式表比外联的样式表优先级更高。button 标签是 HTML 中定义按钮的标签，其中 onclick 属性可以设置一段 JavaScript 代码，当用户单击按钮组件时会调用这段代码，如以上代码所示，当用户单击按钮时，我们让其执行 clickFunc 函数。clickFunc 函数定义在 script 标签中，其实现了简单的计数功能，document 对象是当前的文档对象，调用其 getElementById 方法可以通过元素标签的 id 属性的值来获取对应的元素，调用 innerText 可以对元素标签内的文本进行设置。运行代码，可以看到网页上渲染了一个标题和一个按钮，通过单击按钮，标题上显示的数字会进行累加，如图 1-14 所示。

图 1-14 使用 JavaScript 实现计数器

使用 JavaScript 也可以方便地对标签元素的属性进行设置和修改，例如，在页面中再添加一个图片元素，通过单击按钮来设置其显示和隐藏状态：

【源码见附件代码 / 第 1 章 /4.js.html】

HTML 标签：

```
<div id="img" style="visibility: visible;">
    <img src="demo.png" width="200px">
```

```
</div>
```

JavaScript 代码：

```
<script>
    var count = 0
    function clickFunc() {
        document.getElementById("h1").innerText = '${++count}'
       // 根据 count 的值来设置 img 标签的可见状态，若 count 是偶数，则可见，否则隐藏
    document.getElementById("img").style.visibility = count % 2 == 0 ? "visible" :
"hidden"
    }
</script>
```

可以看到，使用 JavaScript 获取标签的属性非常简单，直接使用点语法即可，同理，我们也可以通过这种方式来灵活地控制网页上元素的样式，只需要修改元素的 style 属性即可。运行上面的代码，在网页中单击按钮，可以看到图片元素会交替进行显示与隐藏。

使用 JavaScript 也可以非常容易地对 HTML 文档中的元素进行增删，有时一个非常简单的 HTML 文档能够实现非常复杂的页面，其实都是通过 JavaScript 来动态渲染的。

1.4.2 JavaScript 语法简介

JavaScript 语言的语法非常简单，入门很容易，对于开发者来说上手也非常快。其语法不像某些强类型语言那样严格，语句格式和变量类型都非常灵活。

1 变量的定义

JavaScript 使用 var 或 let 来进行变量的定义，使用 var 定义和 let 定义会使得变量的作用域不同。在定义变量时，无须关心变量的类型，示例如下：

【源码见附件代码 / 第 1 章 /4.js.html】

```
<script>
    var a = 100                 // 定义变量 a，其存储的值为数值 100
    var b = "HelloWorld"        // 定义变量 b，其存储的值为字符串
    var c = {
        name:"abc"
    }                           // 定义键－值对（对象）变量 c
    var d = [1, 2, 3]           // 定义列表变量 d
    var e = false               // 定义布尔变量 e
</script>
```

JavaScript 中的注释规则与传统的 C 语言类似，我们一般使用“//”来定义注释。

2 表达式

几乎在任何编程语言中都存在表达式，表达式由运算符与运算数构成。运算数可以是任意类型的数据，也可以是任意的变量，只要能够支持我们指定的运算即可。JavaScript 支持很多常规的运算符，例如算数运算符 +、−、*、/ 等，比较运算符 <、>、<=、<= 等，示例如下：

【源码见附件代码 / 第 1 章 /4.js.html】

```
var m = 1 + 1                    // 算数运算
var n = 10 > 5                   // 比较运算
var o = false && false           // 逻辑运算
var p = 1 << 1                   // 位运算
```

3 函数的定义与调用

函数是程序的功能单元，JavaScript 中定义函数的方式有两种，一种是使用 function 关键字进行定义，另一种是使用箭头函数的方式进行定义，无论是使用哪种方式定义的函数，其调用方式都是一样的，示例如下：

【源码见附件代码 / 第 1 章 /4.js.html】

```
// 使用 function 关键字定义函数
function func1(param) {
    console.log(" 执行了 func1 函数 " + param);
}
// 使用箭头函数定义函数
var func2 = (param) => {
    console.log(" 执行了 func2 函数 " + param);
}
// 对函数进行调用
func1("hello")
func2("world")
```

运行上面的代码，在 VSCode 开发工具的控制台可以看到输出的信息。console.log 函数用来向控制台输出信息。

4 条件分支语句

条件语句是 JavaScript 进行逻辑控制的重要语句，当程序需要根据条件是否成立来分别执行不同的逻辑时，就需要使用条件语句，JavaScript 中的条件语句使用 if 和 else 关键词来实现，示例如下：

【源码见附件代码 / 第 1 章 /4.js.html】

```
var i = 0                // 定义变量 i，赋值为 0
var j = 1                // 定义变量 j，赋值为 1
if (i > j) {             // if 为条件判断语句，其中表达式为 true 时执行后面大括号内的语句
    console.log("i > j")
} else if (i == j) {     // else if 进行连续条件判断
    console.log("i == j")
} else {                 // 当所有的 if 条件语句都不为 true 时，执行 else 后大括号内的语句
    console.log("i < j")
}
```

JavaScript 中也支持使用 switch 和 case 关键字多分支语句，示例如下：

【源码见附件代码 / 第 1 章 /4.js.html】

```
var u = 0                // 定义变量 u，赋值为 0
switch (u) {             // 对变量 u 进行匹配判定
```

```
    case 0:              // 当 u 为 0 时执行此 case 后的语句
        console.log("0")
        break            // 直接跳出 switch-case 结构
    case 1:              // 当 u 为 1 时执行此 case 后的语句
        console.log("1")
        break            // 直接跳出 switch-case 结构
    default:             // 没有任何 case 被匹配到时，执行 default 后的语句
        console.log("-")
}
```

5 循环语句

循环语句用来重复执行某段代码逻辑，JavaScript 中支持 while 型循环和 for 型循环，示例代码如下：

【源码见附件代码 / 第 1 章 /4.js.html】

```
var v = 10                          // 定义变量 v，赋值为 10
while (v > 0) {                     // while 条件判断为 true 时，进入后面的循环体执行
    v -= 1                          // 每次执行修改循环变量
    console.log(v)
}
// for 循环语句
// 1. 先将循环变量 v 赋值为 0
// 2. 进行循环条件的判断，v 小于 10 时进入循环体执行
// 3. 执行完一次循环体内的代码后，将循环变量进行自增，再次判断是否继续循环
for(v = 0 ; v < 10; v ++) {
    console.log(v)
}
```

除此之外，JavaScript 还有许多非常强大的语法与面向对象能力，我们在后面的章节中使用时会详细介绍。

1.5 渐进式开发框架 Vue

Vue 的定义为渐进式的 JavaScript 框架，所谓渐进式，是指它被设计为可以自底向上逐层进行应用。我们可以只使用 Vue 框架中提供的某层功能，也可以与其他第三方库整合进行使用。当然，Vue 本身也提供了完整的工具链，使用其全套功能进行项目的构建非常简单。

在使用 Vue 之前，需要掌握基础的 HTML、CSS 和 JavaScript 技能，如果读者对本章前面介绍的内容都已经掌握，那么对于后面使用 Vue 的相关例子会非常容易理解。Vue 的渐进式性质使其使用方式变得非常灵活，在使用时可以使用完整的框架，也可以只使用部分功能。

1.5.1 第一个 Vue 应用

在学习和测试 Vue 的功能时，我们可以直接使用 CDN 的方式来进入 Vue 框架，本书将全

部采用 Vue 3.0.x 的版本来编写示例。首先，使用 VSCode 开发工具创建一个名为 Vue1.html 的文件，在其中编写如下模板代码：

【源码见附件代码 / 第 1 章 /5.Vue1.html】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Vue3 Demo</title>
    <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
</head>
<body>
</body>
</html>
```

其中，我们在 head 标签中加入了一个 script 标签，采用 CDN 的方式引入了 Vue 3 的新版本。以之前编写的计数器应用为例，我们尝试使用 Vue 的方式来实现它。首先在 body 标签中添加一个标题和按钮，代码如下：

【源码见附件代码 / 第 1 章 /5.Vue1.html】

```
<div style="text-align: center;" id="Application">
    <h1>{{ count }}</h1>
    <button v-on:click="clickButton"> 单击 </button>
</div>
```

上面使用了一些特殊的语法，例如在 h1 标签内部使用了 Vue 的变量替换功能，{{ count }} 是一种特殊语法，它会将当前 Vue 组件中定义的 count 变量的值替换过来，v-on:click 属性用来进行组件的单击事件绑定，上面的代码中将单击事件绑定到了 clickButton 函数上，这个函数也是定义在 Vue 组件中的，定义 Vue 组件非常简单，我们可以在 body 标签下添加一个 script 标签，在其中编写如下代码：

【源码见附件代码 / 第 1 章 /5.Vue1.html】

```
<script>
    // 定义一个 Vue 组件，名为 App
    const App = {
        // 定义组件中的数据
        data() {
            return {
                // 目前只用到 count 数据
                count:0
            }
        },
        // 定义组件中的函数
        methods: {
            // 实现单击按钮的方法
            clickButton() {
```

```
                this.count = this.count + 1
            }
        }
    }
    // 将 Vue 组件绑定到页面上 id 为 Application 的元素上
    Vue.createApp(App).mount("#Application")
</script>
```

如以上代码所示，我们定义 Vue 组件时实际上定义了一个 JavaScript 对象，其中 data 方法用来返回组件所需要的数据，methods 属性用来定义组件所需要的方法函数。在浏览器中运行上面的代码，当单击页面中的按钮时，计数器会自动增加。可以看到，使用 Vue 实现的计数器应用比使用 JavaScript 直接操作 HTML 元素方便得多，我们不需要获取指定的组件，也不需要修改组件中的文本内容，通过 Vue 这种绑定的编程方式，只需要专注于数据逻辑，当数据本身修改时，绑定这些数据的元素也会同步修改。

1.5.2　范例演练：实现一个简单的用户登录页面

本节尝试使用 Vue 来构建一个简单的登录页面。在练习之前，我们先来分析一下需要完成哪些工作：

（1）登录页面需要有标题，用来提示用户当前的登录状态。

（2）在未登录时，需要有两个输入框和登录按钮供用户输入账号、密码以及进行登录操作。

（3）在登录完成后，输入框需要隐藏，需要提供按钮让用户登出。

仅完成上面列出的 3 个功能点，使用原生的 JavaScript DOM 操作会有些复杂，而借助 Vue 的单双向绑定和条件渲染功能完成这些需求非常容易。

首先创建一个名为 loginDemo.html 的文件，为其添加 HTML 通用的模板代码，并通过 CND 的方式引入 Vue。之后，在其 body 标签中添加如下代码：

【源码见附件代码 / 第 1 章 /6.loginDemo.html】

```
<div id="Application" style="text-align: center;">
<!-- 标题 -->
    <h1>{{title}}</h1>
<!-- 创建账号和密码输入组件 -->
    <div v-if="noLogin">账号：<input v-model="userName" type="text" /></div>
    <div v-if="noLogin">密码：<input v-model="password" type="password" /></
div>
<!-- 登录按钮 -->
    <div v-on:click="click" style="border-radius: 30px;width: 100px; margin:
20px auto; color: white; background-color: blue;">{{buttonTitle}}</div>
</div>
```

上面的代码中，v-if 是 Vue 提供的条件渲染功能，如果其指定的变量为 true，则渲染这个元素，否则不渲染。v-model 用来进行双向绑定，当输入框中的文字变化时，它会将变化同步到绑定的变量上，同样，当我们对变量的值进行更改时，输入框中的文本也会相应发生变化。

实现 JavaScript 代码如下：

【源码见附件代码 / 第 1 章 /6.loginDemo.html】

```
<script>
    const App = {
        data () { // 定义页面所需要的数据
            return {
                title:" 欢迎您：未登录 ",          // 标题
                noLogin:true,                    // 标记是否已经登录
                userName:"",                     // 记录用户名
                password:"",                     // 记录密码
                buttonTitle:" 登录 "             // 登录按钮标题
            }
        },
        methods: {
            click() { // 单击登录按钮执行的方法
                if (this.noLogin) {              // 如果没有登录，则进行登录操作
                    this.login()
                } else {                         // 如果已经登录，则进行登出操作
                    this.logout()
                }
            },
            // 定义登录操作所执行的函数
            login() {
                // 判断账号、密码是否为空
                if (this.userName.length > 0 && this.password.length > 0) {
                    // 模拟登录操作，进行弹窗提示
                    alert('userName:${this.userName} password:${this.password}')
                    // 登录提示后刷新页面
                    this.noLogin = false
                    this.title = ' 欢迎您 :${this.userName}'
                    this.buttonTitle = " 注销 "
                    this.userName = ""
                    this.password = ""
                } else {
                    alert(" 请输入账号密码 ")
                }
            },
            // 定义登出操作所执行的函数
            logout() {
                // 清空登录数据
                this.noLogin = true
                this.title = ' 欢迎您：未登录 '
                this.buttonTitle = " 登录 "
            }
        }
    }
    Vue.createApp(App).mount("#Application") // Vue 组件绑定
</script>
```

运行上面的代码，未登录时效果如图 1-15 所示。当输入账号和密码登录完成后，效果如图 1-16 所示。

图 1-15　简易登录页面（1）

图 1-16　简易登录页面（2）

1.5.3　Vue 3 的新特性

如果读者之前接触过前端开发，那么对于 Vue 框架应该不陌生。Vue 3 的发布无疑是 Vue 框架的一次重大改进。一款优秀的前端开发框架的设计一定是遵循一定的设计原理的，Vue 3 的设计目标如下：

（1）更小的尺寸和更快的速度。

（2）更加现代化的语法特性，加强 TypeScript 的支持。

（3）在 API 设计方面，增强一致性。

（4）提高前端工程的可维护性。

（5）支持更多、更强大的功能，提高开发者的效率。

上面列举了 Vue 3 的核心设计目标，相较于 Vue 2，Vue 3 有哪些重大的更新呢？本节就来简单介绍一下。

首先，在 Vue 2 时代，被压缩的 Vue 核心代码约为 20KB，目前 Vue 3 的压缩版只有 10KB，足足小了一半。在前端开发中，依赖模块越小，意味着越少的流量和越快的速度，在这方面，Vue 3 的确表现优异。

在 Vue 3 中，对虚拟 DOM 的设计也进行了优化，使得引擎可以更加快速地处理局部的页面元素修改，在一定程度上提升了代码的运行效率。同时，Vue 3 也配套进行了更多编译时的优化，例如将插槽编译为函数等。

在代码语法层面，相较于 Vue 2，Vue 3 有比较大的变化。Vue 3 基本弃用了“类”风格的 API，而推广采用“函数”风格的 API，以便更好地对 TypeScript 进行支持。这种编程风格更有利于组件的逻辑复用，例如 Vue3 组件中新引入的 setup（组合式 API）方法，可以让组件的逻辑更加聚合。

Vue 3 中也添加了一些新的组件，比如 Teleport 组件（有助于开发者将逻辑关联的组件封装在一起），这些新增的组件提供了更加强大的功能以便开发者对逻辑进行复用。

总之，在性能方面，Vue 3 无疑完胜 Vue 2，同时打包后的体积也更小。在开发者编程方面，Vue 3 基本是向下兼容的，开发者无须过多的额外学习成本。同时，Vue 3 对功能方面的扩展对于开发者来说也更加友好。

关于 Vue 3 更详细的介绍与新特性的使用方法，后面的章节会逐步向读者介绍。

1.5.4 为什么要使用 Vue 框架

在真正开始学习 Vue 之前，还有一个至关重要的问题，就是为什么要学习它。

首先，进行前端开发，一定要使用一款框架，这就像生产产品的工厂有一套完整的流水线一样。在学习阶段，我们可以直接使用 HTML、CSS 和 JavaScript 开发出一些简单的静态页面，但是要做大型的商业应用，要完成的代码量非常大，要编写的功能函数非常多，而且对于交互复杂的项目来说，如果不使用任何框架来开发的话，那么后期维护和扩展会非常困难。

既然一定要使用框架，那么为什么要选择 Vue 呢？在互联网 Web 时代早期，前后端的界限还比较模糊，有一个名为 jQuery 的 JavaScript 框架非常流行，其内部封装了大量的 JavaScript 函数，可以帮助开发者操作 DOM，并且提供了事件处理、动画和网络相关接口。当时的前端页面更多是用来展示的，因此使用 jQuery 框架足够应付需要进行的逻辑交互操作。后来随着互联网的飞速发展，前端网站的页面越来越复杂，2009 年就诞生了一款名为 AngularJS 的前端框架，此框架的核心是响应式与模块化，它使得前端页面的开发方式发生了变革，从此前端可以自行处理非常复杂的业务逻辑，前后端职责开始逐渐分离，前端从页面展示向单页面应用发展。

AngularJS 虽然强大，但是其缺点十分明显，总结如下：

（1）学习曲线陡峭，入门难度高。

（2）灵活性很差，这意味着如果用户要使用 AngularJS，就必须按照其规定的一套构造方式来开发应用，要完整地使用一整套的功能。

（3）由于框架本身很庞大，因此速度和性能略差。

（4）在代码层面，某些 API 设计复杂，使用麻烦。

只要 AngularJS 有上述问题，就一定会有新的前端框架来解决这些问题，Vue 和 React 这两个框架就此诞生了。

Vue 和 React 在当下前端项目开发中平分秋色，它们都是非常优秀的现代化前端框架。从设计上，它们有很多相似之处，比如相较于功能齐全的 AngularJS，它们都是“骨架”类的框架，即只包含最基础的核心功能，路由、状态管理等功能都是靠分离的插件来支持的。并且逻辑上，Vue 和 React 都是基于虚拟 DOM 树的，改变页面真实的 DOM 要比虚拟 DOM 更改性能的开销大很多，因此 Vue 和 React 的性能都非常好。Vue 和 React 都引导采用组件化的方式进行编程，模块间通过接口进行连接，方便维护与扩展。

当然，Vue 与 React 也有很多不同之处，Vue 的模板编写采用的是类似于 HTML 的方式，写起来与标准的 HTML 非常像，只是多了一些数据绑定或事件交互的方法，入手非常简单。而 React 则采用 JSX 的方式编写模板，虽然这种编写方式提供的功能更加强大一些，但是 JavaScript 混合 XML 的语言使得代码看起来非常复杂，阅读起来也比较困难。Vue 与 React 还有一个很大的区别在于组件状态管理，Vue 的状态管理本身非常简单，局部的状态只要在 data 中进行定义，其默认就被赋予了响应性，在需要修改时直接将对应属性进行更改即可，对于全局的状态也有 Vuex 模块进行支持。在 React 中，状态不能直接修改，需要使用 setState 方法进行更改，从这一点来看，Vue 的状态管理更加简洁一些。

总之，如果用户想尽快掌握前端开发的核心技能并上手开发大型商业项目，Vue 一定不会让用户失望。

1.6 小结与练习

本章是我们进入 Vue 学习的准备章节，在学习 Vue 框架之前，首先需要熟练应用前端 3 剑客（HTML、CSS 和 JavaScript）。通过本章的学习，我们对 Vue 的使用已经有了初步的体验，相信读者已经体会到了 Vue 在开发中为我们带来的便利与高效。

尝试回答下面的问题，如果每道问题在你心中都有了清晰的答案，那么恭喜你过关成功，快开始下一章的学习吧！

练习 1：在网页开发中，HTML、CSS 和 JavaScript 分别起到什么样的作用？

温馨提示：可以从布局、样式和逻辑处理方面思考。HTML定义了界面的文档结构，是页面的骨架。CSS用来对页面元素的布局和样式进行配置，以增强页面的美观性。JavaScript则进行页面更新、用户交互等复杂逻辑的处理，这是让网页从静态到动态的关键。

练习 2：如何动态地改变网页元素的样式或内容，请你尝试在不使用 Vue 的情况下，手动实现本章 1.5.2 节实现的登录页面。

温馨提示：尝试使用JavaScript的DOM操作来重写示例工程。可参考源码：【附件代码/第1章/7.loginDemo.html】。

练习 3：数据绑定在 Vue 中如何使用，什么是单向绑定，什么是双向绑定？

温馨提示：结合本章1.5.2节的示例进行分析。通常只用来展示的元素，其数据进行单向绑定即可；不仅可以展示，还可以接受用户交互的组件，通常需要进行数据的双向绑定。

练习 4：通过对 Vue 示例工程的体验，你认为使用 Vue 开发前端页面的优势有哪些？

温馨提示：可以从数据绑定、方法绑定条件、循环渲染以及Vue框架的渐进式性质本身进行思考。