第 1 章

流行的前端开发框架Vue.js

Vue.js是目前非常流行的JavaScript框架，允许开发人员构建交互式用户界面。它在开发中提供了超强的可扩展性，这也是Vue.js在市场上非常受欢迎的核心原因之一，尤其是对于网站前端开发者来说。本章将重点学习前端开发技术的发展、MV*模式、安装Vue.js、使用Vue.js和Vue.js 3.x的新变化。

1.1 前端开发技术的发展

Vue.js是基于JavaScript的一套MVVC前端框架。在介绍Vue.js之前，先来了解一下Web前端技术的发展过程。

Web刚起步阶段，只有可怜的HTML，浏览器请求某个URL时，Web服务器就把对应的HTML文件返回给浏览器，浏览器做解析后再展示给用户。随着时间的推移，为了能给不同用户展示不同的页面信息，就慢慢发展出来基于服务器的、可动态生成HTML的语言，例如ASP、PHP、JSP等。

但是，当浏览器接收到一个HTML文件后，如果要更新页面的内容，就只能重新向服务器请求获取一个新的HTML文件，再刷新页面。在2G的流量年代，这种体验很容易让人崩溃，而且还浪费流量。

1995年，Web进入JavaScript阶段，在浏览器中引入了JavaScript。JavaScript是一种脚本语言，浏览器中带有JavaScript引擎，用于解析并执行JavaScript代码，然后就可以在客户端操作HTML页面中的DOM，这样就解决了不刷新页面的情况，动态地改变用户HTML页面的内容。再后来发现编写原生的JavaScript代码太烦琐了，还需要记住各种晦涩难懂的API，最重要的是还需要考虑各种浏览器的兼容性，因此出现了jQuery，并很快占领了JavaScript世界，几乎成为前端开发的标配。

直到HTML5的出现，前端能够实现的交互功能越来越多，代码也越来越复杂，从而出现了各种MV*框架，使得网站开发进入SPA（Single Page Application，单页面应用程序）时代。单页面应用程序是指只有一个Web页面的应用。单页应用程序是加载单个HTML页面，并在用

户与程序交互时动态更新该页面的Web应用程序。浏览器一开始会加载必需的HTML、CSS和JavaScript文件，所有的操作都在这个页面上完成，由JavaScript来控制交互和页面的局部刷新。

2015年6月，ECMAScript 6发布，其正式名称为ECMAScript 2015。该版本增加了很多新的语法，从而拓展了JavaScript的开发潜力。Vue.js项目开发中经常会使用ECMAScript 6语法。

1.2　熟悉 MV*模式

MVC是Web开发中应用非常广泛的一种架构模式，之后又演变成MVVM模式。

1.2.1　MVC模式

随着JavaScript的发展，渐渐显现出各种不和谐：组织代码混乱、业务与操作DOM杂糅，所以引入了MVC模式。

在MVC模式中，M指模型（Model），是后端传递的数据；V指视图（View），是用户所看到的页面；C指控制器（Controller），是页面业务逻辑。MVC模式示意图如图1-1所示。

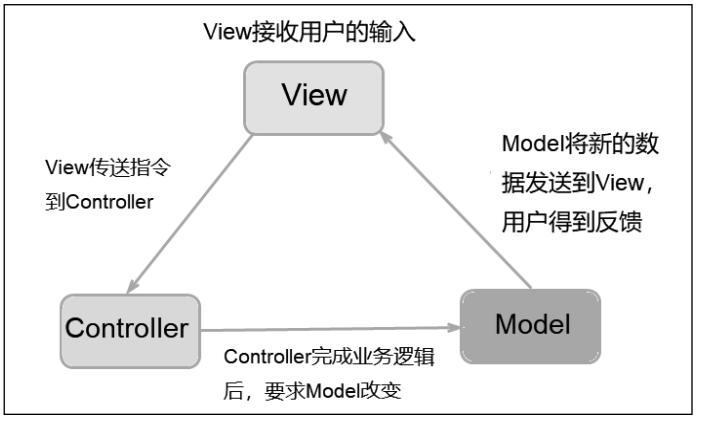

图 1-1　MVC 模式示意图

使用MVC模式的目的是将Model和View的代码分离，实现Web应用系统的职能分工。MVC模式是单向通信的，也就是View和Model需要通过Controller来承上启下。

1.2.2　MVVM模式

随着网站前端开发技术的发展，又出现了MVVM模式。不少前端框架采用MVVM模式，例如，当前比较流行的Angular和Vue.js。

MVVM是Model-View-ViewModel的简写。其中MV和MVC模式中的意思一样，VM指ViewModel，是视图模型。

MVVM模式示意图如图1-2所示。

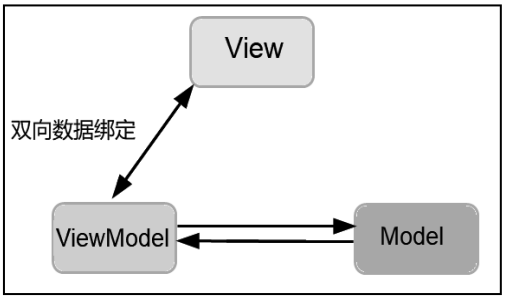

图 1-2　MVVM 模式示意图

ViewModel是MVVM模式的核心，是连接View和Model的桥梁。它有两个方向：

（1）将模型转换成视图，将后端传递的数据转换成用户所看到的页面。
（2）将视图转换成模型，即将所看到的页面转换成后端的数据。

在Vue.js框架中这两个方向都实现了，就是Vue.js中数据的双向绑定。

1.3　Vue.js 概述

Vue.js是一套构建前端的MVVM框架，它集合了众多优秀主流框架设计的思想，轻量、数据驱动、学习成本低，而且可与webpack/gulp构建工具结合以实现Web组件化开发、构建和部署等。

Vue.js本身就拥有一套较为成熟的生态系统：Vue+vue-router+Vuex+Webpack+ Sass/Less，不仅满足小的前端项目开发，也能胜任大型的前端应用开发，包括单页面应用和多页面应用等。Vue.js可实现前端页面和后端业务分离、快速开发、单元测试、构建优化、部署等。

提到前端框架，当下比较流行的有Vue.js、React.js和Angular.js。Vue.js以其容易上手的API、不俗的性能、渐进式的特性和活跃的社区从中脱颖而出。截至目前，Vue.js在GitHub上的受欢迎程度已经超过了其他的前端开发框架，成为最热门的框架。

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。另一方面，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提供驱动。

1.3.1　Vue.js“组件化”思想

Vue.js的目标就是通过尽可能简单的API实现响应、数据绑定和组合的视图组件，核心是一个响应的数据绑定系统。Vue.js被定义成一个用来开发Web界面的前端框架，是个非常轻量级的工具。使用Vue.js可以让Web开发变得简单，同时也颠覆了传统前端开发的模式。

Vue.js是渐进式的JavaScript框架，如果已经有一个现成的服务端应用，可以将Vue.js作为该应用的一部分嵌入其中，带来更加丰富的交互体验。或者，如果希望将更多的业务逻辑放到

前端来实现，那么Vue.js的核心库及其生态系统也可以满足用户的各种需求。

和其他前端框架一样，Vue.js允许将一个网页分割成可复用的组件，每个组件都包含属于自己的HTML、CSS和JavaScript，如图1-3所示，以用来渲染网页中相应的地方。

图 1-3　组件化

这种把网页分割成可复用组件的方式就是框架“组件化”的思想。

Vue.js组件化的理念和React异曲同工——“一切皆组件”。Vue.js可以将任意封装好的代码注册成组件，例如Vue.component('example',Example)，可以在模板中以标签的形式　调用。

example是一个对象，组件的参数配置经常使用到template，它是组件将要渲染的HTML内容。例如，example组件的调用方式如下：

```
<body>
<h1>我是主页</h1>
<!-- 在模板中调用example组件 -->s
<example></example>
<p>欢迎访问我们的网站</p>
</body>
```

如果组件设计合理，在很大程度上可以减少重复开发，而且配合Vue.js的单文件组件（vue-loader），可以将一个组件的CSS、HTML和JavaScript都写在一个文件中，做到模块化的开发。此外，Vue.js也可以与vue-router和vue-resource插件配合起来，以支持路由和异步请求，这样就满足了开发单页面应用程序的基本条件。

在Vue.js中，单文件组件是指一个后缀名为.vue的文件，它可以由各种各样的组件组成，大至一个页面组件，小至一个按钮组件。在后面的章节将详细介绍单文件组件的实现。

Vue.js正式发布于2014年2月，包含70多位开发人员的贡献。从脚手架、构建、组件化、插件化，到编辑器工具、浏览器插件等，基本涵盖了从开发到测试的多个环节。

1.3.2　Vue.js的发展历程

Vue.js的发展过程如下：

2013年12月24日，发布0.7.0。

2014年1月27日，发布0.8.0。
2014年2月25日，发布0.9.0。
2014年3月24日，发布0.10.0。
2015年10月27日，正式发布1.0.0。
2016年4月27日，发布2.0的Preview版本。
2017年第一个发布的Vue.js为v2.1.9，最后一个发布的Vue.js为v2.5.13。
2019年发布Vue.js的2.6.10版本，也是比较稳定的版本。
2020年9月18日，Vue.js 3.0正式发布。
2023年12月28日，Vue.js 3.4正式发布。

1.3.3　Vue.js 3.4的特性

目前，Vue.js的版本已经更新到3.4，这次更新不仅带来了性能上的飞跃，还引入了许多新特性，进一步优化了开发效率。

1. 性能提升

在性能方面，Vue.js 3.4全新重写了模板解析器。与之前基于正则表达式的解析器相比，新的解析器通过单次遍历整个模板字符串来解析模板，显著提高了解析速度，模板解析器速度提高了2倍。无论是小型还是大型的Vue.js模板，新解析器都能保持2倍的性能提升，同时确保了向后兼容性。

2. 重构响应系统

Vue.js 3.4重构了响应系统。在旧版本中，即使计算属性的结果未发生变化，每次依赖项更新，观察者也会被触发。而在Vue.js 3.4中，优化后的系统确保只有在计算结果实际变化时，相关的效果才会被触发，从而减少了组件的不必要渲染，提升了整体的性能。

3. API改进与新特性

Vue.js 3.4中引入了非常多的新特性，主要更新如下：

（1）defineModel API的稳定化及功能：这个API主要用于简化支持v-model的组件实现，并在新版本中增加了对v-model修饰符的支持。

（2）v-bind的同名简写功能：Vue.js 3.4引入了v-bind的同名简写功能，使得开发者在模板中绑定属性时，可以省略重复的变量名。当属性名和绑定的变量名相同时，可以直接使用属性名，从而使模板更加简洁。

（3）watch新增once选项：Vue.js 3.4为watch函数增加了once选项，这使得观察者在第一次检测到变化时就会停止，适用于只需响应一次数据变化的场景。这个新选项提供了一种简洁的方式来防止重复触发。

（4）对MathML的支持：Vue.js 3.4对MathML的支持，意味着开发者现在可以在Vue应用中直接使用MathML来呈现数学公式和符号。MathML是一种标记语言，用于描述数学公式的

结构和内容。这一功能的加入使得Vue.js适用于更广泛的应用场景，特别是在需要展示复杂数学内容的教育和科学出版领域。

1.4 安装 Vue.js 3.x

Vue.js的安装有4种方式：

（1）使用CDN方式。
（2）使用NPM方式。
（3）使用命令行工具（Vue Cli）方式。
（4）使用Vite方式。

1.4.1 使用CDN方式

CDN（Content Delivery Network，内容分发网络）是构建在现有网络基础之上的智能虚拟网络，依靠部署在各地的边缘服务器，通过中心平台的负载均衡、内容分发、调度等功能模块，使用户就近获取所需的内容，降低网络堵塞，提高用户访问响应速度和命中率。CDN的关键技术主要有内容存储和分发技术。

使用CDN方式来安装Vue框架，就是选择一个提供稳定Vue.js链接的CDN服务商。选择好CDN后，在页面中引入Vue的代码如下：

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

1.4.2 NPM

NPM是一个Node.js包管理和分发工具，也是整个Node.js社区最流行、支持第三方模块最多的包管理器。在安装Node.js环境时，安装包中包含NPM，如果安装了Node.js，则不需要再安装NPM。

用Vue构建大型应用时推荐使用NPM安装。NPM能很好地和诸如Webpack或Browserify模块打包器配合使用。使用NPM安装Vue.js 3.x：

```
# 最新稳定版
npm install vue@latest
```

由于在国内访问国外的服务器非常慢，而NPM的官方镜像就是国外的服务器，为了节省安装时间，推荐使用淘宝NPM镜像CNPM，在命令提示符窗口中输入下面的命令并执行：

```
npm install -g cnpm --registry=https://registry.npm.taobao.org
```

之后可以直接使用cnpm命令安装模块，代码如下：

```
cnpm install 模块名称
```

注意 通常在开发Vue.js 3.x的前端项目时，多数情况下会使用Vue CLI先搭建脚手架项目，此时会自动安装Vue.js的各个模块，不需要使用NPM单独安装Vue.js。

1.4.3　命令行工具

Vue提供了一个官方的脚手架（Vue CLI），使用它可以快速搭建一个应用。搭建的应用只需要几分钟的时间就可以运行起来，并带有热重载、保存时lint校验，以及生产环境可用的构建版本。

例如想构建一个大型的应用可能需要将东西分割成各自的组件和文件，如图1-4所示，此时便可以使用Vue CLI快速初始化工程。

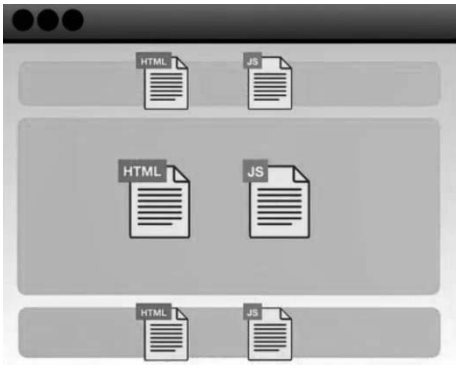

图 1-4　各自的组件和文件

因为初始化的工程可以使用Vue的单文件组件，它包含各自的HTML、JavaScript以及带作用域的CSS或者SCSS，格式如下：

```
<template>
   HTML
</template>
<script>
   JavaScript
</script>
<style scoped>
   CSS或者SCSS
</style>
```

Vue CLI工具假定用户对Node.js和相关构建工具有一定程度的了解。如果是初学者，建议先在熟悉Vue本身之后再使用Vue CLI工具。本书后面的章节将具体介绍脚手架的安装以及如何快速创建一个项目。

1.4.4　使用Vite方式

Vite是Vue的作者尤雨溪开发的Web开发构建工具，它是一个基于浏览器原生ES模块导入的开发服务器。在开发环境下，利用浏览器来解析import，在服务器端按需编译返回，完全跳

过了打包这个概念，服务器随启随用。本书后面的章节将具体介绍Vite的使用方法。

1.5　案例实战——使用 Vue.js 框架

在Vue.js 3.x中，应用程序的实例创建语法规则如下：

```
Vue.createAPP(App)
```

应用程序的实例充当了MVVM模式中的ViewModel。createAPP()是一个全局API，它接受一个根组件选项对象作为参数，该对象可以包含数据、方法、组件生命周期钩子等，然后返回应用程序实例本身。Vue.js 3.x引入createAPP()是为了解决Vue 2.x全局配置代理的一些问题。

创建了应用程序的实例后，可以调用实例的mount()方法制定一个DOM元素，在该DOM元素上装载应用程序的根组件，这样这个DOM元素中的所有数据变化都会被Vue.js框架所监控，从而实现数据的双向绑定。

```
Vue.createAPP(App).mount('#app')
```

下面通过设计一个“产品介绍”的简单页面，了解Vue.js 框架的使用方法。

【例1.1】　编写“产品介绍”页面（源代码\ch01\1.1.html）。

这里使用v-bind指令绑定IMG的src属性，使用{{}}语法（插值语法）显示标题<h2>的内容。

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="UTF-8">
   <title>产品介绍</title>
</head>
<body>
   <div id="app">
      <div><img v-bind:src="url" width="450"></div>
      <h2>{{ explain }}</h2>
   </div>
   <!--引入Vue文件-->
   <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
   <script>
   //创建一个应用程序实例
   const vm= Vue.createApp({
   //该函数返回数据对象
   data(){
      return{
         url:'1.jpg',
         explain:'最新的洗衣机产品。',
      }
   }
   //在指定的DOM元素上装载应用程序实例的根组件
   }).mount('#app');
   </script>
```

```
    </body>
    </html>
```

在Chrome浏览器中运行程序，效果如图1-5所示。以上代码成功创建了第一个Vue.js应用，看起来这跟渲染一个字符串模板非常类似，但是Vue.js在背后做了大量工作。可以通过浏览器的JavaScript控制台来验证。

例如，在浏览器上按F12键，打开控制台并切换到Console选项，修改vm.explain="最新的空调产品。"，修改vm.url="2.jpg"，按回车键后，可以发现页面的内容发生了改变，效果如图1-6所示。

图 1-5　“产品介绍”页面效果

图 1-6　控制台上修改后的效果

出现上面这样的效果，是因为Vue是响应式的。也就是说，当数据变更时，Vue会自动更新所有网页中用到它的地方。除程序中使用的字符串类型外，Vue对其他类型的数据也是响应的。

说明 在之后的章节中，示例不再提供完整的代码，而是根据上下文将HTML部分与JavaScript部分单独展示，省略了<head>、<body>等标签以及Vue.js的加载等，读者可根据上面示例的代码结构来组织代码。

1.6　Vue.js 3.x 的新变化

Vue.js 3.x并没有延用Vue.js 2.x版本的代码，而是从头重写了整个框架，代码采用TypeScript进行编写，新版本的API全部采用普通函数，在编写代码时可以享受完整的性能推断。

与Vue.js 2.x版本相比，Vue.js 3.x具有以下新变化。

1. 重构响应式系统

Vue.js 2.x利用Object.defineProperty()方法侦查对象的属性变化，该方法有一定的缺点：

（1）性能较差。
（2）在对象上新增的属性是无法被侦测到的。
（3）改变数组的length属性是无法被侦测到的。

Vue.js 3.x重构了响应式系统，使用Proxy替换Object.defineProperty。Proxy被称为代理，它的优势如下：

（1）性能更优异。
（2）可直接监听数组类型的数据变化。
（3）监听的目标为对象本身，不需要像Object.defineProperty一样遍历每个属性，有一定的性能提升。
（4）Proxy可拦截apply、ownKeys、has等13种方法，而Object.defineProperty不行。

2. 更好的性能

Vue.js 3.x重写了虚拟DOM的实现，并优化了编译模板，提升了组件的初始化速度，性能提升了1.3~2倍，服务器端的渲染速度提升了2~3倍。

3. tree-shaking支持

Vue.js 3.x只打包真正需要的模块，删除无用的模块，从而减小了产品发布版本的大小。而在Vue.js 2.x中，很多用不到的模块也会被打包进来。

4. 组合API

Vue.js 3.x引入了基于函数的组合API。在引入新的API之前，Vue.js还有其他替代方案，它们提供了诸如Mixin、HOC（高阶组件）、作用域插槽之类的组件之间的可复用性，但是所有方法都有其自身的缺点，因此它们未被广泛使用。

（1）一旦应用程序包含一定数量的Mixins，就很难维护。开发人员需要访问每个Mixin，以查看数据来自哪个Mixin。
（2）HOC模式不适用于.vue单文件组件，因此在Vue开发人员中不被广泛推荐或 使用。
（3）作用域插槽的内容会被封装到组件中，但是开发人员最终拥有许多不可复用的内容，并在组件模板中放置了越来越多的插槽，导致数据来源不明确。

组合API的优势如下：

（1）由于API是基于函数的，因此可以有效地组织和编写可重用的代码。
（2）将共享逻辑分离为功能来提高代码的可读性。
（3）可以实现代码分离。
（4）在Vue.js应用程序中可以更好地使用TypeScript。

5. Teleport

Teleport（传送）是一种能够将模板移动到DOM中的Vue应用程序之外的其他位置的技术。像modals和toast等元素，如果嵌套在Vue的某个组件内部，那么处理嵌套组件的定位、z-index样式就会变得很困难。很多情况下，需要将它与Vue.js应用的DOM完全剥离，管理起来会容易很多，此时就需要用到Teleport。

6. Fragment（碎片化节点）

在Vue 2.x中，每个组件必须有一个唯一的根节点，所以，写每个组件模板时都要套一个父元素。

在Vue 3.x中，为了更方便地书写组件模板，新增了标签元素<Fragment></Fragment>，从而不再限于模板中的单个根节点，组件可以拥有多个节点。这样做的好处在于，减少了标签层级，减小了内存占用。

7. 更好的TypeScript支持

Vue.js 3.x是用TypeScript编写的库，可以享受自动的类型定义提示。JavaScript和TypeScript中的API相同，从而无须担心兼容性问题。结合使用支持Vue.js 3.x的TypeScript插件，开发更加高效，还可以拥有类型检查、自动补全等功能。