第 1 章　Vue. js

1.1　前端技术概述

Web 前端主要技术包括 HTML、CSS 和 JavaScript。HTML 是超文本标记语言，表示前端的内容，常用的版本为 HTML5；CSS 是层叠样式表，表示前端的样式，常用的版本为 CSS3；JavaScript 是脚本语言，常用的版本为 ES6。

JavaScript 是前端技术的核心，JavaScript 库可以使 JavaScript 的开发更加方便、快速、功能强大，一般的工作场景都会使用 JavaScript 库。常用的 JavaScript 库包括传统的 jQuery 和新兴的 Vue. js、react、Anjular. js，后 3 个库都是基于 MVVM 思想的 JavaScript 库。

Typescript 作为 JavaScript 的超集，得到了越来越广泛的使用。在 Vue 3. 0 中，越来越多地使用 Typescript 作为脚本语言。

前端项目在发布时，会使用 webpack 等工具将开发环境的项目发布到生产环境。Node. js 是使用 JavaScript 作为脚本语言的后台开发技术，很多前端的技术岗位也有这种技术需求。微信小程序等小程序的本质也是基于前端技术的技术方法，也属于前端工程师的工作范围。

前端网站的总体实现方案包括同时适配电脑版和手机版的响应式网站，以及分开单独设计的电脑版网站和手机版网站。前者国外常用，后者是国内网站的主要解决方案。一般来说，目前电脑版网站需要设置固定宽度，手机版网站宽度都为视口宽度。

1.2　MVC

MVC(Model-View-Controller)是流行的系统开发架构，它将系统的输入、处理和输出分开，它的 3 个要素分别是表示用户界面的视图(View)、表示业务逻辑的控制器(Controller)、保存数据的模型(Model)。

视图(View)显示用户界面，并可以将客户端的指令发送给控制器；控制器(Controller)根据视图的指令进行逻辑计算，并进而处理模型中的数据；模型(Model)处理应用程序的数据逻辑部分。

在技术实现的角度上，Controller 负责将 Model 的数据在 View 中显示出来，是服务器端的主要逻辑；Model 主要处理对数据库的存取；视图构建用户界面(UI)，用户界面显示来

自 Model 的数据，提供改变 Model 的接口，View 中的数据变化和 Model 相关，和 Controller 相关。

MVC 的软件结构下，前端的视图和后台的业务逻辑相分离，数据库处理和业务逻辑相分离，使系统的结构更加清晰，便于系统的维护和开发。MVC 模式便于团队分工。

MVP 模式是指 Model-View-Presenter。它也是 MVC 模式的一种，Presenter 相当于 Controller，只是对数据的通信做了具体的要求。在 MVP 模式中，View 不能直接访问 Model，必须通过 Presenter 才能访问 Model，View 和 Model 不直接发生关系。View 和 Presenter、Model 和 Presenter 之间的通信都是双向的。View 只负责显示用户界面，主要的逻辑都在 Presenter 中。

MVVM(Model-View-View-Model)模式也是 MVC 模式的一种，它和 MVP 模式很像，使用 ViewModel 代替了 Presenter。在 MVVM 模式中，采用双向数据绑定。MVVM 的数据流向如图 1-1 所示。

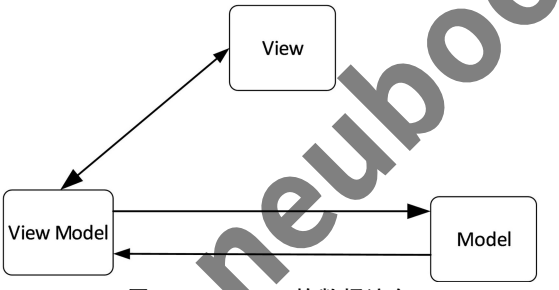

图 1-1 MVVM 的数据流向

MVVM 最早是由微软提出的，ViewModel 使 Model 和 View 的数据自动同步，View 层数据的改变会通过 ViewModel 同步给 Model，Model 数据的改变会通过 ViewModel 同步给 View。MVVM 模式使数据处理更加简单和方便，用户只需要关注数据本身，不需要关注 View 和 Model 中数据的同步。

1.3 Vue.js 概述

Vue(发音同 View)是一套用于构建用户界面的渐进式(Progressive)JavaScript 框架，Vue 的核心库只关注视图层，提供了基于 MVVM 思想的数据双向绑定功能，简单易学，并可以逐步扩展，完成复杂的项目设计。

Vue 的核心代码简单小巧，最小的压缩版本只有 20kb，具有快速的虚拟 DOM 和性能优化功能，可以胜任任何规模的应用。Vue 尤其适合单页应用(SPA)的开发。

Vue 的作者是尤雨溪，出生在中国无锡。Vue 是他在美国谷歌工作的时候，从 Angular 框架中提取的主要内容，是数据绑定的核心部分。对他来说，Angular 对于有些场景过重了。2014 年，尤雨溪在 GitHub 上发布了 Vue.js，并受到了广泛的欢迎，尤雨溪也随即成为专职的自由开发者，专门维护 Vue.js。目前 Vue.js 的最新版本是 3.0 版本。

尤雨溪认为 Vue.js 在所有的 JavaScript 框架中和 React 最像。他认为 Vue 的独特之处在于它是一个渐进的框架。Vue 的核心组成只是数据绑定和组件，和 React 差不多。Vue 只是解决了一小部分很重要的痛点，其他的更重要的功能如路由、状态管理、构建工具链和 CLI 等没有包括在 Vue 核心中。这些功能可以和 Vue 的核心搭配使用，但是不需要使用的时候可以不用。Vue 的核心是数据驱动和组件系统。

大型项目可能使用更加全面的 Vue 功能，常用的包括 Vue 官方支持的 Vue CLI、Vue Router、Vuex，另外一般还包括替代 Vue-resource 的 Axios 和用户界面(UI)的解决方案。这些基于 Vue 的扩展功能的集合、完整的系统前端开发的解决方案，一般称为“Vue 全家桶”。

微信的小程序的语法和 Vue.js 非常相像。

1.4　Vue.js 的安装

Vue 的学习需要有一定的 HTML、CSS、JavaScript 基础，它的实现可以通过基本的 HTML 文件，也可以通过 Vue CLI(脚手架)。Vue 官方不推荐新手直接学习 Vue CLI，推荐先通过基本的 HTML 的形式进行学习，之后再进阶到 Vue CLI。

初学阶段建议使用单独的 JavaScript 文件引入 Vue.js。开发阶段使用的 Vue.js 版本不建议使用压缩的 min 版本，可下载非压缩的版本或直接使用 CDN，参考示例如下：

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

1.5　开发工具

前端工具软件集成开发工具是用来编写 HTML、CSS 和 JavaScript 的软件，目前比较常用的有 VS Code、WebStorm、sublime text、HBuilder、notepad＋＋、Aptana Studio、Dreamweaver等。开发工具可以提供较好的 HTML5、CSS3、JavaScript 等的支持，对代码格式、代码提示、网页框架、智能代码完成、文档结构树、浏览器兼容性提示、网站上传都有较好的支持。总的说来，软件只是工具，它们的本质都是帮助用户更快、更准确、更方便地生成 HTML、CSS 和 JavaScript，所有工具需要完成的结果都是相同的。前端开发的本质是 HTML、CSS 和 JavaScript，集成开发工具是可以更好地善其事的工具。Visual Studio Code 是目前广泛采用的开发工具，是学习前端技术的一种较好的选择，也可选用其他集成开发工具。

Visual Studio Code 简称 VS Code，是 Microsoft 发布的一个源代码编辑器。它可以运行于多种操作系统，如 Windows、Linux、Mac OS X，该编辑器支持多种语言和文件格式的编写，支持通过插件的方式进行功能扩展，是一款优秀的免费、开源的轻量级集成开发工具。

Visual Studio Code 的插件可以大大提高开发效率，多数插件都是自动起作用，部分插

件如 beautify 需要在命令面板(快捷键 F1)中键入命令进行执行。随着学习的深入,可以安装更多的插件提高开发效率。

Visual Studio Code 常用的前端和 Vue 开发的插件如下:

- (Simplified) Language Pack for Visual Studio Code:汉化。
- open in browser(或 view in browser):在浏览器里查看网页。
- HTML Snippets:HTML 代码提示。
- HTMLHint:HTML 代码检测。
- HTML CSS Support:HTML class 名称智能提示。
- Auto Close Tag:自动添加结束标签。
- Auto Rename Tag:重命名 html 标签时,自动修改结束标签。
- beautify:格式化代码。
- live-server:实时加载的小型服务器。
- Path Intellisense:智能提示文件路径。
- JavaScript Snippet Pack:JavaScript 语法提示。
- Bracket Pair Colorizer:配对的括号有不同的颜色。
- Path Autocomplete:路径自动完成。
- Vetur:Vue 语法支持,包括语法高亮、语法代码提示、语法检测等。
- Vue 2 Snippets:自动生成 Vue 2 语法片段。
- Vue 3 Snippets:自动生成 Vue 3 语法片段。
- vue-beautify:Vue 代码格式优化。

1.6 浏览器及插件

为保证网页的浏览器兼容性,建议同时安装 Firefox 和 Chrome 浏览器。所有主流浏览器都提供了开发者工具,通过快捷键 F12 或 Fn+F12 可以在浏览器中打开开发者工具,通过开发者工具可以快速查看网页、调试错误、查看网络加载时间等。

开发者工具是每个开发者必须掌握的工具。开发者工具不但可以调试错误,也是学习网站前端设计的最佳工具,通过开发者工具,可以立刻查看到网站中任何部分对应的 HTML 和其对应的 CSS,是最直观的学习工具。具备了一定的基础后,无论是学习还是调试错误,浏览器开发者工具都是前端的最佳选择。浏览器开发者工具如图 1-2 所示。

Chrome 和 Firefox 这两种浏览器都可以通过安装插件扩展浏览器功能,常用的浏览器插件如下:

- Vue.js Devtools:Vue 开发调试工具,可以 Vue 查看变量、路由、组件的内容。

在浏览器开发者工具中查看该插件,显示效果如图 1-3 所示。

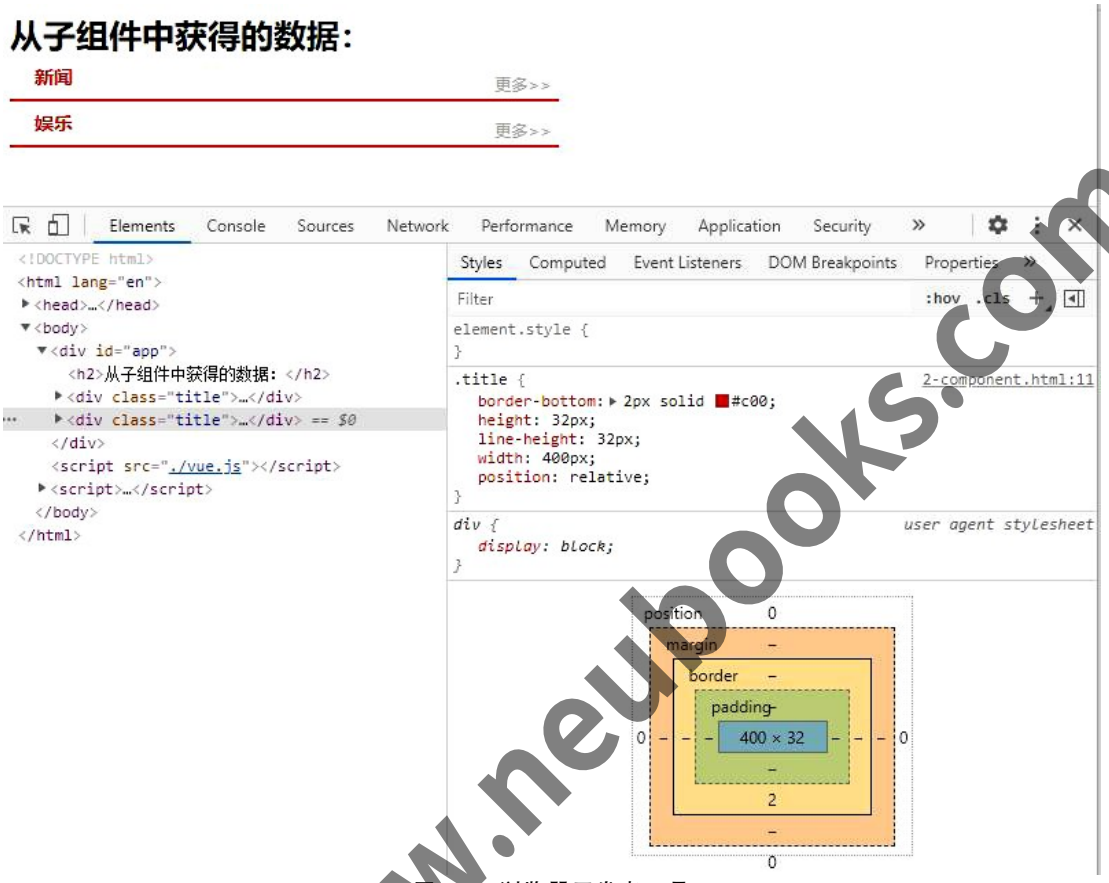

图 1-2　浏览器开发者工具

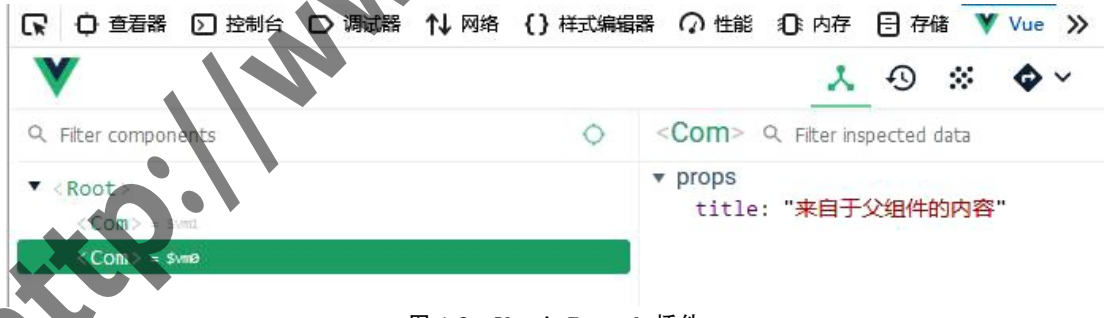

图 1-3　Vue. js Devtools 插件

- Postman：模拟网络请求，可下载独立的软件。
- Web Developer：综合开发插件，主要包括 Disable、Cookies、CSS、Forms、Images、Information、Miscellaneous、Outline、Resize、Tools、View Source 和 Options 等功能。
- FeHelper：综合开发插件，包括多项功能，如图片 Base64 编解码转换、网页取色、代码美化与压缩等。
- JavaScript Errors Notifier：提示 JavaScript 错误。
- JSON Formatter：格式化 JSON。

- Window Resizer:浏览器窗口大小改变为指定宽度和高度。
- FireShot:网页截图,支持全网页截图。
- Image Downloader:浏览和下载网页中的图片。

上述插件大都同时支持 Chrome 和 Firefox 浏览器。Chrome 浏览器由于网络问题,直接在浏览器内安装可能不能成功,可以下载 crx 安装包离线安装,也可以在需要插件功能的时候,使用 Firefox 浏览器。浏览器插件可以在学习进入一定阶段后再进行安装使用。

习 题

1. 安装前端开发工具,安装前端开发所需要的插件。
2. 安装 Chrome 和 Firefox 浏览器,查看其开发者工具。
3. 下载 Vue.js 文件,为正式开始 Vue.js 开发做好准备。
4. 说明你对 MVVM 模式的理解。
5. 说明常用的 JavaScript 库。
6. 在 Chrome 或 Firefox 浏览器中安装 Vue Devtools 插件。
7. 根据上下文填空和回答问题。

(1)在 Vue.js 的 MVVM 模式中,M 表示_____,V 表示_____,VM 表示_____。

(2)MVC 的 C 表示_____。

(3)说明 MVVM 和 MVC 的区别。

第 2 章　Vue.js 基础

2.1　基本结构

Vue.js 可以以一个外部的 JavaScript 文件的形式引入 HTML 文件中，作为一个普通的外部 JavaScript 文件形式使用；在工程化的开发场景下，会使用脚手架(CLI)的方式，快速生成组件化的项目模板。学习的时候，推荐先通过外部 JavaScript 的方式了解 Vue 的特性，然后进一步学习脚手架的开发方式。

在编写第一个 Vue.js 之前，需要下载 Vue.js 文件，目前 Vue.js 官网的 Vue.js 版本为 2.x。在 HTML 网页中引入 Vue.js 文件，就可以按照 Vue.js 的方式进行 JavaScript 文件的编写，Vue 可以通过插值的方式在 HTML 中加入 JavaScript 变量、方法、表达式等；CSS 对 HTML 元素进行样式的修饰，HTML、CSS、JavaScript 互相配合完成网页的设计。Vue.js 仍然是 JavaScript，只是一种适应于某些开发场景的更好的书写 JavaScript 的方式，传统的 JavaScript 仍然可用，别的 JavaScript 库(如 jQuery)仍然可用，但是更推荐所有 JavaScript 都使用 Vue.js 的方式编写，解决网页中的问题。

Vue.js 虽然没有完全遵循 MVVM 模型，但是已经在很大程度上体现了 MVVM 的思想，这也是它能够取代上一代框架的突出优点。当 Vue 对象中的属性改变时，视图就会重新渲染，开发者只需要关注数据的变化，视图的更新是同步的。

【项目 2-1】

【项目描述】

完成第一个 Vue.js 网页，定义一个属性 message，在网页中输出该属性，输出一个数学表达式的值。

【项目实现】

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>第一个 Vue</title>
</head>
<body>
  <div id="app">
```

```
    <h2>{{ message }}</h2>
    <h2>{{ (3+5) * 5/4}}</h2>
  </div>
  <script src="./js/vue.js"></script>
  <script>
    var vm = newVue({
      el: '#app',
      data: {
        message: 'HelloWorld!'
      }
    })
  </script>
</body>
</html>
```

【项目说明】

(1)在 HTML 中,使用插值{{}}的形式引入 Vue 中定义的属性和方法,也可以引入 JavaScript 表达式等。

(2)实例化一个 Vue 对象,对象可以通过它的名称被引用,对象的名称多为 vm 或 app,vm 表示的是 MVVM 中最后两个字母 VM,说明该对象的功能为 ViewModel。

(3)Vue 对象的 el 属性表示使用 Vue 的 HTML 元素,可以只在网页中的某个元素中使用 Vue.js。如果要在整个网页中使用 Vue.js,需要设置所有的 HTML 元素为 el 的属性值对应的 HTML 元素的子元素,el 的属性值一般是一个 ID 选择器,常用名称为 #app 或 #box。

(4)使用 data 定义 Vue.js 的属性,data 的属性值为一个 JavaScript 对象,可以直接通过插值引用 data 对象的属性名称。data 对应着 MVVM 中的 Model。

(5)message 为 data 对象的一个属性,可以为 data 对象定义多个属性,属性之间用逗号分隔,message 的初始值为“Hello World!”。

(6)Vue 的代码一定要在引入 Vue.js 文件之后,推荐的位置为 HTML 中 body 的最后。

2.2 指令

指令是 Vue.js 的重要特性,可以使代码更加简洁、高效,可以使 HTML 元素和 Vue 对象发生关联,可以简化代码、明晰程序结构。指令作为 HTML 元素的属性存在,属性名称以“v-”开头,有的属性有属性值,有的属性不需要属性值。

2.2.1 基础指令

假设 Vue 对象有 msg 属性,常用基础指令的说明和示例如表 2-1 所示。

表 2-1　　基础指令

指令名称	含义	代码示例	备注
v-text	将 Vue 属性渲染为元素的内部文本，属性值为 Vue 对象的实例的属性		相当于 innerText
v-html	将 Vue 属性渲染为元素的内部 HTML		相当于 innerHTML
v-pre	显示元素内容的原始值，内容不进行编译	<span v-pre>{{msg}}	相当于<pre>，其中的代码不会被编译，显示其原始内容
v-once	只在第一次 DOM 时渲染，之后成为静态内容		相当于只绑定一次的 v-text

v-cloak	所有元素都渲染完毕后才显示，可防止插值表达式中的内容闪烁，和 CSS 配合使用	<div v-cloak>{{msg}}	和下列 CSS 配合使用：
			<style>
			[v-cloak]{
			display: none;
			}
			</style>

2.2.2　v-on

Vue 的 v-on 指令可以绑定事件，它对应着一个 JavaScript 事件名称，属性值为一个 Vue 的方法名或者是 JavaScript 表达式。v-on 可以对 HTML 元素的 JavaScript 事件进行监听，也可以对组件中的自定义事件进行监听。

1.缩写与示例

符号@是 v-on 的缩写，如@click 表示 v-on:click。

假设 fn 为 Vue 对象的 methods 中定义的方法名称，msg 为 Vue 对象中定义的属性名称，v-on 的几种语法示例如下：

(1)点击事件

(2)点击事件

(3)<span @click="fn()">点击事件

(4)<span @click.stop="fn()">停止冒泡

(5)<a @click.prevent="fn()">阻止默认事件

(6)<input @keydown.enter="fn()">按下 enter 键触发

(7)<input @keydown.65r="fn()">按下 65 对应的键

2.常用事件

Vue 中常用事件如下：

(1)v-on:click：鼠标点击。

(2)v-on:mouseout:鼠标离开 HTML 元素。

(3)v-on:mouseover:鼠标在 HTML 元素上。

(4)v-on:keydown:键盘上的按键按下。

(5)v-on:keyup:键盘上的按键按下后松开。

3.事件修饰符

事件之后可以定义修饰符,常见的修饰符如下:

(1). stop:阻止事件冒泡,不响应父元素的同名事件。

(2). prevent:阻止默认行为,a 等元素有默认行为。

(3)键盘按键,和键盘事件一起使用,常用的示例如@keyup. 13(回车)、@keyup. enter(回车)、@keyup. left(方向键左)、@keyup. delete(删除键)。

(4). capture:用捕获的事件模式代替默认的冒泡模式,先响应父元素事件,再响应子元素事件。

(5). self:事件只在定义了事件的元素触发,子元素不会触发该事件。

(6). once:事件只触发一次。

(7). passive:滚动的默认行为立即触发。

为事件定义事件修饰符的示例如下:

(1)<div v-on:click. stop. prevent="fn">

(2)<input v-on:keyup. 13="fn">

(3)<input v-on:keyup. alt. 67="fn">

(4)<div @scroll. passive="fn">

在为一个事件定义多个修饰符时,修饰符的定义的顺序不同,含义也可能会不同,例如@click. prevent. self 会阻止所有元素包括子元素的默认行为,而@click. self. prevent 只会阻止对元素自身的默认行为。

4.动态事件名称

也可以使用参数动态指定事件名称,如<a v-on:[eventName]="fn()"> ... ,其中动态参数 eventName 为 JavaScript 表达式,其运算的结果为要进行绑定的事件名称。

2.2.3 v-model

v-model 可以实现表单元素的双向绑定,当改变表单元素中的值的时候,Vue 对象的 data(MVVM 中的 Model)也会同时改变;当 Vue 对象的 data 中的表单元素绑定的属性变化的时候,视图(MVVM 中的 View)中的表单控件中的值也会发生变化。

v-model 只能应用于特定表单元素或包含特定表单元素的组件,特定表单元素包括 input、textarea、select。

v-model 可以添加修饰符,常用的修饰符如下:

(1). lazy:使用 change 事件替代双向绑定的默认的事件 keydown,不再是边输入边改变,而是失去焦点并且内容改变时才修改 Vue 对象中绑定的对应的属性的值。

(2). trim:去除表单控件输入的值前后的空格,一般不用在密码输入框上。

(3). number:将表单控件的值从默认的字符串转换为数字。

v-model 常见的代码示例如下：

(1)<textarea v-model ="description"></textarea>

(2)<input type="text" v-model.lazy="user">

(3)<input type="text" v-model.trim="user">

(4)<input type="text" v-model.num="price">

【项目 2-2】

【项目描述】

项目显示效果如图 2-1 所示。

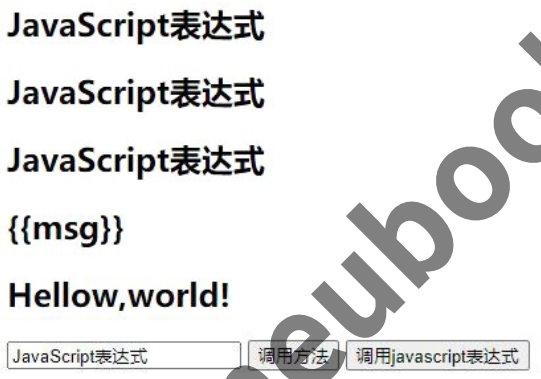

图 2-1　Vue 基础指令

实现一个基于外部 Vue.js 文件的项目，要求在网页中定义一个 Vue 的属性，定义一个 Vue 的方法，在页面中分别使用 v-text、v-html、v-once、v-prek 显示这个属性，使用 v-cloak 修饰所有元素。

在网页中定义一个文本输入框，文本输入框要求绑定 Vue 的属性，在改变文本输入框中的文字时，视图中的 v-text 等指令修饰的 HTML 元素中的文字也会同时改变。

在网页中定义两个按钮，点击两个按钮都会改变文本输入框绑定的 Vue 属性的值，一个按钮使用 methods 中的方法响应事件，一个按钮直接在属性值中写 JavaScript 表达式定义事件。两个按钮分别使用 v-on:click 和缩写形式@click 定义。

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>vue.js 基本指令</title>
    <style>
        [v-cloak] {
            display: none;
        }
    </style>
```

```
</head>
<body>
    <div id="app" v-cloak>
        <h2 v-html="msg"></h2>
        <h2 v-text="msg"></h2>
        <h2>{{msg}}</h2>
        <h2 v-pre>{{msg}}</h2>
        <h2 v-once>{{msg}}</h2>
        <input type="text" v-model="msg">
        <button @click="changemsg()">调用方法</button>
        <button v-on:click="msg='JavaScript 表达式'">调用 javascript 表达式</button>
    </div>
    <script src="./vue.js"></script>
    <script>
        var vm = new Vue({
            el: "#app",
            data: {
                msg: 'Hellow,world!',
            },
            methods: {
                changemsg() {
                    this.msg = "开始学习 vue!"
                }
            }
        });
    </script>
</body>
</html>
```

【项目说明】

v-cloak需要和CSS配合使用,它能解决页面闪烁的问题,适应于复杂的场景,在这个项目中,只是给出它的应用语法,没有起到作用。

Vue对象的属性定义在data中,方法定义在methods中。在方法中使用属性时,一定要使用this来引用属性,方法的另外一种定义形式的代码示例如下:

```
changemsg:function(){…}
```

2.2.4 v-if 和 v-show

v-if是条件渲染,可以根据它的属性值确定是否渲染特定的HTML元素。单纯在视觉的层面,如果渲染HTML元素,则该HTML元素显示;如果不渲染,则该HTML元素隐藏。v-if在切换过程中,HTML元素内的事件监听和子组件也会被销毁和重建。

如果v-if的属性值为false,则不会对HTML元素进行任何渲染,页面中没有该HTML元素,并不是该HTML元素处于隐藏状态。

v-if 可以和 v-else 一起使用，v-else 表示 v-if 的条件为 false 时渲染的 HTML 元素，如果有多重嵌套的条件，可以使用 v-else-if。v-else 只有在 v-if 或者 v-else-if 后面的时候才有意义。

假设 a、b、c 都是 Vue 对象的属性，示例代码如下：

```
<div v-if='a==b'>狮子</div>
<div v-else-if='b==c'>东北虎</div>
<div v-else>华南虎</div>
```

v-show 是根据其属性值的真假，显示或隐藏 HTML 元素，它的隐藏元素是通过设置 CSS 属性的 display 属性来实现的。display 为 none 的时候，HTML 元素隐藏；display 为 HTML 元素默认值(inline 或者 block)的时候，HTML 元素显示。

假设 visible 为 Vue 对象的属性，v-show 的代码示例如下：

```
<div v-show="visible">名称为 div 的 HTML 元素</div>
```

v-if 和 v-show 都可以实现对 HTML 元素的显示和隐藏，但是两者的本质并不相同。v-if 是条件渲染，属性值为 false 时，对应的 HTML 元素并不被加入 DOM 中；而 v-show 在属性值为 false 时，对应的 HTML 元素已被加载到网页中，只是因为 CSS 的属性设置而不能被看见。v-if 在切换状态时，开销较大；v-show 在初始加载时，开销较大。频繁地切换状态时，适合使用 v-show；状态较少改变时，适合使用 v-if。

2.2.5　v-for

v-for 是一个常用的指令，可以遍历 JavaScript 数组、对象或者单纯地完成循环。v-for 的属性值通常为 in 的形式，也可以使用 of，代码示例如下：

```
(1)item in arr
(2)key of obj
```

在使用 in 作为属性值的时候，in 的前面可以包括 3 个参数，分别是 value(值)、key(键)和 index(索引)，一般情况下，使用前两个参数即可。in 后面可以是数组和整数。

v-for 的代码示例如下：

```
(1)<h2 v-for="n in 10"> {{n}} </h2>
(2)<div v-for="(val,key) in arr">{{key}}:{{val}}</div>
(3)<div v-for="(v,k) in list">{{v.id}}  :{{v.name}}</div>
(4)<p v-for="(item,i) in list">{{i}} :{{item}}</p>
(5)<div v-for="(value, name, index) in obj">
    {{ index }}. {{ name }}: {{ value }}
   </div>
```

在使用 v-for 的时候，除了简单的场景，建议同时定义 key 属性，使用 key 属性可以识别循环中的每一个节点，便于增加扩展功能。

key 属性通过 v-bind 的形式绑定，它的属性值为唯一取值的数字或字符串，多数情况下，根据遍历的元素的属性或索引给定值。代码示例如下：

```
<div v-for="item in arr" :key="item.id">
```

在需要循环渲染多个 HTML 元素的时候，v-for 经常和 template(模板)一起使用，

template在编译后不会生成任何HTML元素，只有逻辑性的作用。示例代码如下：

```
< template v-for="(v,i) in arr">
  <span>{{i}}</span>
  <span>{{v}}</span>
</template>
```

在某些场景下，v-for需要和v-if一起使用。两者同时使用时，v-for优先起作用，代码示例如下：

```
<div v-for="v in arr" v-if="v.flag">
          {{ v.title }}
</div>
```

【项目2-3】

【项目描述】

项目显示效果如图2-2所示，项目为一个使用Vue的tab选项卡，只实现了下面内容的变化。当点击上方的国家名称时，下面就会显示相应国家的城市。要求上面的国家名称、下面的城市名称都是通过data中定义的数据显示。

图2-2 基于Vue的tab选项卡

【项目实现】

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Vue tab</title>
    <script src="vue.js"></script>
    <style>
        ul{
            padding:0;
            margin:0;
            list-style: none;
        }
        #box{
            width:300px;
        }
        #nav {
```

```
            display: flex;
        }
        #nav span {
            width: 33%;
            margin-left: 2px;
            background: #111;
            color: #fff;
            text-align: center;
            height: 30px;
            line-height:30px;
        }
        #box ul {
            display: flex;
            flex-wrap: wrap;
            background:#eee;
            padding:15px;
        }
        #box li {
            width: 100px;
            margin: 5px;
            border: 1px solid #ccc;
            border-radius: 5px;
            height: 30px;
            background: linear-gradient(#000, #fff);
            color: #fff;
            font-size: 14px;
            text-align: center;
            line-height: 30px;
        }
    </style>
</head>
<body>
    <div id="box">
        <div id="nav">
            <span v-for="(v,i) in nation" @click="index=i" :key=i>
                {{v.name}}城市
            </span>
        </div>
        <ul>
            <li v-for="v in nation[index].area" :key="i">
                {{v}}
            </li>
```

```
        </ul>
    </div>
    <script>
        var vm = new Vue({
            el: "#box",
            data: {
                nation: [{
                        'name': '中国',
                        area: ["武汉", "长春", "大连", "青岛", "兰州", "沈阳"]
                    },
                    {
                        'name': '美国',
                        area: ["华盛顿", "休斯顿", "纽约", "波特兰"]
                    },
                    {
                        'name': '英国',
                        area: ["伦敦", "利物浦", "曼彻斯特","爱丁堡"]
                    }
                ],
                index: 0
            }
        })
    </script>
</body>
</html>
```

【项目说明】

(1)Vue中应该关注的不是视图的变化,而是数据的变化,一切的视图变化都要将其抽象为数据的变化,根据视图进行数据的定义。数据的定义方法有多种,本项目采用了二维数组的定义方法。

(2)本项目根据tab项的切换定义了index属性,通过改变该属性的值改变下面的具体城市名称。

(3)定义v-for的时候建议动态绑定key属性,其属性值的预期是一个不重复的值。

2.2.6 v-bind

v-bind可以动态地设置HTML的属性值,使HTML的属性值为Vue对象的属性或者JavaScript表达式。经常使用v-bind进行动态设置的HTML属性包括class、style、key、src、href等。

v-bind可以缩写为“:”。假设img和link为Vue对象的属性,v-bind的基本语法示例如下:

(1)<img v-bind:src="img">

(2)<img :src="img">

(3)<a :href="link">超链接

上面代码中的(1)和(2)的含义相同,分别是 v-bind 和 v-bind 的缩写形式。

Vue 对 class 和 style 两个 HTML 属性的功能进行了额外的扩展和增强,它们的属性值具有更大的灵活性,可以使用对象和数组作为属性值。

假设 isA、isB、c1、c2 分别为 Vue 对象的 data 中定义的属性,v-bind 绑定 class 和 style 属性的代码示例和说明如下:

(1) :当 isA 为 true 时,div 有名为 cur 的 class。

(2)<div :class="[c1, c2]"> :div 同时定义了 c1 和 c2 属性的属性值对应的类。

(3)<div :class="[{ cur: isA}, c1]"> :isA 为 truthy 时绑定类 cur,绑定 c1 的属性值对应的类。

(4) :定义 style 的 width 的值为 c1 对应的属性值。

(5)<div :style="{ color: c1, fontSize: c2 + 'px' }"> :style 的 color 的属性值为 c1,font-size 的属性值为 c2。多单词的 CSS 属性名称使用驼峰式的 fontSize,不使用 font-size。

(6) :style 的属性值中包括两个 CSS 代码块,分别是 data 的属性 c1 和 c2 的属性值。

上面的说明中提到了 truthy,truthy 和 falsy 是布尔值的 JavaScript 中自动类型转换的布尔值的扩展。在布尔值的上下文中,truthy 是转换后的值为 true 的值,falsy 是转换后的值为 false 的值,常见的 falsy 值包括 flase、0、null、undefined、NaN、空字符串、document.all,不是 falsy 的其他值为 truthy。

布尔值上下文中,falsy(前两个)和 truthy(后 3 个)的代码示例如下:

```
(1)if(null)
(2)if(NaN)
(3)if(1000)
(4)if({})
(5)if([])
```

v-bind 也支持动态属性名称,属性名称可以为一个 JavaScript 表达式,通过程序逻辑动态获得绑定的属性的名称,如果表达式的值为 null 则移除绑定。JavaScript 表达式写在中括号内,代码示例如下:

```
<div v-bind:[JavaScript 表达式]="c1"> ... </div>
```

绑定的属性名称为 JavaScript 表达式的值,属性值为 data 中定义的属性 c1。

【项目 2-4】

【项目描述】

项目显示效果如图 2-3 所示。上面有 6 个按钮，下面有一个盒子。点击“显示”按钮，隐藏的盒子会显示；点击“隐藏”按钮，显示的盒子会隐藏；点击“切换”按钮，隐藏的盒子会显示，显示的盒子会隐藏；点击“增加类”按钮，盒子会增加一个 CSS 类（蓝色背景，名称为.blue）；点击“删除类”按钮，盒子上给定的 CSS 类会被删除（蓝色背景消失）；点击“切换类”按钮，如果盒子已经定义了指定的 CSS 类，则删除该类，如果没有定义 CSS 类，则为盒子定义该 CSS 类。

图 2-3 Vue 的切换与切换类

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>vue.js</title>
    <style>
        #box {
            width: 300px;
            height: 300px;
            border: 2px solid blue;
            margin: 10px;
        }
        .blue {
            background-color: blue;
            box-shadow: 5px 5px 5px blue;
        }
    </style>
</head>
```

```
<body>
    <div id="app">
        <button @click="show()">显示</button>
        <button @click="hide()">隐藏</button>
        <button @click="toggle()">切换</button>
        <button @click="isBlue=true">增加类</button>
        <button @click="delClass()">删除类</button>
        <button v-on:click="isBlue=! isBlue">切换类</button>
        <div id="box" :class="{blue:isBlue}" v-if="seen"></div>
    </div>
    <script src="./vue.js"></script>
    <script>
        var vm = newVue({
            el: "#app",
            data: {
                seen: true,
                isBlue: true
            },
            methods: {
                show() {
                    this.seen = true;
                },
                hide() {
                    this.seen = false;
                },
                toggle() {
                    this.seen = ! this.seen;
                },
                delClass: function () {
                    this.isBlue = false;
                }
            }
        });
    </script>
</body>
</html>
```

【项目说明】

事件的定义可以使用 v-bind 或 v-bind 的简写@，事件的响应可以使用方法或 JavaScript 表达式。

对视图的变化需要定义对应的数据，不需要在 DOM 的层次考虑问题，需要切换思路在数据的层次考虑解决问题的方法。本项目中分别定义了数据 seen 和 isBlue 来控制盒子的切换过程和类的切换过程。

2.3 计算属性

计算属性是 Vue 中对象的特殊的属性，名称为 computed。computed 在定义时和 methods、data 或 data()等并列，它的使用方式和 data 中定义的其他属性相同，但它不是对象的 data 中定义的属性，它是通过逻辑运算计算出来的属性，它相当于对象的一个方法的运算结果，但是这个结果会有缓存，可以快速地被访问到。计算属性用以封装相对复杂的逻辑，使这些逻辑可以快速、方便地被使用和复用。

计算属性相当于 Vue 中对象的一个方法，在使用时按照属性的使用方法使用，它的值会在调用后被缓存，下次调用直接使用被缓存的值，只有当它依赖的属性等发生变化时，才重新对计算属性进行求值。计算属性对应的方法的内部不建议进行异步操作，如 AJAX、本地存储的读写等。

一般的计算属性只能进行读取(get)，如果要对计算属性进行赋值操作，需要单独定义 set，定义具备 setter 和 getter 功能的计算属性 dpname 的示例代码如下：

```
computed: {
dpname: {
    get: function () {
        return ;
},
        set: function (newValue) {
    }
  }
}
```

【项目 2-5】

【项目描述】

项目显示效果如图 2-4 所示，图中显示了一个购物车，购物车中有各种商品，可以点击商品后面的按钮增加或减少商品数量，可以点击“删除”按钮从购物车中删除商品。所有商品的总价格在下方显示，商品数量变化时，总价格自动改变。

图 2-4 购物车

要求商品数据使用数组表示，“减少数量”按钮减少到 0 后就不能再被点击。

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>购物车</title>
    <style>
        ul {
            list-style: none;
        }
        #app {
            min-width: 1200px;
        }
        #app li {
            width: 900px;
            padding: 10px;
            margin: 10px;
            border: 1px solid #ccc;
            height: 40px;
        }
        #app li span {
            background-color: #c00;
            color: #fff;
            padding: 5px;
            margin: 5px;
            width: 110px;
            display: inline-block;
            text-align: center;
            user-select:none;
        }
    </style>
</head>
<body>
    <div id="app">
        <h2>购物车</h2>
        <ul>
            <li v-for="(v,i) in goods" :key=i>
                <span>序号:{{i+1}}</span>
                <span>品名:{{v.name}}</span>
                <span>价格:¥{{v.price}}</span>
                <span>数量:{{v.count}}</span>
```

```
            <span>{{v.desc}}</span>
            <button @click="add(i)">增加数量</button>
            <button @click="minus(v)" :disabled="v.count==0">减少数量</button>
            <button @click="del(i)">删除</button>
        </li>
    </ul>
    <h3>总价格:{{total()}}</h3>
    <h3>总价格:{{totalp}}</h3>
</div>
<script src="./vue.js"></script>
<script>
    var vm = new Vue({
        el: "#app",
        data: {
            index: 0,
            goods: [{
                    name: "apple",
                    price: 5.5,
                    count: 6,
                    desc: "文字描述"
                },
                {
                    name: "orange",
                    price: 7,
                    count: 3,
                    desc: "商品说明"
                },
                {
                    name: "pear",
                    price: 6,
                    count: 2,
                    desc: "莱阳梨"
                },
                {
                    name: "banana",
                    price: 3,
                    count: 7,
                    desc: "价格便宜"
                }
            ]
        },
        methods: {
```

```
            add: function (i) {
                this.goods[i].count++;
            },
            minus: function (item) {
                item.count--;
            },
            del: function (i) {
                this.goods.splice(i, 1);
            },
            total: function () {
                let sum = 0;
                for (let i = 0; i < this.goods.length; i++)
                    sum += this.goods[i].price * this.goods[i].count;
                return sum;
            }
        },
        computed: {
            totalp: function () {
                let sum = 0;
                for (let i = 0; i < this.goods.length; i++)
                    sum += this.goods[i].price * this.goods[i].count;
                return sum;
            }
        }
    });
</script>
</body>
</html>
```

【项目说明】

总体思路是通过 v-for 将对象数组渲染在页面中。在定义 v-for 的时候，建议动态绑定 key 属性，key 属性不能定义在 template 上。

totalp 为计算属性，total 为方法。可以看出，在本项目中，它们的显示效果完全相同；它们的不同之处在于，计算属性会被缓存，方法每次都被调用和执行。

":disabled="v.count==0"表示当 v.count 为 0 的时候 disabled 属性值为 disabled，可以简写为:disabled="! v.count"，通过给 button 设置该属性使数量为 0 时按钮不能再被点击。

【项目 2-6】

【项目描述】

项目显示效果如图 2-5 所示，项目为一个简化版的 todolist，即要做的事情的计划列表。

图 2-5 todolist

在项目中,要完成下列功能:

(1)点击增加按钮可以增加要做的事情,输入完毕点击回车键也可以添加要做的事情。

(2)双击要做的事情名称可以修改事情的内容,修改完毕按回车键或失去焦点可以保存内容的修改,按 esc 键可以不保存内容的修改。

(3)鼠标到每一行上的时候,在这一行的最右边显示“删除”按钮,点击可以删除这一行;鼠标不在某一行上的时候,“删除”按钮不出现。

(4)可以点击事情前面的 checkbox 选择要做的事情是否已做完,可以自动统计还有多少件没有做的事情。

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>todo-list</title>
    <style>
        .box {
            width: 800x;
            height: 30px;
            padding: 10px;
            margin: 10px;
            border: 1px solid #ccc;
            user-select:none;
        }
        .box span {
            display: inline-block;
            width: 150px;
```

```
            background-color: #030;
            color: #fff;
            text-align: center;
            line-height: 26px;
        }
        .box button {
            display: none;
            float: right;
        }

        .box:hover button {
            display: inline-block;
        }
        .name {
            width: 150px;
            display: none;
        }
        .cur .name {
            display: inline-block;
        }
        .cur span {
            display: none;
        }
        .cur {
            border-color: #c00;
        }
    </style>
</head>
<body>
    <div id="app">
        <h1>todolist</h1>
        <div id="form">
            <input v-model="input" @keydown.enter="add">
            <button @click="add">增加</button>
        </div>
        <div v-for="(v,i) in list" class="box" :class="{cur:index==i}">
            <input type="checkbox" v-model="v.done">
            <span @dblclick="update(i)">{{v.name}}</span>
            <input v-model="v.name" class="name" @keyup.enter="save(i)" @blur="save
(i)" @keyup.esc="undo(i)">
            <button @click="del(i)">删除</button>
        </div>
```

```
    <h3>共有{{tcount}}件未完成任务</h3>
</div>
<script src="./vue.js"></script>
<script>
    var vm = new Vue({
        el: "#app",
        data: {
            list: [{
                    name: "理发",
                    done: false
                },
                {
                    name: "学习 vue",
                    done: true
                },
                {
                    name: "学习 Java",
                    done: false
                },
                {
                    name: "学习 node",
                    done: true
                }
            ],
            index: -1,
            input: "",
            before:""
        },
        methods: {
            add: function () {
                if (this.input ==="") return;
                let item = {
                    name: this.input,
                    done: false
                }
                this.list.push(item);
                this.input ="";
            },
            update: function (i) {
                console.log(i);
                this.index = i;
                this.before = this.list[i].name;
```

```
            },
            save: function (i) {
                this.index = -1;
            },
            undo: function (i) {
                this.list[i].name = this.before;
                this.before ="";
                this.index = -1;
            },
            del(i) {
                this.list.splice(i, 1);
            }
        },
        computed: {
            tcount() {
                return this.list.filter(function (v){
                    return ! v.done;
                }).length;
            }
        }
    });
  </script>
</body>
</html>
```

【项目说明】

完成项目的关键是定义好数据，将视图中的行为转换为数据，定义 index 属性表示正在操作的是第几行；定义 before 属性保存修改前的数据，便于按 esc 键后恢复修改的数据；定义 input 属性保存新增的内容。

@keyup.enter="save(i)"表示在点击回车键之后调用 save 方法，参数为 i。

:class="{cur:index==i}"表示当 index 和 i 的值相等时，类 cur 起作用，同时定义 CSS 选择器 .box:hover button{display:inline-block;}，就可以当鼠标在某一行上时，使本来隐藏(display:none)的删除按钮显示出来。

项目的扩展功能如下：

(1)增加“已完成”按钮或链接，点击后可以显示所有已完成的事情。

(2)增加“未完成”按钮或链接，点击后可以显示所有未完成的事情。

(3)增加“查看所有”按钮或链接，点击后可以显示所有的事情。

(4)增加已完成的事情数量的统计，实时显示。

2.4　watch

watch 一般被称为侦听器或者监听，它是和 computed、methods 等并列的 Vue 对象的属

性。watch 可以监听对象的 data 中定义的属性，当被监听的属性变化的时候，执行 watch 函数，可以在 watch 函数中根据被监听的属性进行自定义的逻辑运算。

当一个属性改变，其他的属性也需要相应改变时，可以使用 watch。计算属性也会在其所依赖的属性变化时自动变化，这覆盖了 watch 要解决的问题的大多数场景，所以在多数此类应用场景下，会使用计算属性而不是 watch。watch 适合于包括异步等开销较大的操作的场景，如 AJAX、本地存储读写等。

使用 watch 时，当第一次绑定值时，不会执行 watch 函数；当值发生变化时，watch 函数会被执行。如果需要第一次绑定值的时候就执行 watch 函数，可以为 watch 定义 immediate 属性。

在默认情况下，使用 watch 监听一个对象类型的属性时，watch 不能够监听到对象内部属性的改变，需要为 watch 定义 deep 属性才能够对对象的内部属性进行监听。

监听 p1 和 p2 属性的 watch 的语法示例如下：

```
watch : {
  p1:function(val) {  },
  P2:function(val) {  }
}
```

包括 immediate 和 deep 属性的 watch 的示例代码如下：

```
watch: {
      p1: {
      handler(newName, oldName) {      },
      deep: true,
      immediate: true
    }
  }
```

上述代码中，p1 为监听的 data 中定义的属性的名称，handler 为处理监听的方法，参数 newName 表示属性 p1 改变之后的值，参数 oldName 表示属性 p1 改变之前的值。也可以使用 $.watch 定义监听过程，$.watch 的返回值是一个取消观察函数，用来停止对属性的监听。

假设 vm 为 Vue 对象的名称，使用 $.watch 定义监听和解除监听的示例代码如下：

```
var unWatch = vm.$watch('text',function (newValue, oldValue) {  })
unWatch();
```

【项目 2-7】

【项目描述】

项目显示效果如图 2-6 所示。全名(fullName)是由 firstName 和 lastName 组成的，页面中有 3 个文本输入框，分别可以修改简单的字符串类型的属性 firstName、lastName 和复杂的 city 对象的属性 state。

输入框下面的 3 行文字分别显示使用 watch 方法维护的属性 fullName(全名)、使用计算属性定义的属性 qm(全名)和 city 对象的属性 state。

图 2-6　Vue watch

fullName 通过监听(watch)firstName 属性和 lastName 属性维护它的值，会自动根据这两个属性的变化而变化。对象类型的属性 city 需要进行特殊的处理，使其子属性 state 的变化也可以被监听到。

初始数据的 firstName 属性的值为 Miacal，在对 city 对象进行 watch 的时候将 firstName 改为 city 对象的 state 属性的值，立即生效。

通过 $watch 监听 firstName 属性，firstName 属性改变时弹出 alert 警告框；9 秒后解除对 firstName 的监听，不再弹出警告框。

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>watch</title>
</head>
<body>
    <div id="app">
     <div>firstName: <input v-model="firstName"></div>
     <div>LastName: <input v-model="lastName"></div>
     <div>state:  <input v-model="city.state"></div>
        <h2>watch 全名:{{fullName}}</h2>
        <h2>计算属性全名:{{qm}}</h2>
        <h2>对象的属性:{{city.state}}</h2>
    </div>
    <script src="./vue.js"></script>
    <script>
     var vm=new Vue({
         el:"#app",
         data:{
             firstName:"Miacal",
             lastName:"Jordan",
             fullName:"",
             city:{state:"Illinois",name:"Chicago"}
```

```
        },
        watch:{
            firstName:function(val){
                this.fullName=val+" "+this.lastName;
            },
            lastName(val){
                this.fullName=this.firstName+" "+val;
            },
            city:{
                handler(newv,oldv){
                    this.firstName=newv.state;
                },
                deep:true,
                immediate:true
            }
        },
    computed:{
        qm:function(){
            return this.firstName+" "+this.lastName;
        }
    }
    });
    var uw=vm.$watch("firstName",function(newv,oldv){
              alert(newv+":"+oldv);
    });
    setTimeout(uw,9000);
    </script>
</body>
</html>
```

【项目说明】

项目中 watch 了 3 个属性:firstName、lastName 和 city。默认情况下,对象类型的属性 city 的改变不能被监听,必须设置属性 deep:true 才可以被监听。

监听默认是在第一次值的改变的时候触发,如果要使监听在绑定的时候就触发,需要设置 immediate:true。所以页面中初次显示的 firstName 的值不是 data 中定义的"Miacal",而是在 city 的 watch 函数中赋值的 newv.state,即 city 的 state 属性,属性值为"Illinois"。

$watch 中默认不是深度监听,不能监听对象类型的 city,它的两个参数分别表示监听的属性改变后的和改变前的值。$watch 的返回值可以解除监听,它和 watch 属性的功能基本相同。

在本项目中,watch 中赋值的变量 fullName 和全局属性 qm 都能够维护变化的 firstName 和 lastName 组成的结果,这两种方法功能相似,但异步操作建议定义在 watch 中。

2.5　过滤器

过滤器可以对输入的数据进行处理。过滤器一般应用在插值表达式中，对插值表达式的结果进行文本格式化。

过滤器可以定义为全局的过滤器和局部的过滤器，使用 Vue.filter()方法定义全局的过滤器，为 Vue 对象定义 filters 属性可以定义局部过滤器。局部过滤器只能被一个 Vue 对象使用，全局过滤器可以被作用范围内的多个 Vue 对象使用。过滤器可以定义一个或多个参数，在使用的时候给定参数的具体的值。

名称为 fname 的全局过滤器的代码示例如下：

```
Vue.filter('fname', function (value) { });
```

局部过滤器使用 filters 定义，和 methods、computed 等并列。名称为 fname 的局部过滤器的无参数和有参数的代码示例分别如下：

```
(1)filters: { fname: function (value) {  } }
(2)filters: { fname: function (value,参数 1,参数 2) {  } }
```

过滤器在插值表达式中使用管道运算符的方式，可以同时使用多个过滤器，每个过滤器左边的内容为该过滤器的输入。

假设 msg 为 Vue 对象中定义的属性，过滤器在插值表达式中使用的代码示例分别如下：

```
(1){{ msg | fname}}
(2){{ msg | fname | fname}}
(3){{ msg | fname('参数') }}
```

【项目 2-8】

【项目描述】

项目显示效果如图 2-7 所示。图中应用了 3 个不同的过滤器，分别是全局定义的过滤器 datefmt、局部定义的过滤器 rs 和 dl。datefmt 将日期转换为 yyyy-mm-dd 的形式，rs 将字符串变为倒序，dl 在字符串前加上字符“¥”。其中，最后一行中多次使用了 dl 定时器。

图 2-7　Vue 过滤器

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Vue filter</title>
    <style>
        h2{
            width:300px;
            border:1px solid #039;
            padding:5px;
            text-align:center;
        }
    </style>
</head>
<body>
    <div id="app">
      <h2>{{now |datefmt("-")}}</h2>
      <h2>{{now |datefmt("  ")}}</h2>
      <h2>{{msg | rs}}</h2>
      <h2>{{188.6 | dl}}</h2>
      <h2>{{78.99 | dl | dl | dl | dl}}</h2>
    </div>
    <script src="./vue.js"></script>
    <script>
       Vue.filter("datefmt",function(input,sc){
            let date=new Date(input);
            let year=date.getFullYear();
            let mon=date.getMonth()+1;
            let day=date.getDate();
            if(mon<10)mon='0'+mon;
            if(day<10)day='0'+day;
            return year+sc+mon+sc+day;
       })
       var vm=new Vue({
            el:"#app",
            data:{
                msg:"yuanlei",
                now:new Date()
            },
            filters:{
                rs:function(input){
```

```
            return input.split("").reverse().join("");
        },
        dl(input){
            return "¥"+input;
        }
    }
})
</script>
</body>
</html>
```

【项目说明】

(1)过滤器的第一个参数 input 为过滤器的管道形式输入的值。

(2)过滤器第二个参数是过滤器的参数,如全局注册的过滤器 datefmt 添加了参数 sc,含义为日期间的分隔符号。使用 datefmt("-")的形式调用参数。

(3){{78.99 | dl | dl | dl | dl}}使用了多个过滤器,使用时,符号"|"前面的内容即为后面的内容的输入,每用一次 dl 过滤器,就会在输入前增加一个字符"¥"。

2.6　自定义指令

Vue 指令功能强大,便于使用。除了系统提供的 Vue 指令外,也可以自定义 Vue 指令,扩展功能,简化编码过程。自定义指令包括全局指令和局部指令,也可以为指令指定参数。全局指令的代码示例如下:

```
Vue.directive('dname', {
  bind: function(){},
  inserted: function(){},
  update: function(){},
  componentUpdated: function(){ },
  unbind: function(){}
})
```

局部自定义指令使用 directives 属性来定义,和 methods、computed 等并列,其代码示例如下:

```
directives: {
  dname: {
    inserted: function (el) { }
      }
}
```

使用自定义指令的代码示例如下:

(1) HTML 元素

(2)<p v-dname>HTML 元素</p>

在定义自定义指令的时候,用到了钩子(hook)函数。钩子函数是 Vue 中经常使用的一

种函数，它可以将自定义的代码加入对象的生命周期的某个过程中，从底层扩展对象的功能。

自定义指令中常用的钩子函数及其说明如下：

(1)bind：指令第一次绑定 HTML 元素时，对元素进行初始化，只会调用一次。

(2)inserted：定义指令的元素插入父节点时被调用。

(3)update：定义指令的元素模板更新时被调用，无论绑定值是否发生变化。

(4)componentUpdated：被绑定元素所在模板完成一次更新周期时调用。

(5)unbind：自定义指令与 HTML 元素解除绑定，只调用一次。

钩子函数的常用参数如下：

(1)el：自定义指令绑定的元素。

(2)binding：绑定对象，包括子属性 name(指令名称)、value(指令值)、expression(指令表达式)等，value 一般在属性值为绑定的变量的时候使用，expression 一般在属性值为字符串或表达式的时候使用。

一般来说，定义自定义指令的时候不需要定义所有钩子函数，如果 bind 和 update 相同，可以简写为一个函数。

【项目 2-9】

【项目描述】

项目显示效果如图 2-8 所示。项目中有 3 个 HTML 元素，分别为 HTML 元素定义了全局的自定义指令和局部的自定义指令，全局的自定义指令可以动态指定 HTML 元素的文字颜色，局部的自定义指令可以动态指定 HTML 元素的背景颜色。颜色由 Vue 的变量给定，点击“随机颜色”按钮，全局指令的文字颜色和局部指令的背景颜色都会改变。

图 2-8 Vue 自定义指令

【项目实现】

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Vue directive</title>
    <style>
        div{
            width:400px;
```

```
            margin:10px;
            padding:10px;
            text-align:center;
            transition:all .6s;
        }
    </style>
</head>
<body>
<div id="app">
            <div v-text-color="color">全局指令</div>
            <div v-bg-color="color">局部指令</div>
            <div v-bg-color="color">局部指令</div>
            <button @click="changeColor">随机颜色</button>
        </div>
        <script src="./vue.js"></script>
        <script>
            Vue.directive('text-color', function(el, bind, vNode){
                el.style.color= bind.value;
                el.style.fontSize="100px";
            });
            var app = new Vue({
                el: '#app',
                data: {
                    color: '#aaa'
                },
                methods: {
                    changeColor: function(){
                        this.color ="#"+(Math.floor(Math.random() * 900)+100);
                    }
                },
                directives: {
                    "bg-color":function(el,bind,vnode){
                        el.style.background=bind.value;
                        el.style.color="#fff";
                    }
                }
            });
    </script>
</body>
</html>
```

【项目说明】

(1)项目定义了全局自定义指令 v-bg-color 和局部自定义指令 v-text-color,自定义指令

的使用方式和普通 Vue 指令相同。

(2)使用 bind. value 获得动态绑定的(来自 data)自定义指令的属性值,使用 bind. expression 获得静态的属性值,参数 el 表示定义自定义属性 HTML 元素。

(3)"#"+(Math. floor(Math. random() * 900)+100)可以获得#100 到#999 的随机颜色值,这是全部随机颜色中的一部分,不是所有随机颜色。

2.7 transition

2.7.1 transition 概述

Vue 使用 transition 为元素和组件的进入和离开(显示和消失)提供过渡的动画效果。HTML 元素或组件的进入和离开的主要场景是使用 v-if 和 v-show、动态组件、组件根节点的时候,如果为 HTML 元素或组件定义了 transition,则显示和消失将以动画的方式进行。

在 HTML 中使用 transition 元素定义过渡,没定义 name 属性和定义了 name 属性的过渡的示例代码分别如下:

(1)没定义 name 属性

```
<transition>
    <div v-if="show">以动画方式出现和消失的元素</div>
</transition>
```

(2)定义了 name 属性

```
<transition name = "tname">
   <div v-if="show">以动画方式出现和消失的元素</div>
</transition>
```

transition 需要设置 CSS 配合使用才能起作用,CSS 选择器的名称是预先定义好的,修饰 transition 的变化过程或变化的起始或结束状态。

transition 的过程包括进入(enter)和离开(leave),一个基础的 transition 的示例如图 2-9所示。在进入前,v-enter 表示进入的初始状态(如 opacity 为 0),v-enter-to 表示进入的结束状态(如 opacity 为 1);v-enter 使 HTML 元素隐藏,v-enter-to 不要设置和 HTML 元素默认显示效果有冲突的属性。

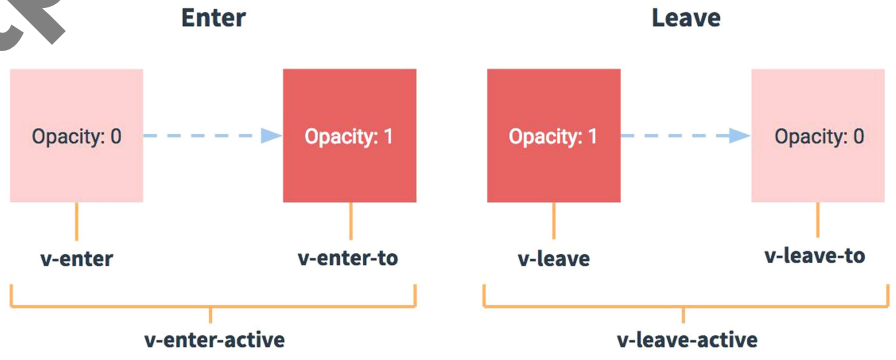

图 2-9 Vue transition 的基本过程

同样，v-leave 表示 transition 的离开的初始状态，此时的 opacity 为 1，这也是 HTML 元素的默认的 opacity 值。很多情况下，v-leave 和 v-enter-to 不需要额外设置，额外的设置可能和默认的 HTML 元素的显示效果冲突，使动画效果不平滑。v-leave-to 表示 transition 的离开过程的终止状态。

定义 Vue transition 的时候，是给组件或元素一个进入和离开的动画效果，这和传统的 CSS3 transition 不是完全相同的。只考虑进入和离开的动画效果，是设计 Vue transition 的基本思路。

定义 Vue transition 的时候，经常使用 opacity、transform 等 CSS 属性，也可以使用 animate.css 中出现和消失的动画效果。

v-enter-active 和 v-leave-active 表示动画变化的过程，需要使用 CSS3 的 transition 或 animation 属性定义这个变化过程，animation 需要另外定义@keyframes。

2.7.2　transition 的类名

1.没有 name 属性的类名

对没有设置 name 属性的 transition，需要设置下列名称的 CSS 类选择器：

(1)v-enter：定义 transition 进入的开始状态，动画的第一个关键帧，此时的元素多为隐藏的状态，如 opacity 为 0、width 为 0、left 为负值等。

(2)v-enter-active：定义进入动画的过程，包括 transition 的时间、延迟时间和动画曲线函数，动画的初始状态和结束状态分别由 v-enter 和 v-enter-to 定义。

(3)v-enter-to：transition 的结束状态，此时的元素多为显示的状态，如 opacity 为 1、width 为固定长度等。

(4)v-leave：定义离开的 transition 的初始状态，离开过程的第一个关键帧，元素为可见。

(5)v-leave-active：定义离开动画的过程，包括 transition 的时间、延迟时间和动画曲线函数，动画的初始状态和结束状态分别由 v-leave 和 v-leave-to 定义。

(6)v-leave-to：离开过渡的结束状态，元素为不可见。

【项目 2-10】

【项目描述】

项目显示效果如图 2-10 所示，点击按钮，则会触发下面的文字的显示或隐藏效果。如果文字隐藏，则以动画方式显示；如果文字显示，则以动画方式消失。

动画属性要求包括透明度(opacity)和任意 transform 的效果，要求动画过程平滑、不突兀，出现和消失的动画持续时间均为 1 秒。

图 2-10　没有 name 的 transition 动画

【项目实现】

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Vue transition</title>
    <script src="vue.js"></script>
    <style>
     .box{
            font-size:40px;
            color:#039;
        }
       .v-enter,.v-leave-to{
           opacity:0;
           transform:rotateX(90deg) translateX(100px);
       }
        .v-enter-active,
        .v-leave-active {
            transition: all 1s;
        }
    </style>
</head>
<body>
    <div id="app">
        <button @click="show=! show">点击动画</button>
        <transition>
            <div v-show="show" class="box">动画实例</div>
        </transition>
    </div>
    <script>
        var vm = new Vue({
            el: '#app',
            data: {
                show: true
            }
        });
    </script>
</body>
</html>
```

【项目说明】

(1)没有设置 v-enter-to 和 v-leave,这两个状态以 HTML 元素的默认样式决定,这样的

动画设计简单、平滑，能够满足大多数应用场景。

(2)transform 不真正改变 HTML 元素的位置，适合设计动画的初始状态和结束状态。

(3)transform：rotateX(90deg) translateX(100px)表示同时使用了 rotateX 和 translateX 两个transform 效果。

2.有 name 属性的类名

之前的 CSS 类选择器的名称是在没有为 transition 定义 name 属性情况下的默认的类的名称，开发者可自定义 name 属性的属性值。如果 transition 元素定义了 name 属性，则上述类名就不能再用，需要将上面的类的名称替换为 name 的属性值，如＜transition name＝"tname"＞，则其对应的 6 个类选择器的名称分别为 tname-enter、tname-enter-active、tname-enter-to、tname-leave、tname-leave-active、tname-leave-to。

很多时候，v-enter-to 和 v-leave 会定义为相同的 CSS 样式，v-enter-active 和 v-leave-active 有默认值，在某些场景下可以省略。

CSS 动画用法同 CSS 过渡，区别是在动画中 v-enter 类名在节点插入 DOM 后不会立即删除，而是在 animationend 事件触发时删除。

Vue 会自动地在元素或组件进入和离开时为它们添加定义的类。transition 经常和第三方 CSS 动画库一起使用，如 animate.css，可以提供更加丰富、美观、成熟的动画。

【项目 2-11】

【项目描述】

项目显示效果如图 2-11 所示，点击上面的按钮，下面的盒子即以动画的方式显示与消失，中间的变化过程使用 CSS3 animation 定义，可以对动画变化过程进行自定义。

按钮旁边是控制动画显示的属性的值，便于在调试动画的过程中确定盒子显示或隐藏的状态。

图 2-11　基于 CSS3 animation 的 Vue transition

【项目实现】

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8">
  <title>animation</title>
  <script src="./vue.js"></script>
</head>
<body>
```

```
<style>
  .box{
    border:4px solid #039;
    color:#039;
    padding:20px;
    margin:10px;
    width:300px;
    font-size:40px;
    text-align:center;
  }
  .tscale-enter-active {
    animation: dh .9s;
  }
  .tscale-leave-active {
    animation: dh .9s reverse;
  }
  @keyframes dh {
    0% {
      transform: scale(0)translateX(50px);
    }
    50% {
      transform: scale(.5)translateX(300px);
    }
    100% {
      transform: scale(1);
    }
  }
</style>
<div id="app">
  <button @click="show = ! show">切换显示与隐藏</button>
  {{show}}
  <transition name="tscale">
    <div v-if="show" class="box">animation</div>
  </transition>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      show: true
    }
  })
```