第一篇

前端高效开发框架技术与应用

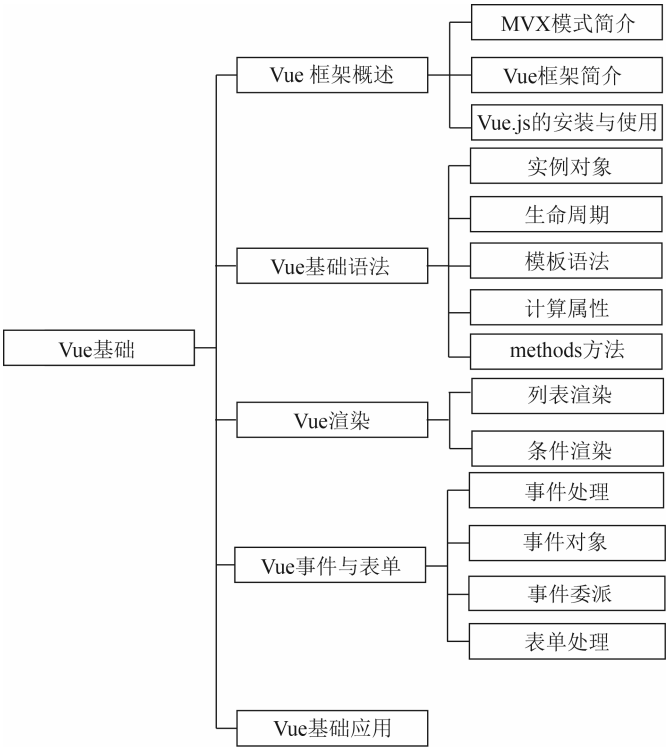
第1章 Vue基础

学习任务

【任务 1】了解 MVX 设计模式及 Vue 的框架和使用。

【任务 2】掌握 Vue 基础语法、Vue 多种渲染方式及 Vue 事件与表单。

学习路线



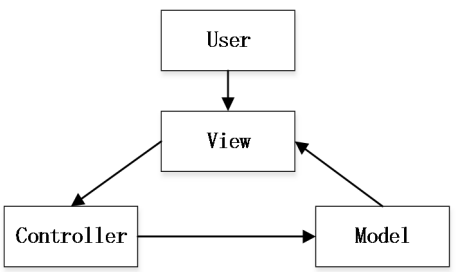
1.1 Vue 框架概述

1.1.1 MVX 模式简介

对于移动互联网公司的前端开发者来说，移动端对前端项目的要求越来越严格，前端项目已经不是简单地通过重新渲染来更新数据的频繁变化，后端的一些 MVC 模式也在往前端框架迁移。在介绍 Vue 框架之前，我们先来了解 MVX 模式。下面我们主要介绍 MVC 模式、MVVM 模式、两者之间的区别，以及 Web 前端开发经历的几个阶段。

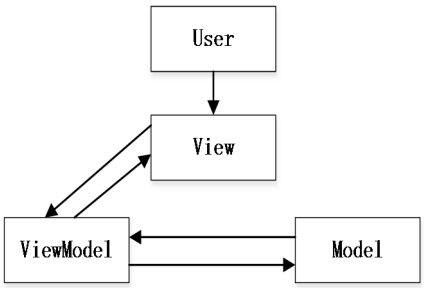
1．MVC 模式

MVC 模式是移动端应用广泛的软件架构之一，MVC 模式将应用程序划分为 3 部分：Model（模型）、View（视图）和 Controller（控制器）。MVC 模式的执行过程是将 View 层展示给用户，也就是通过 HTML 页面接收用户动作，将指令传递给 Controller。如单击一个按钮，就会将按钮触发的业务传递给 Controller；Controller 完成业务逻辑，要求 Model 改变状态。如果业务逻辑是单击按钮时显示列表数据，则 Controller 就处理业务逻辑，要求 Model 改变状态；Model 将新的数据发送给 View，则用户得到反馈。MVC 模式的执行过程如下图所示。



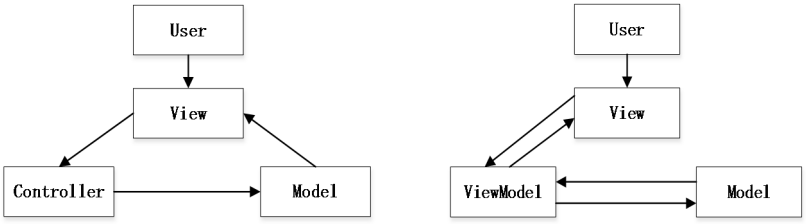
2．MVVM 模式

MVVM 模式是将 MVC 模式的 Controller 改成 ViewModel。View 的变化会自动更新 ViewModel，ViewModel 的变化也会自动同步到 View 层显示。View 层用来接收用户请求（DOM 事件、AJAX 等）；Model 层处理数据，不再与 View 层交互数据；ViewModel 监听 View 层请求状态的变化，同时刷新 View 层显示，ViewModel 和 Model 层之间进行数据双向绑定，Model 层监听 ViewModel 的变化。MVVM 模式的执行过程是 View 层接收到请求告诉 ViewModel，用户需要执行一些处理动作，当 ViewModel 发生变化时，告诉 View 层需要更新页面。所谓的数据双向绑定是 ViewModel 需要更新 Model 层的数据；反之，Model 层的数据改变，在 ViewModel 中的数据状态也要进行相应的改变。MVVM 模式的执行过程如下图所示。



3．MVC 模式和 MVVM 模式的区别

在 MVC 模式中，数据是单向通信的，按照 View→Controller→Model→View 方向循环。在 MVVM 模式中，数据可以双向通信，核心是 ViewModel 对象。这两种模式的数据通信方式如下图所示。



MVC 模式　　MVVM 模式

4．Web 前端开发阶段

- 源代码开发阶段：直接使用 HTML、CSS 和 JavaScript 代码进行前端开发。

优点：学习成本较低，容易入手。

缺点：代码结构混乱，代码冗余，浏览器兼容性不成熟，不利于团队分工合作。

- 代码库开发阶段：使用成熟的开源扩展库进行前端开发（如在 JavaScript 中可以使用 jQuery、在 CSS 样式中可以使用 Bootstrap 响应式框架等）。

优点：开发快速，浏览器兼容性良好。

缺点：视图层和数据层混合在一起，不利于团队分工合作。

- 框架开发阶段：采用前端 MVC 模式或 MVVM 模式开发。

优点：代码分层，利于团队合作，便于后续代码维护。

缺点：学习成本较高，框架更新换代较快。

1.1.2　Vue 框架简介

Vue 是一套构建用户界面的渐进式框架，Vue 只关注视图层，采用自底向上增量开发的设计，Vue 的目标是通过 API 实现数据绑定和组合视图组件。

1．Vue 框架的特性

目前，主流互联网公司及前端开发者喜欢 Vue 框架的原因是它具有自己的特性。

- 数据绑定：对于一些处于交互状态的前端 UI 界面，数据绑定非常简单、方便。
- 指令：拥有内置的简单指令（v-*），用户也可以自定义指令，通过对应表达式值的变化就可以修改对应的 DOM。
- 轻量级：Vue.js 的体积非常小且不依赖其他基础库。
- 插件化：Vue.js 的核心包不包括 Router、AJAX、表单验证等功能，但是可以方便加载对应的插件。

2．Vue 框架的内容

Vue 框架是 Web 开发的核心部分，人们熟练地掌握 Vue 框架的基础知识有助于后续的学习。目前在国内外的公司中，Vue 框架的使用非常广泛。人们在学习 Vue.js 之前需要学习一些网站开发的前导课程，如《Web 前端开发》《JavaScript 程序设计》《响应式设计》等，也需要掌握一些基本技能，如能够使用 HTML、CSS 和 JavaScript 进行静态网站的开发；熟练掌握 Web 开发工具的使用（Chrome 浏览器、Sublime/VS Code 代码编辑器）。Vue 框架的主要内容如下表所示。

模　板	主 要 内 容
Vue 基础知识	Vue 框架基础语法、Vue 开发工具
Vue 组件	Vue 组件、Vue 模块化开发
Vue 工程化工具	npm/yarn 和 webpack 开发工具、Vue CLI 开发工具
Express 服务器开发和 axios 网络请求	创建 Express 应用程序、axios 发送 AJAX 请求、Postman 插件
Vue 路由	单页面应用 SPA、vue-router 实现路由机制
Vuex 状态管理	本地存储
Vue UI 库	Vue 中的 UI 库

3．主流的 MVVM 框架

目前，主流的 MVVM 框架有 Angular、React 和 Vue.js，下面分别介绍这 3 个主流框架的区别。

- Angular：谷歌推出的 MVVM 框架。

优点：功能强大，包含模板、数据双向绑定、路由、模块化、服务等功能，自带了丰富的 Angular 指令。AngularJS 由谷歌维护，社区非常活跃，能够很好地促进它的发展。AngularJS 和 Ionic 是移动端跨平台 App 开发的黄金搭档。

缺点：功能复杂。

- React：Facebook 推出的 JavaScript 库。

优点：学习简单，视图层使用 JSX 语法，增强视图层功能，以 JavaScript 为核心，其规范标准，跨平台移动 App 开发能力强大，可以快速开发移动端应用。

缺点：操作 DOM 能力较差，需要大量扩展的支持。

- Vue.js：尤雨溪于 2014 年开发的 MVVM 框架。

优点：拥有非常强大的 CLI 工具，方便应用快速部署和模块化开发，扩展模块丰富，扩展功能强大，与 Laravel 集成，Laravel 是服务器端的一个 PHP 框架，适用于服务器端 API 开发。

缺点：跨平台移动 App 开发能力薄弱。

4．主流 MVVM 框架的应用场景

MVVM 三大框架主要应用场景如下表所示。

	Angular	React	Vue.js
发布时间	2009 年	2013 年	2014 年
维护者	谷歌	Facebook	尤雨溪
视图引擎	HTML	JSX	HTML
学习曲线	困难	较复杂	简单
大小	500KB	130KB	50KB
功能复杂度	完善	完善	简单
文档	英文	英文	中文
应用场景	大型复杂应用	大中型应用、移动跨平台开发	中小型轻量级应用

1.1.3 Vue.js 的安装与使用

1．Vue.js 的安装

- 在 Vue.js 的官网上直接下载 vue.min.js 并用<script>标签引入。

Vue.js 的下载地址为 https://vuejs.org/js/vue.js。

Vue 官方安装说明网址为 https://cn.vuejs.org/v2/guide/installation.html。

打开下载地址，按“Ctrl+A”组合键选中网页内容并进行复制，打开 VS Code 软件，新建 Vue.js 文件，然后粘贴复制的文件即可。这样我们就准备好了 Vue.js 文件，当使用时直接在应用程序中用<script>标签引入 Vue.js 文件即可。

- 利用 CDN 方式引入 Vue.js 文件。

CDN 是内容分发系统，它的主要作用是将网站的内容发布到最接近用户的网络“边缘”，使用户可以就近取得所需的内容，提高用户访问网站的响应速度。

Vue 推荐使用的网址为 https://unpkg.com/vue。

Staticfile CDN（国内）推荐使用的网址为 https://cdn.staticfile.org/vue/2.2.2/vue.min.js。

unpkg 推荐使用的网址为 https://unpkg.com/vue/dist/vue.js。

cdnjs 推荐使用的网址为 https://cdnjs.cloudflare.com/ajax/libs/vue/2.1.8/vue.min.js。

可以通过如下代码进行引用：

```
<script src="https://unpkg.com/vue"></script>
```

- 采用 npm 和 webpack 模块包形式引入 Vue.js 文件。

npm 是现代前端开发的主要工具，将复杂的 JavaScript 文件库组织成模块化管理方式，在使用时只需引入指定模块即可。

webpack 将模块化编码的程序进行打包生成，从而允许在不支持 ES6 的浏览器中访问。

2．第一个 Vue 实例

根据上面的介绍，我们创建一个 Vue 实例来更好地理解程序，后续的程序可以在这个实例的基础上进行修改。Vue 实例主要包括 HTML 代码、CSS 代码和编写 JavaScript-Vue 代码。创建第一个 Vue 实例的主要步骤如下。

（1）引入 CDN 文件。

（2）创建 Vue 实例对象。

（3）在 HTML 文件中添加关联 Vue 实例对象。

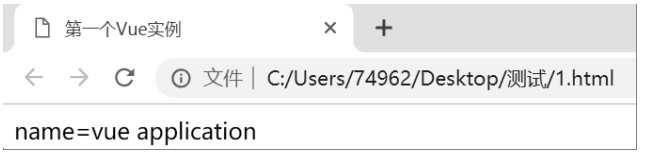
在 Vue 实例对象中，data 属性表示用户需要为 Vue 实例对象传递一些数据。Vue 实例对象的参数需要放在双大括号之间（{{…}}）。

下面的实例，通过<script>标签引入官方推荐的 CDN 文件。首先创建一个 Vue 实例对象，Vue 实例对象通过 el 元素和 html 元素相关联，绑定 id 为 app 的 div 元素，也就是只有 id 为 app 的 div 元素才可以响应 Vue 中的数据。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>第一个 Vue 实例</title>
  <script type="text/javascript"></script>
</head>
<body>
  <div id="app">
    name={{name}}
   </div>
   <script src="https://unpkg.com/vue"></script>
   <script>
    //创建一个 Vue 实例对象
    var vm = new Vue({
       el:'#app',//el 绑定的 id 为 app 的 div 元素
       data:{
          name:'vue application'
        }
      });
```

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

运行结果如下图所示。



3．Vue 框架开发工具

对于 Vue 框架的开发主要用到开发文档、浏览器及代码编写工具等，下面分别介绍 Vue 框架开发工具。

- 开发文档。

Vue 官方文档的下载地址为 https://cn.vuejs.org/v2/guide。

npm 官方文档的下载地址为 https://www.kancloud.cn/shellway/npm-doc/199982。

webpack 官方文档的下载地址为 https://doc.webpack-china.org/concepts/。

- 浏览器。

Chrome 浏览器。

vue-devtools 调试工具。

- 代码编写工具。

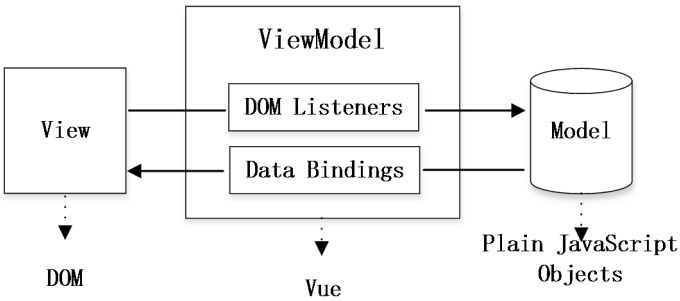
Atom、Sublime Text、Visual Studio Code 等。

1.2 Vue 基础语法

1.2.1 实例对象

Vue 实例对象是 Vue 框架的核心，人们需要掌握 Vue 实例对象的基本使用方法，同时也要深入理解 Vue 实例对象，包括数据的双向绑定机制、实例对象的访问方法与设置方法等。

什么是 Vue 实例对象？Vue 框架的核心是 Vue 实例对象，即 ViewModel 对象。Vue 实例对象连接 View（视图）层和 Model（模型）层，核心就是 ViewModel 对象，在 ViewModel 对象中关联 View 层和 Model 层，其中 View 层执行一个数据的双向绑定，当 View 层触发动作以后，告诉 ViewModel 对象的 DOM Listeners（事件监听机制），从而更新 Model 层中的数据；当 Model 层中的数据发生改变之后，交给数据双向绑定机制，双向绑定机制告诉 View 层要刷新数据。Vue 实例对象如下图所示。



在 Vue 框架中，使用 ViewModel 的方法非常简单，每个 Vue 应用程序都是通过创建一个新的 Vue 实例开始的。在创建 Vue 实例对象时，可以传入一系列实例选项，例如，data、props、methods、computed 等选项。创建 Vue 实例对象的语法格式为：

```
var vm = new Vue({  });
```

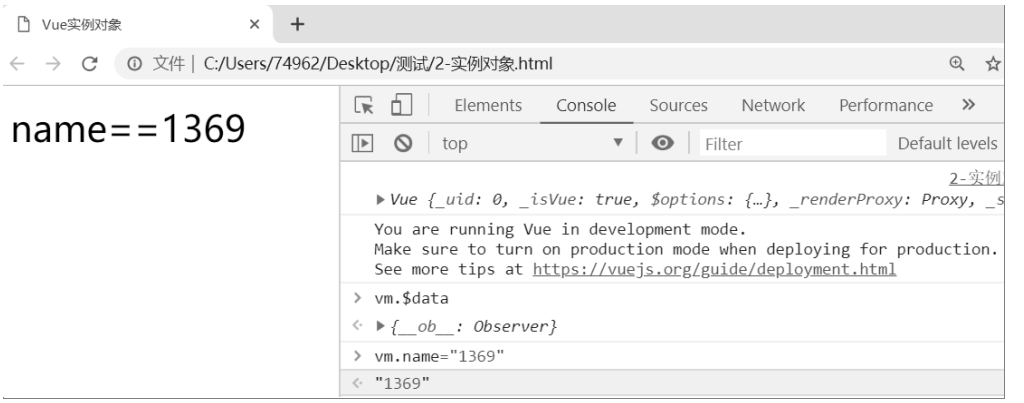
其中，data 属性是向 Vue 实例对象的响应式系统中添加双向绑定的数据，当 Vue 实例对象的 data 属性发生改变时，在视图中的数据也会相应发生改变；当视图中绑定的 data 数据改变时，在 Vue 实例对象 data 属性中的相应数据也会发生改变。在创建 Vue 实例对象之后，可以通过 vm.$data 访问原始数据对象，Vue 实例对象代理了 data 所有的属性，因此访问 vm.a 等价于访问 vm.$data.a。

下面的实例，通过 console.log(vm)可以在控制台打印出 Vue 实例对象的所有属性，在浏览器中显示的是 data 中 name 的属性值'你好'，通过控制台输入 vm.name="1369"可以改变浏览器的显示值，也可以通过 vm.$data 回车显示原始数据对象。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Vue 实例对象</title>
  <script type="text/javascript"></script>
</head>
<body>
   <div id="app">
      name=={{name}}
   </div>
   <script src="https://unpkg.com/vue"></script>
   <script>
      //创建一个 Vue 实例对象
      var vm = new Vue({
        //绑定的 DOM 元素
        el:'#app',
        //双向绑定的数据 VM 对象可以映射到 Model 中，Model 改变以后也会影响 VM 数据
        //VM 可以改变 View 中的数据，View 中的数据又会影响 VM 中的数据
        //双向绑定的数据都是键值对的形式
        data:{              //通过 data 属性可以向双向绑定机制中传入数据
```

```
            name:'你好'
        }
    });
        console.log(vm); //包含一系列属性
    </script>
</body>
</html>
```

运行结果如下图所示。



1.2.2 生命周期

Vue 生命周期是 Vue 实例对象衍生出的一个机制，生命周期其实就是 Vue 实例对象在创建过程中所实现的回调函数，我们可以在回调函数中编写代码，从而实现一些特定的功能。

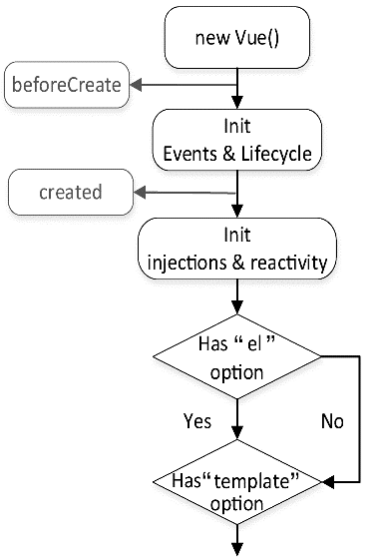
每个 Vue 实例对象在被创建之前都要经过一系列初始化过程，同时在这个过程中也会运行一些生命周期的回调函数，给予用户自定义启动过程的权利，具体的回调函数说明如下表所示。

回 调 函 数	说　明
beforeCreate()	Vue 实例对象创建之前
created()	Vue 实例对象创建之后
beforeMount()	Vue 实例对象和文档节点挂载之前
mounted()	Vue 实例对象和文档节点挂载之后
beforeUpdate()	View 视图更新之前
updated()	View 视图更新之后
beforeDestroy()	Vue 实例对象销毁之前
destroyed()	Vue 实例对象销毁之后

对于上文生命周期中的回调函数，我们要明白这些函数的使用方法、意义及使用场景。根据表格中的回调函数，大致可以分为 4 类，下面分别进行讲解。

1．beforeCreate()和 created()

创建 Vue 实例对象会经历两个阶段：第一个阶段是初始化事件绑定机制、初始化生命周期的循环，初始化之后会触发 beforeCreate()回调函数；第二个阶段是初始化注入器、初始化实体对象，此时表示 Vue 实例对象已经创建成功，初始化之后会触发 created()回调函数，如下图所示。这些回调函数使用起来很简单，直接在 Vue 实例对象构造参数中给出即可。



beforeCreate()：Vue 实例对象创建之前的回调，此时 el 属性和 data 属性均为空。

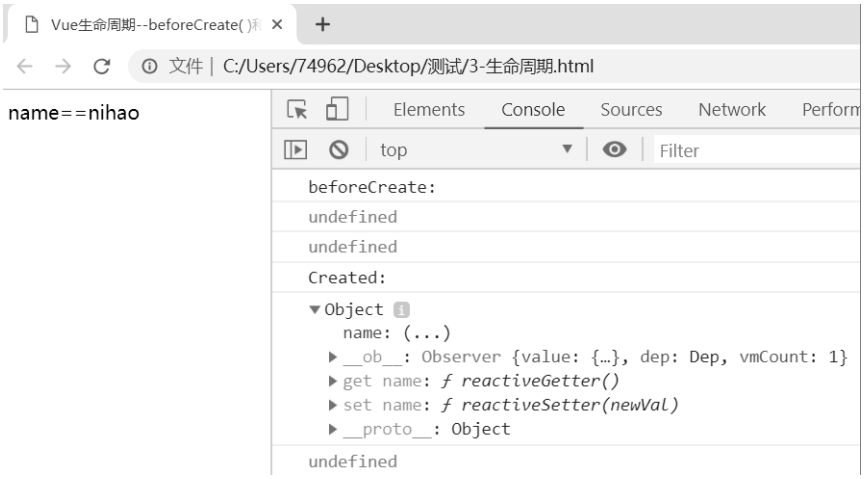
created()：Vue 实例对象创建之后的回调，此时 el 属性为空，但 data 属性已经存在。

下面的实例，在 Vue 实例对象创建之前，查看 data 属性和 el 属性是否存在且 data 属性和 el 属性分别为空。在 Vue 实例对象创建之后，查看 data 属性和 el 属性是否存在，此时 data 属性已经存在，el 属性为空，说明此时还没有对 Vue 实例对象进行 DOM 绑定。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Vue 生命周期--beforeCreate()和 created()</title>
  <script type="text/javascript"></script>
</head>
<body>
   <div id="app">
       name=={{name}}
   </div>
   <script src="https://unpkg.com/vue"></script>
   <script>
```

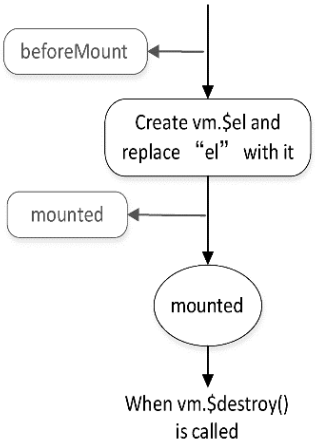
```
        var vm = new Vue({
            el:'#app',
            data:{
                name:'nihao'
            },
            //生命周期回调函数
            beforeCreate:function(){
                //Vue 实例对象创建之前的回调
                console.log('beforeCreate:');
                console.log(this.$data);
                console.log(this.$el);
            },
            created:function(){
                //Vue 实例对象创建之后的回调
                console.log('Created:');
                console.log(this.$data);
                console.log(this.$el);
            }
        });
    </script>
</body>
</html>
```

运行结果如下图所示。



2．beforeMount()和 mounted()

在创建 el 属性之前首先执行 beforeMount()回调函数，然后创建 el 属性并与 DOM 进行绑定之后才会触发 mounted()回调函数，如下图所示。



beforeMount()：View 和 Model 绑定完成之前的回调，存在于 Vue 实例对象和文档节点挂载之前，此时 el 属性为绑定之前的值。

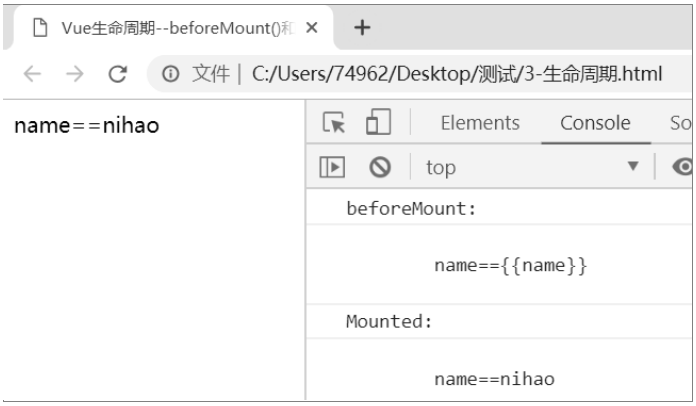
mounted()：View 和 Model 绑定完成之后的回调，存在于 Vue 实例对象和文档节点挂载之后，此时 el 属性为绑定之后的值。

下面的实例，绑定之前的 name 值是原始值，绑定之后的 name 值是替换数据以后的 HTML。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Vue 生命周期--beforeMount()和 mounted()</title>
  <script type="text/javascript"></script>
</head>
<body>
   <div id="app">
      name=={{name}}
   </div>
   <script src="https://unpkg.com/vue"></script>
   <script>
     var vm = new Vue({
         el:'#app',
         data:{
             name:'nihao'
         },
         //生命周期回调函数
         beforeMount:function(){
             console.log('beforeMount:');
             console.log(this.$el.innerHTML);
         },
         mounted:function(){
             console.log('Mounted:');
```

```
            console.log(this.$el.innerHTML);
        }
    });
  </script>
</body>
</html>
```

运行结果如下图所示。



3．beforeUpdate()和 updated()

beforeUpdate()：View 视图更新之前的回调，此时 el 属性为更新之前的值。

updated()：View 视图更新之后的回调，此时 el 属性为更新之后的值。

4．beforeDestroy()和 destroyed()

beforeDestroy()：Vue 实例对象销毁之前的回调，此时 el 属性和 data 属性仍然具有原始值。

destroyed()：Vue 实例对象销毁之后的回调，此时 el 属性和 data 属性仍然具有原始值，但是后续再修改 Model 将不会改变 View。

1.2.3 模板语法

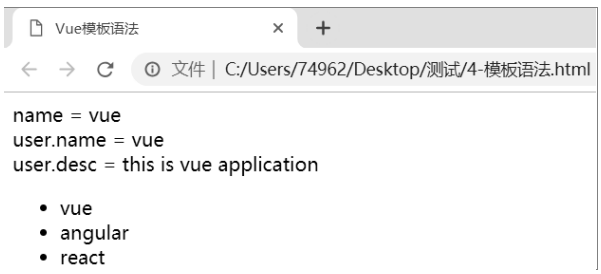
所谓模板语法主要是如何在 HTML 中输出 Vue 实例对象的数据，具体而言，我们在 HTML 元素中输出 Vue 实例对象的变量，或者在 HTML 属性中输出 Vue 实例对象的变量，在一般情况下，我们需要做一些计算，针对这些计算，可以输出一些 JavaScript 表达式。

- 输出模板变量：使用“Mustache”语法（双大括号）的文本插值形式在 HTML 中输出模板变量，模板变量包括普通变量、对象数据、数组数据等。下面的实例，分别准备了双向绑定的数据，包括普通字符串、对象数据、数组数据等，在浏览器中可以通过双大括号的形式将这些变量数据显示出来。示例代码如下：

```
<!DOCTYPE html>
<html>
```

```
<head>
  <meta charset="UTF-8">
  <title>Vue 模板语法</title>
  <script type="text/javascript"></script>
</head>
<body>
  <div id="app">
    <div>name = {{name}}</div>
    <div>
        user.name = {{user.name}}<br>
        user.desc = {{user.desc}}
    </div>
    <ul>
        <li>{{names[0]}}</li>
        <li>{{names[1]}}</li>
        <li>{{names[2]}}</li>
    </ul>
  </div>
   <script src="https://unpkg.com/vue"></script>
   <script>
     //准备双向绑定数据
      var data={
        name:'vue',                          //普通字符串
        user:{                               //对象
            name:'vue',
            desc:'this is vue application'
        },
        names:['vue','angular','react'],    //数组
    }
    var vm = new Vue({
        el:'#app',
        data:data
    });
   </script>
</body>
</html>
```

运行结果如下图所示。



- 在 HTML 元素中输出 Vue 实例对象的变量：使用 v-html 指令用于输出 HTML 代码，若数据变量为 HTML 源代码，默认输出 HTML 源代码；若解析 HTML 代码输出，可以使用 v-html 指令。下面的实例，数据变量是 HTML 源代码，第一个 div 直接通过双大括号的形式输出，可以将 HTML 代码完整输出；第二个 div 通过 v-html 指令的形式输出，可以解析 HTML 代码，将字体设置为红色。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Vue 模板语法</title>
    <script type="text/javascript"></script>
</head>
<body>
    <div id="app">
        <div>
            {{html}}
        </div>
        <div v-html="html"></div>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        //准备双向绑定数据
        var data={
            html:'<font color="red">这是红色字体</font>'
        }
        var vm = new Vue({
            el:'#app',
            data:data
        });
    </script>
</body>
</html>
```

运行结果如下图所示。

- 在 HTML 属性中输出 Vue 实例对象的变量：此时在某些情况下，需要在 HTML 元素属性中输出变量，此时使用 v-bind 指令。具体的语法形式是 v-bind:属性名，可以简写成：":属性名"形式；v-bind 表示在属性中输出模板变量。下面的实例，在属性中输出超链接的地址，第一行是 v-bind 指令的全写形式，第二行是 v-bind 指令的简

写形式，当单击超链接时就可以跳转到超链接的页面。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Vue 模板语法</title>
    <script type="text/javascript"></script>
</head>
<body>
    <div id="app">
        <div>
            <a v-bind:href="link.link">全写：{{link.name}}</a><br/>
            <a :href="link.link">简写：{{link.name}}</a>
        </div>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        var data={
            link:{
                name:'Vue 框架开发',
                link:'https://cn.vuejs.org/v2/guide'
            }
        }
        var vm = new Vue({
            el:'#app',
            data:data
        });
    </script>
</body>
</html>
```

运行结果如下图所示。

需要注意的是若动态属性的变量值为 false、null 或 undefined，则该属性不存在。

```
<a v-bind:href="link.link" :class="myclass">
    {{link.name}}
</a>
var data={
    myclass:false,
    link:{
        name:'Vue 框架开发',
```

```
            link:'https://cn.vuejs.org/v2/guide'
        }
    }
```

- 表达式：Vue 框架提供了 JavaScript 表达式、四则运算、条件控制、字符串处理，同样在 v-bind 绑定中也可以进行一些计算，对于所有的数据绑定（无论 HTML 元素还是 HTML 属性），Vue.js 提供了完整的 JavaScript 表达式。

下面的实例，首先进行加法运算；其次进行条件控制；再次进行字符串处理，对其进行逆序输出；最后在 v-bind 绑定中进行一些计算。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Vue 模板语法</title>
    <script type="text/javascript"></script>
</head>
<body>
    <div id="app">
        {{5+5}}<br/>
        {{ ok ? 'YES':'NO' }}<br/>
        {{ message.split('').reverse().join('') }}
        <div v-bind:id="'list-' + id">百度地图</div>
    </div>
    <script src="https://unpkg.com/vue"></script>
    <script>
        var data={
            ok: true,
             message: '床前明月光',
            id:1
        }
        var vm = new Vue({
            el:'#app',
            data:data
        });
    </script>
</body>
</html>
```

运行结果如下图所示。

1.2.4　计算属性

为什么使用计算属性？在模板中可以使用 JavaScript 表达式做一些简单运算，设计它的初衷是用于简单运算，但在模板中放入太多的处理逻辑会让模板过重且难以维护。

例如，在外卖系统中，怎样计算用户订单金额？订单金额是由商品个数（count）、商品单价（price）、折扣率（discount）、运费（freight）等信息构成的。若直接使用 JavaScript 表达式，视图部分会变得越来越复杂，不符合 Vue 实例对象简洁的设计原则。订单金额={{ count* price * discount+freight }}。

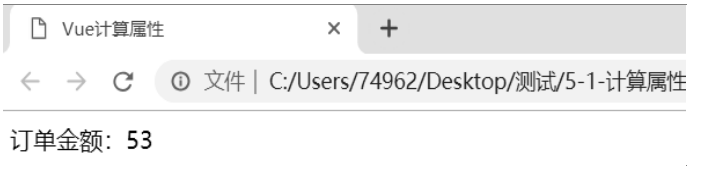
这个例子比较简单，但是在实际生产环境中，这样的计算方式很复杂，针对这种情况，我们应该使用计算属性。所谓计算属性是一种特殊的属性，只不过这些属性允许计算，允许一些函数操作，如使用 computed 属性定义计算属性。在使用计算属性前先定义计算属性，在 Vue 实例对象中，使用 computed 属性定义计算属性，使用计算属性就是在模板中直接使用已经定义好的计算属性名称即可。

下面的实例，使用计算属性计算订单金额，首先在创建 Vue 实例对象时定义计算属性，其次每一个计算属性值是函数形式，并且必须有返回值，返回值是计算的订单金额。示例代码如下：

```
<!DOCTYPE html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Vue 计算属性</title>
  <script type="text/javascript"></script>
</head>
<body>
    <div id="app">
        <div>
            订单金额：{{amount}}
        </div>
    </div>
   <script src="https://unpkg.com/vue"></script>
   <script>
    //准备双向绑定数据
```

```
    var data={
        count:5,        //商品个数
        price:12,       //商品单价
        discount:0.8,   //折扣率
        freight:5       //运费
    };
    var vm = new Vue({
        el:'#app',
        data:data,
        //定义计算属性
        computed:{
            //每一个计算属性值是函数形式，并且必须有返回值
            amount:function(){
                //this 表示当前 Vue 实例对象
                //this.$data 表示当前 Vue 实例对象装载的双向绑定数据
                //代理：this.$data.count<=>this.count
                return this.$data.count*this.$data.price*
                this.$data.discount+this.$data.freight;
            }
        }
    });
  </script>
</body>
</html>
```

运行结果如下图所示。



关于计算属性需要注意：

- computed 属性定义由一系列 JSON 方法组成，表示一系列计算属性。
- 每一个计算属性是一个 function，并定义了一个函数，这个函数必须由 return 语句返回计算属性的返回值。

通过观察发现，计算属性的使用和普通的 data 属性一致，计算属性就是把函数的形式转化成属性的使用方法，从而使用在模板中。改变计算属性的参数，其数字也会有变化，可以看出计算属性和 data 属性的操作一致，只不过是定义的形式不同。

当原始数据发生改变时，计算属性值也会发生改变；在计算属性中也可以使用 HTML 代码，需要注意在模板中要使用 v-html 属性形式。示例代码如下：

```
<div>
    订单金额：<span v-html="amount"></span>
</div>
```