

项目一 Vue.js简介

学习目标

- 了解Vue是什么及Vue的特点
 - 了解Vue在前端开发中的优势
 - 掌握Vue的下载以及如何引入并应用
 - 掌握实例化Vue对象、数据和方法
 - 掌握数据挂载到DOM页面
 - 掌握Vue中的MVVM模式
-

本章内容

- 1.1 Vue.js概述
 - 1.2 Vue安装
 - 1.3 实例化Vue对象、数据和方法
 - 1.4 MVVM模式
-

1.1 Vue.js概述

- 1.1.1 什么是Vue.js
 - 1.1.2 为什么使用Vue.js
 - 1.1.3 Vue.js的主要特点
 - 1.1.4 Vue.js的优势
 - 1.1.5 Vue.js有什么不同
-

1.1.1 什么是Vue.js

- ❑ Vue.js 是一套响应式系统，前端开发库
 - ❑ Vue.js 是一套构建用户界面的渐进式框架。采用自底向上增量开发的设计
 - ❑ Vue 的核心库只关注视图层
 - ❑ Vue 采用单文件组件和生态系统支持的库开发的复杂单页应用
 - ❑ 提供了 MVVM 数据绑定和一个可组合的组件系统
-

1.1.2 为什么使用Vue.js

- Vue 核心库只关注视图层
 - 解决网页结构之间存在依赖或依存关系，不需要数据和视图全混合在一起
 - 避免使用jQuery 选择器及 DOM 操作本身存在性能缺失
 - 不需要不断地一层层向上寻找父辈元素
-

1.1.3 Vue.js 的主要特点

- 轻量级的框架
 - 双向数据绑定
 - 指令
 - 组件化
 - 客户端路由
 - 状态管理
-

1.1.4 Vue.js 的优势

- 可进行组件化开发，使代码量减少
- 可对数据进行双向绑定
- 编写出来界面效果本身是响应式的，使网页能显示好看效果
- 实现网络页面之间跳转，与Vue 路由相比，超链接不会页面刷新

传统的页面是通过超链接来实现页面的切换和跳转的

1.1.5 Vue.js 有什么不同

- 在前端开发中，会遇到动画、交互效果、页面特效等业务，原生的JavaScript或jQuery库通过操作DOM来实现，数据和界面是连接在一起的，例如下面的示例。

1.1.5 Vue.js 有什么不同

□ 例1.1 JQuery操作DOM示例

```
<div>
  <p>担任的课程名称是<span id="course">HTML</span>课程。 </p>
  <button id="updata">修改</button>
</div>
<script>
  $("#updata").click(function () {
    $("#course").text("VueJs");
  });
</script>
```

1.1.5 Vue.js 有什么不同

□ 例1.2 Vue操作DOM示例

```
<div id="app">
  <p>这门课程名称是<span>{{course}}</span>课程。 </p>
  <button v-on:click="update">修改</button>
</div>
<script>
  new Vue({
    el: '#app',
    data:{
      course:"HTML"
    },
    methods:{
      update:function(){
        this.course = "Vue.js工程化项目开发";
      }
    }
  })
</script>
```

1.1.5 Vue.js 有什么不同

总结以上示例：

- ❑ jQuery首先要获取DOM对象，然后对DOM对象进行值的修改操作
 - ❑ Vue首先把值和Javascript对象进行绑定，然后修改JS对象的值，Vue框架就会自动把DOM的值进行更新
 - ❑ Vue为我们做了DOM操作，以后使用Vue只需修改对象的值以及做好元素和对象的绑定，Vue框架就会自动做好DOM的相关操作
 - ❑ DOM元素跟随JS对象值的变化而变化，叫作**单向数据绑定**；如果JS对象的值也跟随着DOM元素的值的变化而变化，叫作**双向数据绑定**
-

回顾知识

- jQuery首先要获取DOM对象，然后对DOM对象进行值的修改操作
 - Vue首先把值和Javascript对象进行绑定，然后修改JS对象的值，Vue框架就会自动把DOM的值进行更新
 - Vue为我们做了DOM操作，以后使用Vue只需修改对象的值以及做好元素和对象的绑定，Vue框架就会自动做好DOM的相关操作
 - DOM元素跟随JS对象值的变化而变化，叫作**单向数据绑定**；如果JS对象的值也跟随着DOM元素的值的变化而变化，叫作**双向数据绑定**
-

1.2 Vue的安装

- 1.2.1 直接用<script>引入
 - 1.2.2 NPM
 - 1.2.3 命令行工具(CLI)
-

1.2.1 直接用<script>引入

- 直接使用<script>标签引入有两种方式，一种是官网下载独立的版本，另一种是使用CDN的方式。在制作项目时，要使用Vue的脚手架进行创建。
 - **1.独立的版本**
 - **2.CDN方式**
-

1.2.1 直接用<script>引入

- 1.独立的版本
- 进入官网<https://cn.vuejs.org/>进行下载，并用<script>引入



The screenshot shows the Vue.js Chinese website. At the top, there is a navigation bar with a search bar and links for '学习' (Learn), '生态系统' (Ecosystem), '团队' (Team), '资源列表' (Resource List), '支持 Vue' (Support Vue), '多语言' (Multi-language), and '参与翻译' (Participate in Translation). The main content area features the Vue.js logo (a large green 'V' with a dark blue center) and the text '渐进式 JavaScript 框架' (Progressive JavaScript Framework). Below this, there are three buttons: 'WHY VUE.JS?' (with a play icon), '起步' (Getting Started), and 'GITHUB'. A section titled '特别赞助' (Special Sponsor) highlights the 'xinguan2020 抗疫项目' (Xinguan2020 Anti-Epidemic Project) with the slogan '病毒无情，开发者有情' (The virus is ruthless, but developers have a heart) and a link to '成为特别赞助商' (Become a special sponsor). At the bottom, three key features are listed: '易用' (Easy to use), '灵活' (Flexible), and '高效' (Efficient). Each feature has a brief description: '易用' (Already familiar with HTML, CSS, JavaScript? Start building applications immediately by reading the guide!), '灵活' (A constantly growing ecosystem, allowing for seamless integration between libraries and a complete framework), and '高效' (20kB min+gzip runtime size, ultra-fast virtual DOM, and the most optimized).

1.2.1 直接用<script>引入

□ 2.使用CDN方法

- 对于制作原型或学习，可以这样使用最新版本：

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

- 对于生产环境，推荐链接到一个明确的版本号 and 构建文件，以避免新版本造成的不可预期的破坏：

```
<script src="https://cdn.jsdelivr.net/npm/vue@2.6.11" > </script>
```

1.2.2 NPM

- 在用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。同时Vue也提供了配套工具来开发单文件组件。
- 由于NPM安装速度慢，推荐使用淘宝NPM镜像CNPM。
 - `$ npm install -g cnpm --registry=https://registry.npm.taobao.org`

如果电脑中 npm 命令不是内部命令，需要先安装 node.js。

- # 最新稳定版
 - `$ npm install vue => $ cnpm install vue`
 - 提示：对于中国用户，建议将NPM源设置为国内的淘宝NPM镜像，可以大幅提升安装速度。
-

1.2.3 命令行工具(CLI)

- Vue提供了一个官方的脚手架（CLI），为单页面应用(SPA)快速搭建繁杂的脚手架。它为现代前端 workflow 提供了batteries-included的构建设置。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。
-

1.2.3 命令行工具(CLI)

■ 安装步骤:

- 查看 npm 的版本号并安装 Vue
 - `npm v => cnpm install vue`
 - 安装脚手架 vue-cli `cnpm install vue-cli -g`
 - 创建一个基于 webpack 模板的新项目
 - `vue init webpack my-project`
 - 使用 cd 命令进入项目 my-project 中，并安装项目依赖项 `cd my-project => cnpm install`
 - 使用 `cnpm run dev` 运行项目
 - 成功执行以上命令后访问：
-

1.2.3 命令行工具(CLI)

- 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。如果是新手，建议先在在熟悉Vue本身之后再使用CLI。后面课程再介绍，脚手架的安装以及如何快速创建一个项目。
-

1.3 实例化 Vue 对象、数据和方法

- 1.3.1 实例化对象
- 1.3.2 数据和方法
- 1.3.3 将数据挂载到DOM页面

1.3.1 实例化 Vue 对象

- 通过构造函数 `Vue ()` 创建一个 Vue 的根实例，每一个 `new Vue ()` 都是一个 Vue 构造函数实例

```
var vm = new Vue({  
  // (选项) 这里编写我的代码。  
})
```

- Vue 构造器要求实例化时需传入选项对象

选项对象包括挂载元素 (`el`)、数据 (`data`)、方法 (`methods`)、模板 (`template`)、生命周期钩子函数等选项。

例1.3 创建第一个Vue 实例

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>实例化 Vue 对象</title>
</head>
<body>
  <!-- app 是根容器-->
  <div id="app"> <div>{{ name }}</div> </div>
  <script src="js/Vue.js" ></script>
  <script>
    new Vue({
      el: '#app',
      data: {
        name: '欢迎学习Vue.js!',
      }
    })
  </script>
</body>
</html>
```

运行结果：欢迎学习Vue.js！

//实例化 vue 对象
// 每个 Vue.js 应用都通过构造函数
Vue 创建一个 Vue 根实例启动的
// 实例化 Vue 时，需传入选项对象

el:element 需要获取的元素，一定是 html 中的根容器元素
data: 用于数据的存储

el:挂载点

el: 是用来设置Vue实例挂载（管理）的元素

```
<div id="app">
  {{message}}
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"hello,world"
  }
})
```

el:挂载点

```
<div id="app">  
  hello,world  
</div>
```

```
var app = new Vue({  
  el:"#app",  
  data:{  
    message:"hello,world"  
  }  
})
```

el:挂载点

□ Vue实例的作用范围是什么呢?

- Vue会管理el选项命中的元素及其内部的后代元素

□ 是否可以使用其他的选择器?

- 可以使用其他的选择器,但是建议使用ID选择器

□ 是否可以设置其他的dom元素呢?

- 可以使用其他的双标签,不能使用HTML和BODY
-

1.3.2 Vue 数据和方法

□ 数据 (data)

- ◆ **data** 用于定义属性，实例中有三个属性分别为：site、url、alex。

□ 属性方法 (methods)

- ◆ **methods** 用于定义的函数，可以通过 return 来返回函数值。

□ `{{ }}` 用于输出对象属性和函数返回值。

数据 (data)

- Vue中用到的数据定义在data中
 - data中可以写复杂类型的数据
 - 渲染复杂类型数据时,遵守js的语法即可
-

数据 (data)

```
<div id="app">
  {{message}}
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"hello,world"
  }
})
```

数据 (data)

```
<div id="app">
  {{message}}
</div>
```

```
var app = new Vue({
  el:"#app",
  data:{
    message:"hello,world",
    array:[],
    obj:{},
  }
})
```

1.3.2 Vue 数据和方法

例1.4 vueData示例

```
<div id="app">
  <h1>{{ say() }}</h1>
  <h1>学校位于{{ address }}</h1>
  <h1>邮编是{{ post }}</h1>
</div>
<script>
  new Vue({
    el: '#app',
    data: {
      school: 'scetop',
      address: '四川省成都市高新区西区大道2000号',
      post: '611743'
    },
    methods: {
      say: function () {
        return "欢迎来到:" + this.school
      }
    }
  })
</script>
```


1.3.3 将数据挂载到 DOM 页面

例1.5 将数据挂载到 DOM

```
<body>
<script src="https://unpkg.com/vue/dist/Vue.js"></script>
<div id="app-2">
  <p>{{ message }}</p>
<button onclick="app.message = '欢迎你！ 未来的工程师。';">更新！ </button>
</div>
<script>
var app = new Vue({
  el: '#app-2',
  data: { message: 'Hello Vue.js!' },
  created: function () { console.log('message is: ' + this.message); },
  mounted: function() { console.log("已挂载到 DOM 页面上。"); },
  updated: function () { console.log("已更新 DOM! "); }
})
</script>
</body>
```

'this' 指向 vm 实例

编写第一个vue.js程序的步骤

- 导入开发版本的Vue.js
 - 创建Vue实例对象, 设置el属性和data属性
 - 使用简洁的模板语法把数据渲染到页面上
-

1.4 MVVM模式

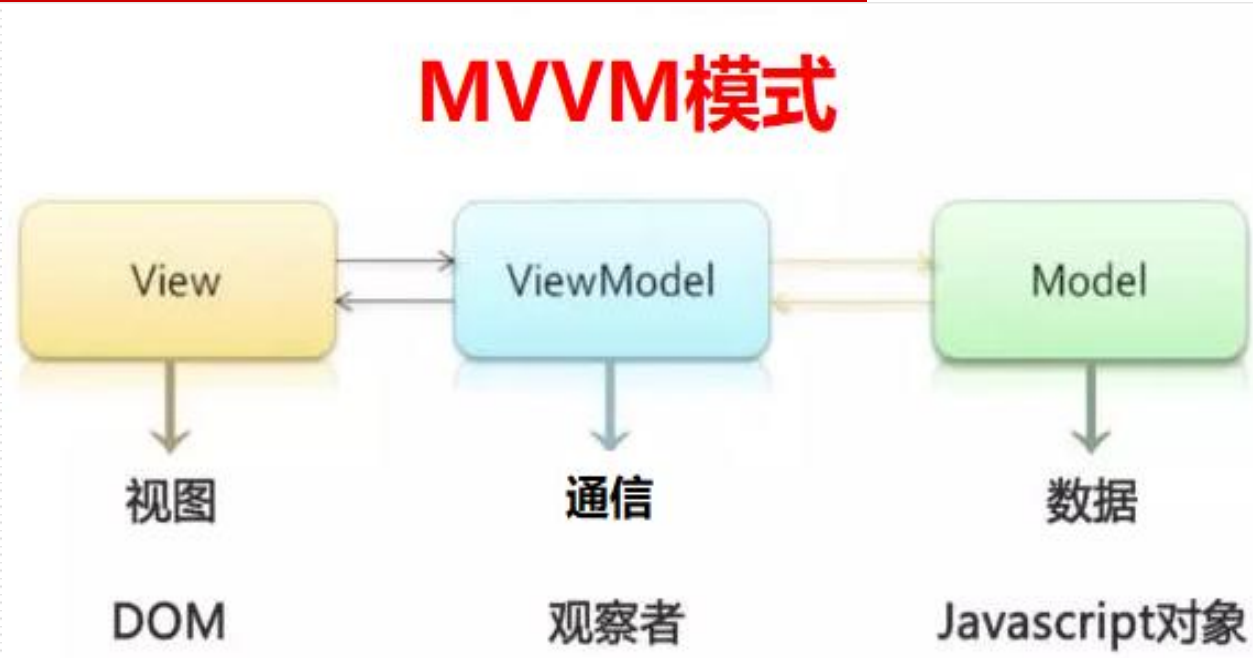
- MVVM是一种基于前端开发的架构模式
 - 其核心是提供对 View 和 ViewModel 的双向数据绑定，这使得一方更新时可自动传递到另一方，即数据双向绑定。
 - ViewModel 负责连接 View 和 Model，保证视图和数据的一致性
-

1.4 MVVM模式

□ MVVM模式 (Model-View-ViewModel) ， 即模型-视图-视图模型。

- Model:是后端传递的数据，负责数据存储
 - View:指的是HTML页面，负责页面展示
 - ViewModel: MVVM模式的核心，它是连接View和Model的桥梁，负责业务逻辑处理（比如Ajax请求等），对数据进行加工后交给视图展示。
-

1.4 MVVM模式



Vue.js 是一个提供了 MVVM 风格的双向数据绑定的 Javascript 库，专注于 View 层。它的核心是 MVVM 中的 ViewModel。

1.4 MVVM模式

例1.6 演示MVVM模式

```
<div id="App">
  {{ property }}
  <input type="text" v-model="property"/>
</div>
<script>
  var exampleData = {
    property: 'Hello World'
  }
  new Vue ({
    el: '#App',
    data: exampleData
  })
</script>
```

1.4 MVVM模式

代码分析

- ❑ 通过使用v-model指令把{{property}}和文本框进行绑定;
 - ❑ 在创建V实例时, 传了一个选项对象。
 - ❑ 选项对象的el属性指向View, data属性指向Model, 这要就实现了双向绑定
 - ❑ {{property}}和文本框一方更新, 另一方也会做同样的更新
-

本章小结

- 了解Vue 主要特点, Vue 的优势
 - 明白Vue 的下载及使用
 - 如何创建 Vue 实例、数据和方法的基本使用
 - 数据挂载到 DOM 页面中
 - MVVM 模式理解 Vue 的数据双向绑定
-