

Vue基础

一、MVVM开发概念及SPA概念

Vue (读音 /vju:/, 类似于 view) 是一套用于构建用户界面的渐进式JavaScript框架。与其它大型框架不同的是, Vue 被设计为可以自底向上逐层应用。Vue 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 也完全能够为复杂的单页应用 (SPA) 提供驱动。**

在学习vue的框架之前先了解一个概念叫MVVM (model view and viewmodel), **其核心理念就是数据驱动页面**。与之相对应的有一套技术叫 **MVC**

在以前我们很多后端的开发框架里面都会提到MVC的概念 (Model View Controller), 这个MVC的概念是前后端结合的一种开发方式【目前基本上已经被替代】。现在的主流开发方式是前后端分离式开发 (前后端在开发的时候在2个不同的项目里面, 真正开发完了以后还可以再次合到一起) 如java里面会有 **servlet** 或 **spring MVC**, 如 **php** 语言里面的 **thinkPHP**, 如 **.net** 平台正同会有 **.net MVC**
现在这些技术在后端里面也慢慢的被淘汰了, 正是因为后端改变了开发方式, 如java里面的 **spring MVC** 现在换成了 **spring boot**, 那么前端也要相应的做出一些改变, 前端就从后端的代码里面分开了, 成了一个单独的项目

在目前的前端里面, 我们学习过很多与页面相关的知识, 如布局我们使用CSS, 如操作页面我们使用JS, 而JS在操作页面的时候使用 **DOM** 技术, dom技术本质也是一种开发思路。但是今天我们要完全放弃DOM操作了

MVVM不依赖于dom操作, 它完全造数据, 用数据去驱动页面

目前流行的MVVM框架或类似于MVVM核心理念的框架有很多, 主流的在以下几个

1. **vue** 【它是一个入门型的开发框架, 很容易上手, 但是提高很难】【尤雨溪, 中国人】
2. **react** 【上手难, 提高更难】【facebook】
3. **angular** 【难上加难, 它需要借用于typescript去完成】【google开发的】
4. **svelte** 【上手简单, 但是很考验基本功, 特别是原生JS和底层JS, 还有ES6】

那什么是SPA的概念呢

SPA的全称叫单页面应用程序, 指的是整个程序在经过处理以后只有一个页面

单页Web应用 (single page web application, SPA), 就是只有一张Web页面的应用, 是加载单个HTML页面并在用户与应用程序交互时动态更新该页面的Web应用程序

这里就再次说明, **vue/react/angular** 都可以实现单页面的开发

二、Vue的安装与引入

目前vue的版本主要包含在 **vue2** 和 **vue3**, 同时 **vue3** 里面包含了 **vue2** 基本上所有的功能核心, 所以在这里, 我们使用 **vue3** 的版本, 同讲解目前的2种语法格式

1. 第一种就是基于 **vue2** 的语法格式叫 **options api** 【选项式API】
2. 第二种就是基于 **vue3** 的语法格式叫 **composition api** 【组合式的API】

如果需要安装, 我们可以通过 **npm** 去下载, 也可以直接从官网去下载

vue2的官网: <https://v2.cn.vuejs.org/>

vue3的官网: <https://cn.vuejs.org/>

vue的作用是尤雨溪, 它是一个中国人, 以一人之力对抗google与facebook

```
1 $ npm info vue
```

在vue里面, 我们直接使用vue的话, 我们要使用 `vue.global.js` 这个文件

```
1 <script src="./js/vue.global.js"></script>
```

三、Vue接管页面数据

vue是一个**数据驱动页面的框架**, 数据驱动页面就是使用数据管理页面, **简而言之就是数据变化, 页面变化; 同时页面变化, 数据也变化**

数据与页面实现了双向的 **绑定**

在旧版本里面, 我们使用的命令如下

```
1 new Vue();
```

这一种方式在新版本里面会报错, **新版本里面使用如下**

```
1 <!DOCTYPE html>
2 <html lang="zh">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>vue的第一课</title>
9 </head>
10
11 <body>
12   <!-- Vue要去管理这个区域 -->
13   <div id="app">
14     <h2>大家好, 这是一个标题</h2>
15     <input type="text" v-model="userName">
16     <h1>{{userName}}</h1>
17   </div>
18 </body>
19 <script src="./js/vue.global.js"></script>
20 <script>
21   Vue.createApp({
22     //vue是接管页面的数据的, 那页面的数据应该来源于什么地方呢?
23     data() {
24       //返回的这个对象就是页面要接管的区域的数据
25       return {
26         userName: "标哥哥"
27       }
28     }
29   }).mount("#app"); //创建的vue的对象要接管id为box的区域
30 </script>
31
32 </html>
```

在上面的案例里面, 我们可以发现, 如果要继续操作页面, 已经可以不再依赖于DOM这种技术

四、Vue的数据绑定指令

1.插值语法

```
1   <body>
2     <div id="app">
3       <h2>{{userName}}</h2>
4       <h2>{{age}}</h2>
5       <h2>{{age>=18?"成年人":"未成年人"}}</h2>
6       <h2>{{num1.toString(2)}}</h2>
7       <h2>{{new Date().toLocaleDateString()}}</h2>
8       <h2>{{1+1}}</h2>
9       <div>
10        {{str1}}
11      </div>
12    </div>
13  </body>
14  <script src="./js/vue.global.js"></script>
15  <script>
16    Vue.createApp({
17      data() {
18        return {
19          userName: "张三",
20          age: 20,
21          num1: 18,
22          str1:"<input type='text' placeholder='请输入账号' />"
23        }
24      }
25    }).mount("#app");
26  </script>
```

插值语法通过 `{{}}` 形式在页面上显示数据，这个数据的来源是vue的 `data` 里在的东西，同时插件语法里面也可以写ES的普通代码，如条件运算符，如方法的调用
这一种插值语法就相当于设置了某一个元素的 `innerText` 的属性

2.v-text

这一种方式与上面的插值语法的试试是一模一样的，如下所示

```
1   <body>
2     <div id="app">
3       <h2>{{userName}}</h2>
4       <h2 v-text="age"></h2>
5       <h2 v-text="num1"></h2>
6       <div v-text="str1"></div>
7     </div>
8  </body>
9  <script src="./js/vue.global.js"></script>
10 <script>
11   Vue.createApp({
12     data() {
13       return {
14         userName: "张三",
15         age: 20,
16         num1: 18,
17         str1:"<input type='text' placeholder='请输入账号' />"
```

```

18         }
19     }
20     }).mount("#app");
21 </script>

```

这个 `v-text` 其实也相当于设置 `innerText` 的属性

3.v-html

```

1
2 <body>
3   <div id="app">
4     <h2>{{userName}}</h2>
5     <h2 v-text="userName"></h2>
6     <h2 v-html="userName"></h2>
7     <hr>
8     <div>{{str1}}</div>
9     <div v-text="str1"></div>
10    <div v-html="str1"></div>
11  </div>
12 </body>
13 <script src="./js/vue.global.js"></script>
14 <script>
15   Vue.createApp({
16     data() {
17       return {
18         userName: "张三",
19         age: 20,
20         num1: 18,
21         str1:"<input type='text' placeholder='请输入账号' />"
22       }
23     }
24   }).mount("#app");
25 </script>

```

这个与上面是不一样的，它相当于设置了元素的 `innerHTML` 的属性

张三

张三

张三

```
<input type='text' placeholder='请输入账号' />
```

```
<input type='text' placeholder='请输入账号' />
```

4.v-model

在页面上面显示数据的时候，我们还有一种特殊的元素就是表单如果，如果要绑定表单元素的值，则使用 `v-model`

```
1 <body>
2   <div id="app">
3     <!-- 表单元素的绑定是一个双向绑定的过程 -->
4     <input type="text" v-model="userName">
5     <h2>{{userName}}</h2>
6     <input type="range" v-model="age">{{age}}
7     <h2>{{age>=18?"你成年了":"你未成年"}}</h2>
8   </div>
9 </body>
10 <script src="./js/vue.global.js"></script>
11 <script>
12   Vue.createApp({
13     data() {
14       return {
15         userName: "张珊123",
16         age: 18
17       }
18     }
19   }).mount("#app");
20 </script>
```

张珊123

52

你成年了

下面还有单选框的表单绑定

```
1 <body>
2   <div id="app">
3     <input type="radio" value="男" v-model="a" name="sex">男
4     <input type="radio" value="女" v-model="a" name="sex">女
5     <hr>
6     <h2>你选择的性别是: {{a}}</h2>
7   </div>
8 </body>
9 <script src="./js/vue.global.js"></script>
10 <script>
11   Vue.createApp({
12     data() {
13       return {
14         a: "女"
15       }
16     }
17   }).mount("#app");
18 </script>
```

再来一个多选框的表单绑定

```
1 <body>
2   <div id="app">
3     <input type="checkbox" name="hobby" v-model="hobbyList" value="看书">看书
4     <input type="checkbox" name="hobby" v-model="hobbyList" value="睡觉">睡觉
5     <input type="checkbox" name="hobby" v-model="hobbyList" value="玩游戏">玩游
6     <input type="checkbox" name="hobby" v-model="hobbyList" value="吃饭">吃饭
7     <hr>
8     <h2>{{hobbyList}}</h2>
9   </div>
10 </body>
11 <script src="./js/vue.global.js"></script>
12 <script>
13   Vue.createApp({
14     data() {
15       return {
16         hobbyList: ["看书", "吃饭"]
17       }
18     }
19   }).mount("#app");
20 </script>
```

多选框的绑定需要使用数组的方式去完成

5.v-once

v-once 是单次绑定，它只会在第一次的时候做数据渲染，渲染完了以后就与vue断开了关联，后期如果数据发生了变化，这一个地方也不再变化

```
1 <body>
2   <!-- 创建接管区域 -->
3   <div id="app">
4     <input type="text" v-model="userName">
5     <h2>{{userName}}</h2>
6     <h2 v-text="userName"></h2>
7     <h2 v-html="userName"></h2>
8     <hr>
9     <h1 v-once>{{userName}}</h1>
10  </div>
11 </body>
12 <script src="./js/vue.global.js"></script>
13 <script>
14   // const app = Vue.createApp()
15   // app.mount("#app");
16   Vue.createApp({
17     data() {
18       return {
19         userName: "张珊"
20       }
21     }
22   }).mount("#app");
23 </script>
```

张珊

张珊

张珊

张珊

如果后期我们要是把输入框的值改变，那么下面的值也会改变，但是 `v-once` 的地方就不会再变了

张珊123

张珊123

张珊123

张珊

6.v-show

```
1 <body>
2   <div id="app">
3     <h2 v-show="true">今天是11月的第1天</h2>
4     <h2 v-show="false">天气也还不错</h2>
5     <h2 v-show="1>2">气温好像也不低,比较暖和</h2>
6     <h2 v-show="123">大家都在教室里面认真的听课</h2>
7     <h2 v-show="0">马淑圆很聪明</h2>
8     <!-- 大家要知道 6 个false的值是什么 -->
9     <h2 v-show="'0'">马淑圆很聪明,回答得非常好</h2>
10
11   </div>
12 </body>
13 <script src="./js/vue.global.js"></script>
14 <script>
15   Vue.createApp({
16
17   }).mount("#app")
18 </script>
```

通过上面的例子，我们可以看到，这里面显示或不显示已经变得非常简单的，只要 `v-show` 后面的值为真，就会显示这个元素。它在隐藏这个元素的时候，是通过 `display:none` 来完成的

```

▼ <div id="app" data-v-app>
  <h2>今天是11月的第1天</h2>
  <h2 style="display: none;">天气也还不错</h2> == $0
  <h2 style="display: none;">气温好像也不低,比较暖和</h2>
  <h2>大家都在教室里面认真的听课</h2>
  <h2 style="display: none;">马淑圆很聪明</h2>
  <!-- 大家要知道 6 个false的值是什么 -->
  <h2>马淑圆很聪明,回答得非常好</h2>
</div>
<script src="./js/vue.global.js"></script>

```

在工作当中, `v-show` 使用得非常频繁, 如下例子所示

```

1  <body>
2    <div id="app">
3      <input type="range" v-model="age">{{age}}
4      <h2 v-show="age<18">你的年龄没有达到18岁,所以你不能进网吧</h2>
5      <hr>
6      <input type="text" v-model="userName" placeholder="请输入用户名">
7      <span v-show="userName.length==0" style="color: red;">用户名不能为空</span>
8      <hr>
9      <input type="text" v-model="money">
10     <span v-show="!/^\\d+(\\.\\d+)?$/.test(money)" style="color: red;">必须是数字
11   </span>
12 </div>
13 </body>
14 <script src="./js/vue.global.js"></script>
15 <script>
16   Vue.createApp({
17     data() {
18       return {
19         age: 22,
20         userName: "",
21         money: 0
22       }
23     }
24   }).mount("#app")
25 </script>

```

22

请输入用户名 用户名不能为空

必须是数字

7.v-if, v-else, v-else-if

条件渲染, 根据条件来决定是否渲染

这个东西与我们的v-show看起来非常像, 但是本质上面是完全不一样的, 如下所示

```

1  <body>
2    <div id="app">

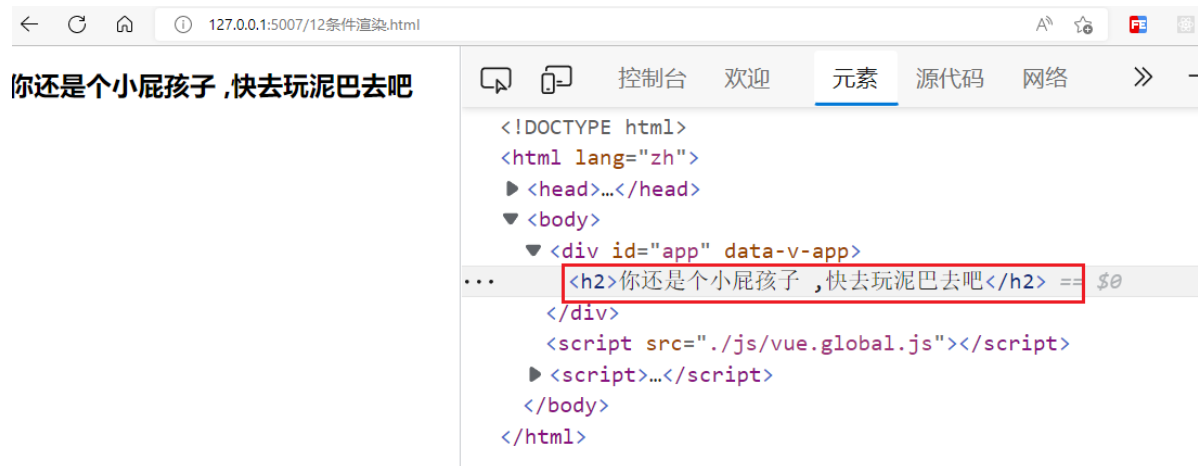
```



```

3       <h2 v-if="age>=18">你是成年人了,你可以去领结婚证了</h2>
4       <h2 v-else>你还是个小屁孩子 ,快去玩泥巴去吧</h2>
5     </div>
6   </body>
7   <script src="./js/vue.global.js"></script>
8   <script>
9     Vue.createApp({
10       data() {
11         return {
12           age: 10
13         }
14       }
15     }).mount("#app")
16   </script>

```



`v-if` 只是根据一个条件渲染了一个元素上去,而如果使用 `v-show` 的话,它是有多元素的,不符合条件的元素只是通过 `display:none` 隐藏掉了

同时要注意 `v-if` 还可以有多个条件去同时进行

```

1   <body>
2     <div id="app">
3       <h2 v-if="score>=90">优秀</h2>
4       <h2 v-else-if="score>=80">良好</h2>
5       <h2 v-else-if="score>=70">中等</h2>
6       <h2 v-else-if="score>=60">及格</h2>
7       <h2 v-else>不及格</h2>
8     </div>
9   </body>
10  <script src="./js/vue.global.js"></script>
11  <script>
12    Vue.createApp({
13      data() {
14        return {
15          score: 85
16        }
17      }
18    }).mount("#app")
19  </script>

```

在明天的时候,我们会讲到生命周期,讲到Vue的生命周期的时候,我们要重点再来看 `v-show` 与 `v-if` 的区别

8.v-for

列表渲染，学完这个东西以后，同学们就把 `art-template` 丢了,不用它了

```
1 <body>
2   <div id="app">
3     <h1>列表渲染</h1>
4     <!-- item代表的就是遍历的这一项,这个名子可以更改 -->
5     <h2 v-for="(item,index) in stuList">{{item}}----{{index}}</h2>
6   </div>
7 </body>
8 <script src="./js/vue.global.js"></script>
9 <script>
10   Vue.createApp({
11     data() {
12       return {
13         stuList: ["张珊", "李四", "王五", "赵六"]
14       }
15     }
16   }).mount("#app");
17 </script>
```

列表渲染

张珊----0

李四----1

王五----2

赵六----3

语法格式: `v-for="(当前遍历的项, 当前遍历的索引) in 集合"`

这种语法格式 我更想让同学们理解为 `v-for=(value ,key) in 对象`

```
1 <body>
2   <div id="app">
3     <h1>列表渲染</h1>
4     <!-- item代表的就是遍历的这一项,这个名子可以更改 -->
5     <h2 v-for="(item,index) in stuList">{{item}}----{{index}}</h2>
6     <hr>
7     <h2 v-for="(item,index) in Array.from(teachers)">{{item}}---{{index}}
8   </h2>
9     <hr>
10    <h2 v-for="(item,index) in obj">{{item}}---{{index}}</h2>
11    <hr>
12    <h2 v-for="(item,index) in obj2">{{item}}---{{index}}</h2>
13  </div>
```

```
13 </body>
14 <script src="./js/vue.global.js"></script>
15 <script>
16     Vue.createApp({
17         data() {
18             return {
19                 stuList: ["张珊", "李四", "王五", "赵六"],
20                 teachers: new Set(["标哥", "桃哥", "柴哥"]),
21                 obj: {
22                     0: "陈润福",
23                     1: "肖中",
24                     2: "魏成勇",
25                 },
26                 obj2: {
27                     userName: "小白",
28                     sex: "女"
29                 }
30             }
31         }
32     }).mount("#app");
33 </script>
```

列表渲染

张珊----0

李四----1

王五----2

赵六----3

标哥---0

桃哥---1

柴哥---2

陈润福---0

肖中---1

隗成勇---2

小白---userName

女---sex

当 `v-for` 与 `v-if` 同时进行的时候，这个时候在V3的版本下面是有一点区别改动的，如下所示

```
1 <body>
2   <div id="app">
3     <h2>v-for</h2>
4     <!-- 把这些东西渲染在ul下面的li里面,怎么办 -->
5     <!-- 现在只想显示偶数项 -->
6     <ul>
7       <template v-for="(item,index) in stuList">
8         <li v-if="index%2==0">{{item}}---{{index}}</li>
9       </template>
10    </ul>
11  </div>
12 </body>
```

```

13 <script src="./js/vue.global.js"></script>
14 <script>
15   Vue.createApp({
16     data() {
17       return {
18         stuList: ["张珊", "李四", "王五", "赵六", "田七"]
19       }
20     }
21   }).mount("#app");
22 </script>

```

这里要注意，`v-for` 与 `v-if` 可以写在一个元素上面，但是访问不到 `index` 的这个值，需要通过 `<template>` 再去包裹一层

v-for

- 张珊---0
- 王五---2
- 田七---4



9.v-if与v-for的扩展及注意事项

第一个注意事项

当我们同时去使用 `v-if` 与 `v-for` 的时候，看下面的代码

```

1 <body>
2   <div id="app">
3     <ul>
4       <li v-for="(item,index) in stus">{{item}}----{{index}}</li>
5     </ul>
6     <hr>
7     <ul>
8       <li v-for="(item,index) in stus" v-if="index%2==0">{{item}}----
9         {{index}}</li>
10    </ul>
11  </div>
12 </body>
13 <script src="./js/vue.global.js"></script>
14 <script>
15   Vue.createApp({
16     data() {
17       return {
18         stus: ["张三", "李四", "王五", "赵六", "田七"]

```

```

18         }
19     }
20     }).mount("#app")
21 </script>

```

当我们在执行第2个遍历的时候，就会报警告信息

You are running a development build of Vue. [vue.global.js:10938](#)
 Make sure to use the production build (*.prod.js) when deploying for production.

⚠ [Vue warn]: Property "index" was accessed during render [vue.global.js:1622](#) ⓘ
 out is not defined on instance.
 at <App>

[Five Server] connecting... [fiveserver.js:1](#)

[Five Server] connected. [fiveserver.js:1](#)

原理是因为 `v-if` 的优先级比 `v-for` 要高一些，当2这个指令在同一个元素上面的时候，它会先执行 `v-if`，再去执行 `v-for`

在上面的代码里面，当我们先去执行 `v-if` 的时候，里面的条件是 `index%2==0`，这个时候因为还没有执行 `v-for`，所以 `index` 还不存在，这个时候就会报警告信息

⚠ 注意

同时使用 `v-if` 和 `v-for` 是不推荐的，因为这样二者的优先级不明显。请转阅[风格指南](#)查看更多细节。

当它们同时存在于一个节点上时，`v-if` 比 `v-for` 的优先级更高。这意味着 `v-if` 的条件将无法访问到 `v-for` 作用域内定义的变量别名：

在外新包装一层 `<template>` 再在其上使用 `v-for` 可以解决这个问题 (这也更加明显易读)：

```

<template v-for="todo in todos">
  <li v-if="!todo.isComplete">
    {{ todo.name }}
  </li>
</template>

```

第二个注意事项

在 `V3` 里面执行 `v-for` 的遍历时候，不一定非要是数组，还可以是对象，对象在遍历的时候，它的语法如下

```
1 <元素 v-for="(value,key,index) in 对象"></元素>
```

所以看下面的案例

```

1 <body>
2   <div id="app">
3     <h2 v-for="(value,key,index) in obj">{{value}}---{{key}}---{{index}}</h2>
4   </div>
5 </body>

```

```

6   <script src="./js/vue.global.js"></script>
7   <script>
8       Vue.createApp({
9           data() {
10              return {
11                  obj: {
12                      userName: "张三",
13                      sex: "男"
14                  }
15              }
16          }
17      }).mount("#app")
18  </script>

```

张三---userName---0

男---sex---1

第三个注意事项

v-for 还可以执行普通的 **for** 循环了，如下所示

```

1   <!-- item从1开始, index从0开始 -->
2   <h2 v-for="(item,index) in 10">{{item}}---{{index}}</h2>

```

第四个注意事项

v-for 在执行遍历的时候可以直接解构了

```

1   <body>
2       <div id="app">
3           <ul>
4               <li v-for="(item,index) in stuList">{{item.stuName}}</li>
5           </ul>
6           <hr>
7           <ul>
8               <li v-for="({stuName,sex},index) in stuList">{{stuName}}---{{sex}}
9           </li>
10          </ul>
11      </div>
12  </body>
13  <script src="./js/vue.global.js"></script>
14  <script>
15      Vue.createApp({
16          data() {
17              return {
18                  stuList: [{
19                      stuName: "张珊",
20                      sex: "女"
21                  }, {
22                      stuName: "李四",
23                      sex: "男"
24                  }, {
25                      stuName: "王五",

```

```

25         sex: "男"
26     }
27 }
28 }
29 }).mount("#app")
30 </script>

```

- 张珊---女
- 李四---男
- 王五---男

10.v-once与v-show的惰性

```

<body>
<div id="app">
  <input type="text" v-model="userName"> <button type="button" @click="flag=true">按钮</button>
  <h2>{{userName}}---{{flag}}</h2>
  <h2 v-once v-show="flag">{{userName}}</h2>
</div>
</body>
<script src="./js/vue.global.js"></script>
<script>
  Vue.createApp({
    data() {
      return {
        userName: "张珊",
        flag: false
      }
    }
  }).mount("#app")
</script>

```

当我们把输入的框的值改变 张珊123以后，再去点击按钮的时候，这个时候下面的h2的结果应该是A →
 把张珊改成张珊123以后，点击按钮，flag变为true下的h2标签 会出现什么结果
 A：不会做任何变化
 B：会显示，显示内容是第一次的 张珊
 C：会显示，显示张珊123
 D：程序报错

张珊123

按钮

张珊123---true

11.v-bind

之前我们所学习的所有的指令都是让页面上面显示内容，但是如果想对一个HTML的属性进行绑定动态的值，要使用 **v-bind:属性** 这一种方式来完成

```

1 <body>
2   <div id="app">
3     <!-- {{}}相当于innerText -->
4     <!-- 如果要绑定一个属性 -->
5     <a v-bind:href="y">{{x}}</a>
6   </div>
7 </body>
8 <script src="./js/vue.global.js"></script>
9 <script>
10   Vue.createApp({
11     data() {

```



```

12         return {
13             x: "百度一下",
14             y: "https://www.baidu.com"
15         }
16     }
17     }).mount("#app")
18 </script>

```

```

▼ <div id="app" data-v-app>
  <!-- {{}} 相当于 innerText -->
  <!-- 如果要绑定一个属性 -->
  <a href="https://www.baidu.com">百度一下</a>
</div>
<script src="./js/vue.global.js"></script>

```

因为属性绑这种操作在后期会经常使用，所以 `vue` 里面提供了一种简写的办法

```

<div id="app">
  <!-- {{}} 相当于 innerText -->
  <!-- 如果要绑定一个属性 -->
  <a v-bind:href="y">{{x}}</a>
  <hr>
  <a :href="y">{{x}}</a>
</div>
</body>
<script src="./js/vue.global.js"></script>

```

简写的属性绑定

Vue的事件及事件对象

`vue` 是数据驱动页面的框架，所以它不仅仅可以接管页面的数据，还可以接管页面的事件

1. vue绑定事件的方式

```

1 <button type="button" v-on:click="">按钮1</button>

```

在上面的代码里面，我们可以看到 `v-on:事件名` 就是 `vue` 里面的事件绑定

因为事件的绑定也是一个非常常见的操作，所以在这里它也有简化的方法

```

1 <button type="button" @click="">按钮1</button>

```

它的简化方法就是 `@事件名` 即可

2. 以方法的形式接管事件

```
1  <body>
2    <div id="app">
3      <!-- 在接管区域里面，如果要做事件绑定，非常简单 -->
4      <!-- 这里的事件方法是没有加括号，因为它不需要传递参数 -->
5      <button type="button" v-on:click="sayHello">按钮1</button>
6      <!-- 这里的事件方法加了括号，加了括号以后，我们就可以传递参数 -->
7      <button type="button" v-on:click="abc(18)">按钮2</button>
8    </div>
9  </body>
10 <script src="./js/vue.global.js"></script>
11 <script>
12   Vue.createApp({
13     // 负责接管页面的数据
14     data() {
15       return {
16
17       }
18     },
19     // 接管页面的事件方法
20     methods: {
21       sayHello() {
22         alert("你好啊，我在弹窗");
23       },
24       abc(age) {
25         alert(`我今年${age}岁了`)
26       }
27     }
28   }).mount("#app");
29 </script>
```

代码分析

1. `data` 是负责接管页面的数据的，`methods` 就是负责接管页面上面的方法
2. `v-on:事件` 可以赋值一个函数名，这个函数名可以有括号，也可以没有括号【如果你需要传递参数，你就添加括号，如果你不需要传递参数，则这个括号就可以省略】

3. 以行内的方式接管事件

这一种写法非常简单，它直接把代码写在了事件的属性名里面

```
1  <body>
2    <div id="app">
3      <h2>{{userName}}</h2>
4      <button type="button" @click="aaa">按钮1</button>
5      <button type="button" @click="userName = '王五'">按钮2</button>
6    </div>
7  </body>
8  <script src="./js/vue.global.js"></script>
9  <script>
10   Vue.createApp({
11     data() {
12       return {
```

```

13         userName: "张三"
14     },
15 },
16     methods: {
17         aaa() {
18             this.userName = "李四";
19         }
20     }
21
22     }).mount("#app");
23 </script>

```

注意事项

1. 如果行内的事件代码要操作vue自身的数据里面的数据，不要加 `this` 【在vue的托管区域内部是严禁出现this关键字】
2. 如果是行内的事件代码，它内部没有 `window`，所以也就没有 `window` 下面的那些对象

4. vue的方法访问vue的数据

掌握使用方法，了解一下原理就可以了

```

1 <body>
2   <div id="app">
3     <h2>{{girlName}}</h2>
4     <button type="button" v-on:click="test1">按钮1</button>
5   </div>
6 </body>
7 <script src="./js/vue.global.js"></script>
8 <script>
9   Vue.createApp({
10     // 负责接管页面的数据
11     data() {
12       return {
13         girlName: "小红"
14       }
15     },
16     // 接管页面的事件方法
17     methods: {
18       test1() {
19         //如果在事件方法里面,想拿到自己的data里面的数据，可以直接通过自己的对象来完成
20
21         // 完整的写法
22         console.log(this.$data.girlName);
23         console.log(this.girlName); //它是一个代理
24         //当面试官问你，为什么this.girlName可以取到this.$data.girlName
25         //因为this[vue]对象它是一个proxy代理对象，内部的handler代理方法代理了
26         // $data的取值操作
27         console.log(this);
28       }
29     }).mount("#app");
30 </script>

```

这里可以会涉及到vue的源代码里面的代理模式这一块，先暂时不用了解

综合所述，如果要在方法里面访问data里面的数据则通过 `this.数据名` 就可以得到

vue的事件对象

在vue里面，我们可以在事件方法里面使用 `$event` 来获取事件对象，如下所示

```
1 <body>
2   <div id="app">
3     <button type="button" @click="aaa($event)">按钮</button>
4   </div>
5 </body>
6 <script src="./js/vue.global.js"></script>
7 <script>
8   Vue.createApp({
9     methods: {
10       aaa(event) {
11         console.log("你好");
12         console.log(event);
13       }
14     }
15   }).mount("#app")
16 </script>
```

Vue事件修饰符

修饰符：用来装饰某一个东西，所以事件修饰符应该也是用来装饰事件的

```
<body>
  <div id="app">
    <div class="box" @click="aaa">
      <button type="button" @click="bbb($event)">按钮</button>
    </div>
  </div>
</body>
<script src="./js/vue.global.js"></script>
<script>
  Vue.createApp({
    methods: {
      aaa(){
        console.log("我是aaa的事件")
      },
      bbb(event){
        console.log("我是bbb的事件")
        event.stopPropagation();
      }
    }
  }).mount("#app")
</script>
```

在以前如果要阻止事件的冒泡，我们要获取事件对象以后再进行操作

但是vue的操作已经变得非常简单了

```

body>
  <div id="app">
    <div class="box" @click="aaa">
      <button type="button" @click.stop="bbb">按钮</button>
    </div>
  </div>
</body>
<script src="/js/vue.global.js"></script>

```

如果想阻止事件的冒泡，可以直接使用这个修饰符

上面的 `@click.stop` 就是使用事件修饰了，`stop` 的修饰符就是阻止事件冒泡，与之相似的修饰符还有很多，我现在列举一下

事件修饰符

1. `.stop`
2. `.prevent`
3. `.self`
4. `.capture`
5. `.once`
6. `.passive`

按键修饰符

Vue 为一些常用的按键提供了别名：

- `.enter`
- `.tab`
- `.delete` (捕获“Delete”和“Backspace”两个按键)
- `.esc`
- `.space`
- `.up`
- `.down`
- `.left`
- `.right`

鼠标修饰符

- `.left`
- `.right`
- `.middle`

vue的小案例

```

1  <body>
2    <div id="app">
3      <!-- 在输入框里面输入内容以后，回车，添加在下面的列表里面 -->
4      <input type="text" v-model.trim="txt" @keydown.enter="stuList.add(txt)">
5      <hr>
6      <ul>
7        <!-- 当我们双击某一项的时候，要把某一项删除 -->
8        <li v-for="(item,index) in stuList" @dblclick="stuList.delete(item)">
9          {{item}}---{{index}}</li>
10     </ul>
11   </div>
</body>

```

```

12 <script src="./js/vue.global.js"></script>
13 <script>
14   Vue.createApp({
15     data() {
16       return {
17         txt: "",
18         stuList: new Set(["张三", "李四"])
19       }
20     }
21   }).mount("#app");
22 </script>

```

this.\$set与this.\$delete使用

这个问题在vue3的版本里面已经做了处理，不需要再使用 `this.$set()` 和 `this.$delete()` 去完成了，因为 `vue3` 的内部使用的是代理全局拦截，即使是新增的属性也可以实现响应

以下的代码是vue2的代码

```

1 <body>
2   <div id="app">
3     <h2>{{userInfo.userName}}</h2>
4     <h2>{{userInfo.sex}}</h2>
5     <button type="button" @click="aaa">按钮1</button>
6   </div>
7 </body>
8 <script src="./js/vue.js"></script>
9 <script>
10   new Vue({
11     el: "#app",
12     data: {
13       userInfo: {
14         userName: "张三"
15       }
16     },
17     methods: {
18       aaa() {
19         // 第一种方式
20         // this.userInfo.sex = "男";
21         // 强制更新, 强制重新渲染页面, 这个非常消耗性能
22         // this.$forceUpdate();
23
24         // 第二种方法
25         this.$set(this.userInfo, "sex", "男");
26
27         // 可以实现新增的属性不能响应的问题
28       }
29     }
30   })
31 </script>
32

```

在上面的代码上面，我们可以看到，最开始渲染的时候，`userInfo` 这个对象里面是没有 `sex` 这个属性的，但是后面我们主动追加了这个属性，结果，页面并没有正常显示这个属性，因为vue对新增的属性并没有实现响应

如果要解决这个问题，可以使用下面2个方法

1. `this.$forceUpdate()` 强制更新vue的状态，但是会非常消耗浏览器的性能
2. 使用 `this.$set()` 来扩展新属性，这样扩展出来的新属性 就是响应的

同理，`this.$delete` 也是一样的效果

```
1 <body>
2   <div id="app">
3     <h2 v-show="!userInfo.flag">{{userInfo.userName}}</h2>
4     <button type="button" @click="aaa">按钮</button>
5   </div>
6 </body>
7 <script src="./js/vue.js"></script>
8 <script>
9   new Vue({
10     el: "#app",
11     data: {
12       userInfo: {
13         userName: "张三",
14         flag: 123
15       }
16     },
17     methods: {
18       aaa() {
19         //第一种情况
20         // delete this.userInfo.flag;
21         // this.$forceUpdate();
22
23         this.$delete(this.userInfo, "flag");
24       }
25     }
26   })
27 </script>
```

vue计算属性

之前我们讲过，vue是数据驱动页面的，它的数据主要来源于 `data` 这个区域，那么在这里我要扩展一下，接管区域里面的数据不一定仅仅只来源于data,还可以来源于其它的方

计算属性是需要经过计算以后才会得到的值，如下所示

```
1 <body>
2   <div id="app">
3     <h2>{{userName}}</h2>
4     <hr>
5     <h2>男生: 3</h2>
6     <h2>女生: 2</h2>
7   </div>
```

```

8   </body>
9   <script src="./js/vue.global.js"></script>
10  <script>
11      Vue.createApp({
12          // 接管区域里面的数据来源于data
13          data() {
14              return {
15                  userName: "张三",
16                  stuList: [{
17                      stuName: "岳圣哲",
18                      sex: "男"
19                  }, {
20                      stuName: "黄相立",
21                      sex: "男"
22                  }, {
23                      stuName: "王杜娉",
24                      sex: "女"
25                  }, {
26                      stuName: "马淑圆",
27                      sex: "女"
28                  }, {
29                      stuName: "兰宏宇",
30                      sex: "男"
31                  }]
32              }
33          }
34      }).mount("#app")
35  </script>

```

在上面的代码里面，如果我们想得到男或女的人数一共是多少，这样就非常困难，所以如果要实现这样的点，我们必须将数据经过计算以后才会得到结果，对于这一种需求，vue是可以实现的

现在我们先从一个简单的Demo来入手

```

1   <body>
2       <div id="app">
3           <h2>{{userName}}</h2>
4           <hr>
5           <h2>{{aaa}}</h2>
6           <h2>{{bbb}}</h2>
7       </div>
8   </body>
9   <script src="./js/vue.global.js"></script>
10  <script>
11      Vue.createApp({
12          data() {
13              return {
14                  userName: "张三"
15              }
16          },
17          // 计算属性
18          computed: {
19              //每一个计算属性本质上面就是一个方法，只是这个方法有返回值了
20              aaa() {
21                  return "hello world";

```



```

22         },
23         bbb() {
24             return ~~(Math.random() * 100)
25         }
26     },
27     },
28     ).mount("#app");
29 </script>

```

在上面的代码里面，我们可以看到的，页面上面的数据不仅仅只来源于 `data`，还可以来源于 `computed` 下面的计算属性

张三

hello world

53

计算属性的本质上面其实就是一个函数，这个函数需要有一个返回值而已，同时还要注意一点，计算属性也是可以访问 `data` 里面的数据的。如下所示

```

1  <script>
2      Vue.createApp({
3          data() {
4              return {
5                  userName: "张三",
6                  age: 19
7              }
8          },
9          // 计算属性
10         computed: {
11             //每一个计算属性本质上面就是一个方法，只是这个方法有返回值了
12             aaa() {
13                 return "hello world";
14             },
15             bbb() {
16                 return ~~(Math.random() * 100)
17             },
18             ccc() {
19                 return this.age + 20;    //在这里，我们使用了data里面的变量
20             }
21         },
22     }).mount("#app");
23 </script>

```

经过了上面的demo以后，我们再来看一下刚刚的问题

```

1
2  <body>
3      <div id="app">
4          <h2>{{userName}}</h2>

```

```

5         <hr>
6         <h2>男生: {{boyCount}}</h2>
7         <h2>女生: {{girlCount}}</h2>
8     </div>
9 </body>
10 <script src="./js/vue.global.js"></script>
11 <script>
12     Vue.createApp({
13         // 接管区域里面的数据来源于data
14         data() {
15             return {
16                 userName: "张三",
17                 stuList: [{
18                     stuName: "岳圣哲",
19                     sex: "男"
20                 }, {
21                     stuName: "黄相立",
22                     sex: "男"
23                 }, {
24                     stuName: "王杜娉",
25                     sex: "女"
26                 }, {
27                     stuName: "马淑圆",
28                     sex: "女"
29                 }, {
30                     stuName: "兰宏宇",
31                     sex: "男"
32                 }]
33             }
34         },
35         computed: {
36             // 男生数量
37             boyCount() {
38                 let count = this.stuList.filter(item => item.sex ===
"男").length;
39                 return count;
40             },
41             //女生数量
42             girlCount() {
43                 return this.stuList.filter(item => item.sex === "女").length;
44             }
45         }
46     }).mount("#app")
47 </script>

```

上面的 `computed` 里面的东西就是计算属性，它是一个函数，对 `data` 里面的 `stuList` 做了二次计算，然后返回了一个结果，这个结果当前计算属性的值在调用计算属性的时候，我们不用加上这个括号

vue侦听属性（监听器）

```
1   <body>
2     <div id="app">
3       <h2>{{userName}}</h2>
4       <button type="button" @click="userName='小姐姐'">改变userName的值</button>
5     </div>
6   </body>
7   <script src="./js/vue.global.js"></script>
8   <script>
9     Vue.createApp({
10       data() {
11         return {
12           userName: "张三",
13         }
14       },
15       // 监听器
16       watch: {
17         //你要监听谁，你就哪个数据的同名函数
18         userName(newValue, oldValue) {
19           console.log("你发生了变化");
20           // newValue代表更改之后的值，oldValue更改之前的值
21           console.log(newValue, oldValue);
22         }
23       }
24     }).mount("#app");
25   </script>
```

监听器就是写在 `watch` 下面的一个函数，如果要监听某一个变量，写写这个变量的同名函数，当这个变量发生变化的时候就会自动调用这个函数，同时这个函数会有2个参数，第一个参数代表变化以后的值，第二个参数代表变化之前的值

深度监听

在监听器里在，默认情况下，它只能监听基本数据类型，对于对象的变化它是监听不到的，如果想实现对象的监听，应该使用深度监听

```
1 <body>
2   <div id="app">
3     <h2>{{userInfo.userName}}</h2>
4     <button type="button" @click="userInfo.userName = '小姐姐'">我要变化</button>
5   </div>
6 </body>
7 <script src="./js/vue.global.js"></script>
8 <script>
9   Vue.createApp({
10     data() {
11       return {
12         userInfo: {
13           userName: "张珊"
14         }
15       }
16     },
17     watch: {
18       userInfo(newValue,oldValue){
19         console.log("你发生变化了")
20       }
21     }
22   }).mount("#app");
23 </script>
```

The diagram illustrates the Vue.js data structure. It shows a `userInfo` object with a `userName` property. A red box highlights the `userInfo` object, and a red arrow points to it with the text: "当对象的内部发生变化以后，是监听不到的" (When the internal state of the object changes, it cannot be monitored). This indicates that the current `watch` configuration only monitors shallow changes.

```
1 <body>
2   <div id="app">
3     <h2>{{userInfo.userName}}</h2>
4     <button type="button" @click="userInfo.userName = '小姐姐'">我要变化
5   </button>
6 </div>
7 </body>
8 <script src="./js/vue.global.js"></script>
9 <script>
10   Vue.createApp({
11     data() {
12       return {
13         userInfo: {
14           userName: "张珊"
15         }
16       }
17     },
18     watch: {
19       // userInfo(newValue,oldValue){
20       //   console.log("你发生变化了")
21       // }
22       userInfo: {
23         deep: true,
24         handler(newValue,oldValue){
25           console.log(newValue)
26           console.log("我看你发生了变化了");
27         }
28       }
29     }
30   }).mount("#app");
31 </script>
```

上面的 `userInfo` 就是实现了深度监听，深度监听不仅仅只是一个单独的函数了，它是一个对象，这个对象上面有 `deep` 这个属性，用于设置是否处于深度监听状态，还有一个 `handler` 函数是变化以后的处理函数

案例

使用vue来完成下面的操作部分

购物车列表

序号	商品名称	价格	数量	总价
1	iphone 14	4999	<div>- 1 +</div>	4999
2	充电宝	129	<div>- 3 +</div>	387
3	数据线	13	<div>- 2 +</div>	26
4	红米手机	1999	<div>- 2 +</div>	3998

商品总数：8件，商品总金额9410元

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>练习</title>
9      <style>
10         .page-title {
11             text-align: center;
12         }
13
14         .table1 {
15             border: 1px solid black;
16             width: 1200px;
17             margin: auto;
18             border-collapse: collapse;
19         }
20
21         .table1 :is(td, th) {
22             border: 1px dotted black;
23             height: 35px;
24             padding: 5px;
25         }
26
27         .total-info {
28             font-weight: bold;
29             font-size: 22px;
30             width: 1200px;
31             margin: auto;
32             padding: 10px 0;
33         }
34     </style>
35 </head>
36
37 <body>
38     <div id="app">
39         <h2 class="page-title">购物车列表 </h2>
```

```

40     <table class="table1">
41         <tr>
42             <th>序号</th>
43             <th>商品名称</th>
44             <th>价格</th>
45             <th>数量</th>
46             <th>总价</th>
47         </tr>
48         <tr v-for="(item,index) in goodsList">
49             <td>{{index+1}}</td>
50             <td>{{item.goodsName}}</td>
51             <td>{{item.price}}</td>
52             <td>
53                 <!-- <button type="button" @click="item.count>1&&item.count-
54                 --></button> -->
55                 <!-- <button type="button" @click="item.count--" v-
56                 if="item.count>1"></button> -->
57                 <button type="button" @click="item.count--"
58                 :disabled="item.count<=1"></button>
59                 {{item.count}}
60                 <button type="button" @click="item.count++">+</button>
61             </td>
62             <td>{{item.count*item.price}}</td>
63         </tr>
64     </table>
65     <div class="total-info">
66         商品总数: {{totalInfo.totalCount}}件, 商品总金额{{totalInfo.totalMoney}}
67         元
68     </div>
69 </div>
70 </body>
71 <script src="./js/vue.global.js"></script>
72 <script>
73     Vue.createApp({
74         data() {
75             return {
76                 // 商品数据
77                 goodsList: [{
78                     goodsName: "iphone 14",
79                     price: 4999,
80                     count: 1
81                 }, {
82                     goodsName: "充电宝",
83                     price: 129,
84                     count: 3
85                 }, {
86                     goodsName: "数据线",
87                     price: 13,
88                     count: 2
89                 }, {
90                     goodsName: "红米手机",
91                     price: 1999,
92                     count: 2
93                 }
94             ]
95         }
96     })

```

```

92         computed: {
93             totalInfo() {
94                 //一次性得到totalCount和totalMoney
95                 let totalCount = 0,
96                     totalMoney = 0;
97                 this.goodsList.forEach(({count,price}) => {
98                     totalCount += count;
99                     totalMoney += count * price;
100                 });
101                 return {
102                     totalCount,
103                     totalMoney
104                 }
105             }
106         }
107     }).mount("#app");
108 </script>
109
110 </html>

```

vue样式class绑定

为什么需要样式绑定？

在以前的dom开发里面，如果我们要动态的添加切换页面元素的样式，需要动态的去设置 `classList` 或 `style` 属性，这样操作起来非常麻烦，如下Demo所示

```

1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>没有动态样式绑定</title>
9      <style>
10         .ul1 li {
11             height: 40px;
12             line-height: 40px;
13             border: 2px solid black;
14         }
15
16         /* 点击哪一项，把哪一项的背景设置为红色，字体加粗 */
17         .ul1 li.active {
18             font-weight: bold;
19             background-color: red;
20         }
21     </style>
22 </head>
23
24 <body>
25     <ul class="ul1">
26         <li>第一项</li>
27         <li>第二项</li>

```

```

28     <li>第三项</li>
29   </ul>
30 </body>
31 <script>
32   let ul1 = document.querySelector(".ul1");
33   ul1.addEventListener("click", function(event) {
34     if (event.target.matches("li")) {
35       //这里就证明是li点击了
36
37       //删除之前的效果
38       let activeElement = document.querySelector(".ul1>li.active");
39       if(activeElement){
40         activeElement.classList.remove("active");
41       }
42       event.target.classList.add("active");
43     }
44   });
45 </script>
46
47 </html>

```

- 第一项
- 第二项
- 第三项

上面的代码就是当我们点击某一项li的时候，给它添加一个样式，其它的样式则移除，这种效果做起来会非常麻烦，同时因为每次操作都要去查找DOM元素，所以它的效果非常低下

在现有的vue的开发里面，因为不再纯粹的依赖于DOM操作，所以可使用Vue里面的数据驱动页面的思路来完成

*如果想改变界面，则应该是改变了数据，数据vue是双向数据绑定，数据改变页面改变，页面改变数据也改变

```
9     <style>
10       .box {
11         width: 200px;
12         height: 200px;
13         border: 2px solid black;
14       }
15
16       /* 如何让这个元素动态的添加或删除active这个样式呢 */
17
18       .box.active {
19         font-weight: bold;
20         background-color: red;
21       }
22     </style>
23   </head>
24
25   <body>
26     <div id="app">
27       <!-- 根据需求，这个时候其实我们就发现，这个class里面的值它不再是一个固定的值，是一个动态的 -->
28       <div class="box">这是一个盒子</div>
29     </div>
30   </body>
31   <script src="./js/vue.global.js"></script>
32   <script>
33     Vue.createApp({
34
35     }).mount("#app")
36   </script>
37 </html>
```

动态样式绑定的语法有2种，如所

1. 对象语法

它的语法格式如下

```
1 <元素 :class="{样式名:布尔值, 样式名:布尔值}"></元素>
```

如果布尔值的结果为真，则当前的样式生效，样式名后面的布尔值不成立，则不应该这个样式

```
1 <div class="box" :class="{active:true}">这是一个盒子</div>
```

在上面的代码里面，因为 `active` 的样式后面接的是一个 `true`，那么我们就可以看到它的样式生效了

```
<div class="box active">这是一个盒子</div> == $0
```

```
1 <!DOCTYPE html>
2 <html lang="zh">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>动态样式绑定</title>
9   <style>
10     .box {
11       width: 200px;
12       height: 200px;
13       border: 2px solid black;
14     }
15
```

```

16      /* 如何让这个元素动态的添加或删除active这个样式呢 */
17
18      .box.active {
19          font-weight: bold;
20          background-color: red;
21      }
22  </style>
23 </head>
24
25 <body>
26   <div id="app">
27     <!-- 根据要求，这个时候其实我们就发现，这个class里面的值它不再是一个固定的值，是一个动态的 -->
28     <div class="box" :class="{active:flag}" @click="flag=!flag">这是一个盒子
29   </div>
30 </body>
31 <script src="./js/vue.global.js"></script>
32 <script>
33   Vue.createApp({
34     data(){
35       return {
36         flag:false
37       }
38     }
39   }).mount("#app")
40 </script>
41
42 </html>

```

代码分析

在上面的代码里面我们可以看到，active的样式是否生效主要看后面的 `flag` 变量，这个变量默认是 `false`，然后我们在当前的元素上面添加了 `click` 事件，并且设置 `flag=!flag`，这样 `flag` 的值就会在 `true/false` 之间来回切换，那么样式也就实现了切换的效果

现在我们来通过一个案例看效果

```

1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <title>vue的classs样式绑定</title>
9    <style>
10      .ul1>li {
11        height: 40px;
12        line-height: 40px;
13        border: 1px solid black;
14      }
15
16      .ul1>li.selected {
17        background-color: red;
18        font-weight: bold;

```

```

19     }
20     </style>
21 </head>
22
23 <body>
24     <div id="app">
25         <ul class="ul1">
26             <li v-for="(item,index) in list" :class="
27 {selected:currentIndex==index}"
28             @click="currentIndex=index">{{item}}</li>
29         </ul>
30     </div>
31 </body>
32 <script src="./js/vue.global.js"></script>
33 <script>
34     Vue.createApp({
35         data() {
36             return {
37                 list: ["第一项", "第二项", "第三项"],
38                 currentIndex: -1
39             }
40         }
41     }).mount("#app")
42 </script>
43 </html>

```

- 第一项
- 第二项
- 第三项

2.数组语法

它的语法格式如下

```

1 <元素 :class="[条件?'样式一':'样式二',条件?'样式一':'样式二']"></元素>

```

数组语法本质上面是一个条件运算符

```

1 <div class="box" :class="[2>1?'active':null]">这是一个盒子</div>

```

现在我们根据语来看一下代码

```

1 <!DOCTYPE html>
2 <html lang="zh">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width, initial-scale=1.0">
8     <title>数组语法</title>
9     <style>

```

```

10     .box {
11         width: 200px;
12         height: 200px;
13         border: 2px solid black;
14         display: flex;
15         justify-content: center;
16         align-items: center;
17     }
18
19     .active {
20         font-weight: bold;
21         background-color: pink;
22     }
23
24     .unActive {
25         background-color: lightseagreen;
26     }
27 </style>
28 </head>
29
30 <body>
31     <div id="app">
32         <div class="box" :class="[age>18?'active':'unActive']"
33         @click="flag=!flag">
34             你的年龄是{{age}}
35             <button type="button" @click="age++">+</button>
36         </div>
37     </div>
38 <script src="./js/vue.global.js"></script>
39 <script>
40     Vue.createApp({
41         data() {
42             return {
43                 age: 1
44             }
45         }
46     }).mount("#app")
47 </script>
48
49 </html>

```

在上面的代码里面，我们使用 `:class="[age>18?'active':'unActive']"` 去完成了动态样式的切换

vue样式style绑定

有了class的动态绑定以后，为什么还需要style的绑定，因为 `style` 属性直接操作样式，不要单独去书写一个类，有时候会很方便

1.对象语法

```
1 <元素 :style="{属性名:属性值, 属性名:属性值}"></元素>
```

```
1 <!DOCTYPE html>
2 <html lang="zh">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>style绑定</title>
9   <style>
10     .box {
11       width: 100px;
12       height: 100px;
13       border: 2px solid black;
14     }
15   </style>
16 </head>
17
18 <body>
19   <div id="app">
20     <div class="box" :style="{backgroundColor:c}"></div>
21     <hr>
22     <button type="button" @click="c='red'">红色</button>
23     <button type="button" @click="c='blue'">蓝色</button>
24     <button type="button" @click="c='yellow'">黄色</button>
25   </div>
26 </body>
27 <script src="./js/vue.global.js"></script>
28 <script>
29   Vue.createApp({
30     data() {
31       return {
32         c: "white"
33       }
34     }
35   }).mount("#app")
36 </script>
37
38 </html>
```

在上面的代码里面，我们看到了 `style` 的绑定的用法

```
1 <!DOCTYPE html>
2 <html lang="zh">
3
4 <head>
5   <meta charset="UTF-8">
6   <meta http-equiv="X-UA-Compatible" content="IE=edge">
7   <meta name="viewport" content="width=device-width, initial-scale=1.0">
8   <title>style的绑定</title>
9   <style>
10     .progress {
11       width: 400px;
```

```

12         height: 30px;
13         border: 1px solid lightgray;
14         border-radius: 15px;
15         box-shadow: 0 0 15px 5px gray;
16         background-image: linear-gradient(to right, red, red);
17         background-repeat: no-repeat;
18         /* background-size: 50% 100%; */
19         display: flex;
20         justify-content: center;
21         align-items: center;
22     }
23
24     span.active {
25         color: white;
26     }
27 </style>
28 </head>
29
30 <body>
31     <div id="app">
32         <div class="progress" :style="{backgroundSize:`${percentage}% 100%`}">
33             <span :class="{active:percentage>55}">{{percentage}}%</span>
34         </div>
35         <hr>
36         <button type="button" @click="percentage--">--</button>
37         <button type="button" @click="percentage++">+</button>
38     </div>
39 </body>
40 <script src="./js/vue.global.js"></script>
41 <script>
42     Vue.createApp({
43         data() {
44             return {
45                 percentage: 50
46             }
47         },
48         //监听样式
49         watch: {
50             percentage(newValue, oldValue) {
51                 if (newValue > 100) {
52                     //大于100
53                     this.percentage = 100;
54                 } else if (newValue < 0) {
55                     //小于0
56                     this.percentage = 0;
57                 }
58             }
59         }
60     }).mount("#app")
61 </script>
62 </html>

```

50%

2. 数组语法

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>数组语法</title>
9      <style>
10         .box {
11             width: 100px;
12             height: 100px;
13             border: 1px solid black;
14         }
15     </style>
16 </head>
17
18 <body>
19     <div id="app">
20         <div class="box" :style="[1<2?obj:null,obj2]">123123123</div>
21         <button type="button" @click="change">改变样式</button>
22     </div>
23 </body>
24 <script src="../js/vue.global.js"></script>
25 <script>
26     Vue.createApp({
27         data() {
28             return {
29                 obj: {
30                     backgroundColor: "red",
31                     borderRadius: "50%"
32                 },
33                 obj2:{
34
35                 }
36             }
37         },
38         methods: {
39             change() {
40                 this.obj.width = "200px";
41                 this.obj.height = "200px";
42                 this.obj2.transform="rotateZ(45deg)";
43             }
44         }
45     }).mount("#app")
46 </script>
47
48 </html>
```

vue的过滤器

vue3已废弃,但是在vue2里面是非常常用的一种情况

在vue2里面，过滤器就是数据格式化工具，或者数据需要二次处理的时候就使用它

全局过滤器

顾名思义就是可以在全局使用的，它的语法格式如下

```
1  Vue.filter("过滤器名子", function(参数){
2      //这里的第一个参数就是要格式化处理的值
3      //过滤器必须要返回一个值
4  })
```

在使用过滤器的时候我们如下所示

```
1  <h2>{{content | 过滤器}}</h2>
```

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>过滤器</title>
9  </head>
10
11 <body>
12     <div id="app">
13         <h2 v-for="(item,index) in list">
14             {{item | formatDate}}
15         </h2>
16     </div>
17 </body>
18 <script src="../js/vue.js"></script>
19 <script>
20     // 编写一个过滤器【数据格式化工具】
21     //这个参数就是你要格式化的值
22     Vue.filter("formatDateTime", function(d) {
23         let _d = new Date(d);
24         let year = _d.getFullYear();
25         let month = _d.getMonth() + 1;
26         let day = _d.getDay();
27         let hour = _d.getHours();
28         let second = _d.getSeconds();
29         let minute = _d.getMinutes();
30         return [year, month, day].join("-") + " " + [hour,minute,second].join(":");
31     });
32
33     new Vue({
34         el: "#app",
35         data: {
```



```

36         list: [
37             new Date(),
38             1667356029627,
39             "2020-12-2 12:04:14"
40         ]
41     }
42 })
43 </script>
44
45 </html>
46 <!--
47     我现在希望这个日期全部以 xxxx-xx-xx xx:xx:xx
48 -->

```

正同这个就是使用了过滤器去截取字符串，并且过滤器还可以接收参数

```

1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta http-equiv="X-UA-Compatible" content="IE=edge">
7      <meta name="viewport" content="width=device-width, initial-scale=1.0">
8      <title>过滤器</title>
9      <style>
10         .box {
11             width: 300px;
12             font-size: 22px;
13             border: 2px solid red;
14             padding: 10px;
15         }
16     </style>
17 </head>
18
19 <body>
20     <div id="app">
21         <div class="box">
22             {{content | txtEllipsis(40)}}
23         </div>
24         <div class="box">
25             {{content2 | txtEllipsis(40)}}
26         </div>
27     </div>
28 </body>
29 <script src="./js/vue.js"></script>
30 <script>
31     Vue.filter("txtEllipsis", function(str, maxLength = 20) {
32         if (str.length > maxLength) {
33             return str.substr(0, maxLength) + `...`;
34         } else {
35             return str;
36         }
37     })
38
39     new Vue({
40         el: "#app",

```

```

41     data: {
42         content: "10月31日，中共中央总书记、国家主席习近平在人民大会堂同越共中央总书记阮富仲举行会谈。中越双方谈了哪些话题，释放了哪些信号，一起来看康辉在现场发回的报道。",
43         content2: "中国疫情防护取得良好效果"
44     }
45 })
46 </script>
47
48 </html>
49 <!--
50     我现在希望这个日期全部以 xxxx-xx-xx xx:xx:xx
51     -->

```

提起港星温碧霞的名字，总让人不禁联想起21年前，她在电视剧《封神榜》中饰演的妲己...

中国疫情防护取得良好效果

局部过滤器

```

1
2 <body>
3     <div id="app">
4         <div class="box">
5             {{content | txtEllipsis}}
6         </div>
7         <div class="box">
8             {{content2 | txtEllipsis}}
9         </div>
10    </div>
11 </body>
12 <script src="../js/vue.js"></script>
13 <script>
14     new Vue({
15         el: "#app",
16         data: {
17             content: "提起港星温碧霞的名字，总让人不禁联想起21年前，她在电视剧《封神榜》中饰演的妲己一角，精湛的演技、对角色恰到好处的拿捏以及出尘的容貌气质，一度被无数观众奉为荧幕女神，同时也承载了80、90后两代人的青春记忆",
18             content2: "中国疫情防护取得良好效果"
19         },
20         filters: {
21             txtEllipsis(str, maxLength = 20) {
22                 if (str.length > maxLength) {
23                     return str.substr(0, maxLength) + "...";
24                 } else {
25                     return str;
26                 }
27             }
28         }

```

```
29     })
30   </script>
```

这一种的定义方式我们叫局部过滤器，在这一种过滤上面，它除了定义方式不一样以外，其它的都是一样的

那为什么会叫局部，为什么会叫全局？先不要慌，我们后面会讲到组件

思考：vue3为什么会废弃过滤器

```
1  <!DOCTYPE html>
2  <html lang="zh">
3
4  <head>
5    <meta charset="UTF-8">
6    <meta http-equiv="X-UA-Compatible" content="IE=edge">
7    <meta name="viewport" content="width=device-width, initial-scale=1.0">
8    <title>vue3实现过滤器</title>
9    <style>
10      .box {
11        width: 300px;
12        border: 2px solid red;
13        padding: 5px;
14        font-size: 22px;
15      }
16    </style>
17  </head>
18
19  <body>
20    <div id="app">
21      <div class="box" v-html="txtEllipsis(content,30)">
22      </div>
23    </div>
24  </body>
25  <script src="./js/vue.global.js"></script>
26  <script>
27    Vue.createApp({
28      data() {
29        return {
30          content: "提起港星温碧霞的名字，总让人不禁联想起21年前，她在电视剧《封神榜》
31          中饰演的妲己一角，精湛的演技、对角色恰到好处的拿捏以及出尘的容貌气质，一度被无数观众奉为荧幕女
32          神，同时也承载了80、90后两代人的青春记忆"
33        }
34      },
35      methods: {
36        txtEllipsis(str, maxLength = 20) {
37          if(str.length>maxLength){
38            return str.substr(0,maxLength)+`...<a href="#">更多</a>`;
39          }
40          else{
41            return str;
42          }
43        }
44      }
45    }).mount("#app")
```

```
46     </script>
47
48     </html>
```

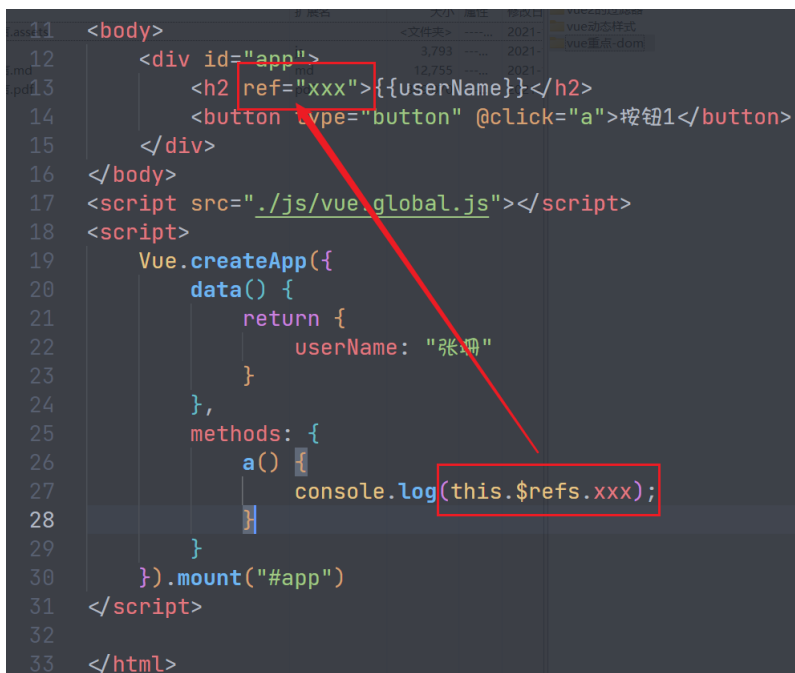
通过上面的代码我们可以看到，vue3没有过滤器了，但是仍然可以实现过滤器的效果【过滤器的本质就是一个函数，所以我们可以直接将这个函数定义在 `methods` 里面就行了】，同时vue3里面去实现这种功能的时候比vue2还要更好一些

vue的DOM操作

在vue框架的开发里面，我们是不推荐进行DOM的操作的，但是并不代表vue不能进行DOM的选取与操作，它是完全可以的

在进行DOM操作的时候主要方式有以下几种

1. 最原始的方式获取DOM元素 `document.xxxxx` 方式
这种情况的使用要注意，不要退无可退，不能使用这种方式
2. 可以利用事件对象当中的 `currentTarget` 事件绑定者与 `target` 的事件触发者来找某一个DOM对象
这个东西也有缺点，它只能找自己或以自己为中心的元素，不能找别人
3. 通过 `ref` 引用去查找。这是vue内部推荐的方式



```
11 <body>
12   <div id="app">
13     <h2 ref="xxx">{{userName}}</h2>
14     <button type="button" @click="a">按钮1</button>
15   </div>
16 </body>
17 <script src="./js/vue.global.js"></script>
18 <script>
19   Vue.createApp({
20     data() {
21       return {
22         userName: "张珊"
23       }
24     },
25     methods: {
26       a() {
27         console.log(this.$refs.xxx);
28       }
29     }
30   }).mount("#app")
31 </script>
32
33 </html>
```

严重的注意事项：非常容易错的点

每一个注意事项，ref相同的时候

```
<body>
  <div id="app">
    <h2 ref="xxx">{{userName}}</h2>
    <h3 ref="xxx">{{age}}</h3>
    <button type="button" @click="a">按钮1</button>
  </div>
</body>
<script src="./js/vue.global.js"></script>
<script>
  Vue.createApp({
    data() {
      return {
        userName: "张珊",
        age: 18
      }
    },
    methods: {
      a() {
        console.log(this.$refs.xxx);
      }
    }
  }).mount("#app")
</script>
```

当ref值相同的时候，取的是最后一次出现的

结果如下

<h3>18</h3>

通过v-for生成的ref每一个都是一个数组

```
<body>
  <div id="app">
    <h2 v-for="(item,index) in list" :ref="'xxx'+index">{{item}}----{{index}}</h2>
    <button type="button" @click="a">按钮1</button>
  </div>
</body>
<script src="./js/vue.global.js"></script>
<script>
  Vue.createApp({
    data() {
      return {
        list: [
          "张三", "李四", "王五", "赵六"
        ]
      }
    },
    methods: {
      a() {
        // console.log(this.$refs.xxx);
        console.log(this.$refs);
      }
    }
  }).mount("#app")
</script>
</html>
```

通过 v-for 生成的 ref 值，最终会形成一个数组

- ▼ Proxy {xxx0: Array(1), xxx1: Array(1), xxx2: Array(1), xxx3: ,
 - ▶ [[Handler]]: Object
 - ▼ [[Target]]: Object
 - ▶ xxx0: [h2]
 - ▶ xxx1: [h2] **每一个都是一个数组**
 - ▶ xxx2: [h2]
 - ▶ xxx3: [h2]
 - ▶ [[Prototype]]: Object
 - ▶ [[IsRevoked]]: false

所以在通过 `v-for` 去生成 `ref` 的时候没有必要去定义不同的 `ref`，可以直接定义成相同的 `ref`，它会自动帮我们形成一个数组，如下所示

```

<div id="app">
  <h2 v-for="(item,index) in list" ref="xxx">{{item}}----{{index}}</h2>
  <button type="button" @click="a">按钮1</button>
</div>

```

- ▼ Proxy {xxx: Array(4)} ⓘ
 - ▶ [[Handler]]: Object
 - ▼ [[Target]]: Object
 - ▼ xxx: Array(4)
 - ▶ 0: h2
 - ▶ 1: h2
 - ▶ 2: h2
 - ▶ 3: h2
 - length: 4
 - ▶ [[Prototype]]: Array(0)
 - ▶ [[Prototype]]: Object
 - ▶ [[IsRevoked]]: false