

# 03-指令补充+样式绑定+计算属性+侦听器

## 一、指令补充

### 1. 指令修饰符

#### 1.1 什么是指令修饰符？

所谓指令修饰符就是让指令的 功能更强大，书写更便捷

#### 1.2 分类

##### 1.2.1 按键修饰符

- @keydown.enter: 当enter键按下时触发
- @keyup.enter: 当enter键抬起时触发

代码演示:

```
1 <script setup>
2   // 处理按键函数
3   const onKeyDown = () => {
4     console.log('onKeyDown')
5   }
6 </script>
7 <input type="text" @keydown.enter="onKeyDown" />
```

##### 1.2.2 事件修饰符

1. @事件名.stop —> 阻止冒泡
2. @事件名.prevent —> 阻止默认行为
3. @事件名.stop.prevent —> 可以连用 即阻止事件冒泡也阻止默认行为

```
1 <script setup>
2
3   // p标签的点击事件
4   const onPClick = () => {
5     console.log('onPClick')
6   }
7
8   // div标签的点击事件
```

```

9   const onDivClick = () => {
10     console.log('onDivClick')
11   }
12 </script>
13
14 <template>
15   <!-- 事件修饰符 -->
16   <div @click="onDivClick">
17     <!-- .prevent: 阻止默认行为 -->
18     <a href="https://baidu.com" @click.prevent>百度一下</a>
19     <!-- .stop: 阻止冒泡，同名事件不会向上级传递 -->
20     <p @click.stop="onPClick"></p>
21
22     <!-- 修饰符的链式调用：表明两个同时阻止 -->
23     <a href="https://baidu.com" @click.stop.prevent>百度一下</a>
24   </div>
25 </template>
26
27 <style>
28   div {
29     width: 400px;
30     height: 200px;
31     background: plum;
32   }
33   div a {
34     display: block;
35     width: 100px;
36     text-decoration: none;
37     background: tomato;
38     text-align: center;
39     color: #fff;
40   }
41   div p {
42     width: 200px;
43     height: 100px;
44     background: rebeccapurple;
45   }
46 </style>

```

### 1.2.3 v-model修饰符

- v-model.trim —> 去除首尾空格
- v-model.number —> 尝试用parseFloat()转数字
- v-model.lazy —> 失去焦点时同步数据，而不是输入时同步数据

```

1 <script setup>
2   import { reactive } from 'vue'
3   const goods = reactive({
4     name: '',
5     price: ''
6   })
7 </script>
8
9 <template>
10  <div>
11    <!-- .lazy修饰符 -->
12    <!-- 名称:
13    <input
14      type="text"
15      v-model.lazy="goods.name" /><br /><br /> -->
16
17    <!-- .trim修饰符 -->
18    名称:
19    <input
20      type="text"
21      v-model.trim="goods.name" /><br /><br />
22
23    <!-- .number修饰符 -->
24    价格:
25    <input
26      type="text"
27      v-model.number="goods.price" /><br /><br />
28  </div>
29 </template>
30
31 <style scoped></style>

```

## 1.3 总结

### 1. 如何监听回车键？

答: 1、@keydown.enter -> 回车按下

2、@keyup.enter -> 回车抬起

### 2. 如何阻止默认行为、阻止冒泡？

答: 1、@click.prevent -> 阻止默认行为

2、@click.stop -> 阻止冒泡

3、@click.prevent.stop -> 可以链式调用, 二者都阻止

### 3. v-model的3个修饰符的作用是什么？

答: 1、.lazy -> 失去焦点再同步

2、.trim -> 去除首尾空格

3、.number -> 尝试转数字

## 2. v-model用在其他表单元素上

### 2.1 讲解内容

常见的表单元素都可以用 v-model 绑定关联, 作用是可以快速 获取 或 设置 表单元素的值  
它会根据 控件类型 自动选取 正确的属性 来更新元素

- |        |                             |  |
|--------|-----------------------------|--|
| 1 输入框  | <code>input:text</code>     | —> <code>value</code>                        |
| 2 文本域  | <code>textarea</code>       | —> <code>value</code>                        |
| 3 下拉菜单 | <code>select</code>         | —> <code>value</code>                        |
| 4 单选框  | <code>input:radio</code>    | —> <code>value</code>                        |
| 5 复选框  | <code>input:checkbox</code> | —> <code>checked</code> / <code>value</code> |

### 2.2 代码准备

```
1 <script setup>
2   import { ref } from 'vue'
3   // 自我介绍
4   const intro = ref('')
5
6   // 收集城市
7   const city = ref('SH')
8
9   // 收集血型
10  const blood = ref('ab')
11
12  // 是否同意用户协议
13  const isAgree = ref(false)
14
15  // 收集爱好
16  const hobby = ref(['ZQ', 'PB'])
17 </script>
18
19 <template>
20   <div>
21     <!-- 文本域 -->
22     <textarea
23       v-model="intro"
24       cols="30"
```

```
25     rows="4"
26     placeholder="请输入自我介绍"></textarea>
27
28 <br />
29 <br />
30 <!-- 下菜单 -->
31 <select v-model="city">
32     <option value="BJ">北京</option>
33     <option value="SH">上海</option>
34     <option value="SZ">深圳</option>
35     <option value="HZ">杭州</option>
36 </select>
37 <br />
38 <br />
39 <!-- 单选框：多个当中只能选择一个，需要给单选框手动添加 value 属性 -->
40 <input
41     type="radio"
42     value="a"
43     v-model="blood" />A
44 <input
45     type="radio"
46     value="b"
47     v-model="blood" />B
48 <input
49     type="radio"
50     value="ab"
51     v-model="blood" />AB
52 <input
53     type="radio"
54     value="o"
55     v-model="blood" />O
56
57 <br />
58 <br />
59 <input
60     type="checkbox"
61     v-model="isAgree" />是否同意用户协议
62
63 <br />
64 <br />
65 <input
66     v-model="hobby"
67     type="checkbox"
68     value="LQ" />篮球
69 <input
70     v-model="hobby"
71     type="checkbox"
```

```

72     value="ZQ" />足球
73     <input
74         v-model="hobby"
75         type="checkbox"
76         value="YMQ" />羽毛球
77     <input
78         v-model="hobby"
79         type="checkbox"
80         value="PPQ" />乒乓球
81     <br />
82     <input
83         v-model="hobby"
84         type="checkbox"
85         value="PB" />跑步
86     <input
87         v-model="hobby"
88         type="checkbox"
89         value="YY" />游戏
90     <input
91         v-model="hobby"
92         type="checkbox"
93         value="PLT" />普拉提
94     <input
95         v-model="hobby"
96         type="checkbox"
97         value="LDW" />拉丁舞
98 </div>
99 </template>
100
101 <style scoped></style>

```

## 2.3 总结

### 1. v-model如何收集下拉列表的值？

答: v-model写在select上, 关联是选中option的`value`

### 2. v-model如何收集单选框的值？

答: 给单选框添加value属性, v-model收集选中单选框的value

### 3. v-model作用在复选框上组要注意什么？

答: 1、一个复选框, v-model绑定 布尔值, 关联 checked 属性

2、一组复选框, v-model绑定 数组, 关联 value 属性, 给复选框 手动添加 value

## 二、样式绑定

## 1. 基本介绍

### 1概述

为了方便开发者进行样式控制,Vue 扩展了 v-bind 的语法, 可以针对 **class 类名** 和 **style 行内样式** 两个属性进行控制, 进而通过数据控制元素的样式

## 2、分类

### 2.1、操作class

### 2.2、操作style

## 2、操作class

### 1、语法

```
:class = "三元表达式 / 对象"
```

### 2、三元表达式

```
1 <p :class="条件 ? '类名1' : '类名2'"></p>
```

### 3、对象语法

当class动态绑定的是**对象**时, **键就是类名, 值就是布尔值**, 如果值是**true**, 就添加这个类, 否则删除这个类

```
1 <p class="box" :class="{ 类名1: 布尔值1, 类名2: 布尔值2 }"></p>
```

适用场景: 一个类名, 来回切换

## 4、静态class与动态class共存

静态class与动态class共存可以**共存**, 二者会**合并**

```
1 <p class="item" :class="条件 ? '类名1' : '类名2'"></p>
```

## 4、代码演示

```

1 <script setup>
2   import { ref } from 'vue'
3
4   // 是否处于激活
5   const isActive = ref(true)
6 </script>
7
8 <template>
9   <div>
10     <!-- 1. 三元绑定 -->
11     <p :class="isActive ? 'active' : ''">Active1</p>
12     <!-- 2. 对象绑定 -->
13     <p :class="{ active: isActive }">Active2</p>
14     <!-- 3. 静态class与动态class共存 -->
15     <p class="item" :class="{ active: isActive }">Active3</p>
16   </div>
17 </template>
18
19 <style>
20   .active {
21     color: red;
22   }
23 </style>

```

## 5、总结

### 1. 如何通过class动态控制元素的样式？

答：1、 `:class="三元表达式"`

2、 `:class="{ 类名: 布尔值 }"`，布尔值为true则添加类名；否则移除

### 2. 静态和动态class可以共存吗？

答：可以 `共存`，二者会合并

## 3、京东秒杀-tab栏切换导航高亮

### 1、需求

当我们点击哪个tab页签时，哪个tab页签就高亮

京东秒杀

每日特价

品类秒杀

### 2、静态代码(可复制)



```
1 <script setup>
2   // tabs 列表
3   const tabs = [
4     { id: 1, name: '京东秒杀' },
5     { id: 2, name: '每日特价' },
6     { id: 3, name: '品类秒杀' }
7   ]
8 </script>
9
10 <template>
11   <div>
12     <ul>
13       <li><a class="active" href="#">京东秒杀</a></li>
14     </ul>
15   </div>
16 </template>
17
18 <style>
19   * {
20     margin: 0;
21     padding: 0;
22   }
23   ul {
24     display: flex;
25     border-bottom: 2px solid #e01222;
26     padding: 0 10px;
27   }
28   li {
29     width: 100px;
30     height: 50px;
31     line-height: 50px;
32     list-style: none;
33     text-align: center;
34   }
35   li a {
36     display: block;
37     text-decoration: none;
38     font-weight: bold;
39     color: #333333;
40   }
41   li a.active {
42     background-color: #e01222;
43     color: #fff;
44   }
45 </style>
```

### 3、思路

1. 基于数据，动态渲染tab (v-for)
2. 准备一个下标 记录高亮哪一个 tab
3. 基于下标动态切换class的类名

### 4、完整代码

```
1 <script setup>
2   import { ref } from 'vue'
3
4   // tabs 列表
5   const tabs = [
6     { id: 1, name: '京东秒杀' },
7     { id: 2, name: '每日特价' },
8     { id: 3, name: '品类秒杀' }
9   ]
10  // 当前高亮的下标，默认是第一个，下标为 0
11  const currentIndex = ref(0)
12
13 </script>
14
15 <template>
16   <div>
17     <ul>
18       <li
19         v-for="(item, index) in tabs"
20         :key="item.id"
21         @click="currentIndex = index">
22         <a
23           :class="{ active: currentIndex === index }"
24           href="#"
25           >{{ item.name }}</a>
26       >
27     </li>
28   </ul>
29 </div>
30 </template>
31
32 <style>
33   * {
34     margin: 0;
35     padding: 0;
36   }
37   ul {
```

```
38     display: flex;
39     border-bottom: 2px solid #e01222;
40     padding: 0 10px;
41 }
42 li {
43     width: 100px;
44     height: 50px;
45     line-height: 50px;
46     list-style: none;
47     text-align: center;
48 }
49 li a {
50     display: block;
51     text-decoration: none;
52     font-weight: bold;
53     color: #333333;
54 }
55 li a.active {
56     background-color: #e01222;
57     color: #fff;
58 }
59 </style>
60
```

## 4、操作style

### 1、语法

```
1 <div class="box" :style="{ CSS属性名1: CSS属性值, CSS属性名2: CSS属性值 }"></div>
```

### 2、代码演示

```
1 <script setup>
2   import { reactive } from 'vue'
3
4   // 行内样式对象
5   const styleObj = reactive({
6     color: '#fff',
7     backgroundColor: 'purple'
8   })
9 </script>
10 <template>
11   <div>
```

```
12     <p :style="styleObj">Some Text...</p>
13 </div>
14 </template>
15 <style></style>
```

### 3、进度条案例

```
1 <script setup></script>
2
3 <template>
4   <div class="progress">
5     <div class="inner" style="width: 50%">
6       <span>50%</span>
7     </div>
8   </div>
9   <button>设置25%</button>
10  <button>设置50%</button>
11  <button>设置75%</button>
12  <button>设置100%</button>
13 </template>
14
15 <style>
16   .progress {
17     height: 25px;
18     width: 400px;
19     border-radius: 15px;
20     background-color: #272425;
21     border: 3px solid #272425;
22     box-sizing: border-box;
23     margin-bottom: 30px;
24   }
25   .inner {
26     height: 20px;
27     border-radius: 10px;
28     text-align: right;
29     position: relative;
30     background-color: #409eff;
31     background-size: 20px 20px;
32     box-sizing: border-box;
33     transition: all 1s;
34   }
35   .inner span {
36     position: absolute;
37     right: -25px;
38     bottom: -25px;
```

```
39   }  
40 </style>
```

#### 4、完整代码

```
1 <script setup>  
2   import { ref } from 'vue'  
3  
4   // 份数  
5   const x = ref(0)  
6 </script>  
7  
8 <template>  
9   <div class="progress">  
10     <div  
11       class="inner"  
12       :style="{ width: `${(x / 4) * 100}%` }">  
13       <span>{{ (x / 4) * 100 }}%</span>  
14     </div>  
15   </div>  
16   <button @click="x = 1">设置25%</button>  
17   <button @click="x = 2">设置50%</button>  
18   <button @click="x = 3">设置75%</button>  
19   <button @click="x = 4">设置100%</button>  
20 </template>  
21  
22 <style>  
23   .progress {  
24     height: 25px;  
25     width: 400px;  
26     border-radius: 15px;  
27     background-color: #272425;  
28     border: 3px solid #272425;  
29     box-sizing: border-box;  
30     margin-bottom: 30px;  
31   }  
32   .inner {  
33     height: 20px;  
34     border-radius: 10px;  
35     text-align: right;  
36     position: relative;  
37     background-color: #409eff;  
38     background-size: 20px 20px;  
39     box-sizing: border-box;  
40     transition: all 1s;
```

```

41   }
42   .inner span {
43     position: absolute;
44     right: -25px;
45     bottom: -25px;
46   }
47 </style>

```

## 5、总结

### 1. 如何给元素动态绑定style?

答: `:style="{ 属性名1: 表达式1, 属性名2: 表达式2, ... }"`

## 三、计算属性

### 1. 基本使用

#### 1.1 概念

基于现有的数据，计算出来的新数据;当现有的数据变化，会自动重新计算。

#### 1.2 语法

##### 1. 使用 **computed** 函数，计算得到一个新数据进行展示

```

1 const 新数据 = computed(() => {
2   // some code ...
3   return 结果
4 })
5
6 // 商品列表(原始数据)
7 const goodsList = ref([
8   { id: 1, name: '篮球', num: 1 },
9   { id: 2, name: '玩具', num: 3 },
10  { id: 3, name: '书籍', num: 2 }
11 ])

```

#### 1.3 案例

比如我们可以使用计算属性实现下面这个业务场景



## 1.4 静态代码(可复制)

```
1 <script setup>
2
3   import { ref } from 'vue'
4
5   // 商品列表(原始数据)
6   const goodsList = ref([
7     { id: 1, name: '篮球', num: 1 },
8     { id: 2, name: '玩具', num: 3 },
9     { id: 3, name: '书籍', num: 2 }
10  ])
11
12 </script>
13 <template>
14   <h3>比特人的礼物清单</h3>
15   <table>
16     <tr>
17       <th>名字</th>
18       <th>数量</th>
19     </tr>
20     <tr
21       v-for="item in goodsList"
22       :key="item.id">
23       <td>{{ item.name }}</td>
24       <td>{{ item.num }}个</td>
25     </tr>
26   </table>
27   <!-- 目标: 统计求和, 展示礼物总数 -->
28   <p>礼物总数: ? 个</p>
29 </template>
30
31 <style>
32   table {
33     width: 350px;
34     border: 1px solid #333;
35   }
36   table th,
37   table td {
38     border: 1px solid #333;
39   }
40   table td {
41     text-align: center;
42   }
43 </style>
```

## 1.5 完整代码

```
1 <script setup>
2   import { computed, ref } from 'vue'
3   // 商品列表(原始数据)
4   const goodsList = ref([
5     { id: 1, name: '篮球', num: 1 },
6     { id: 2, name: '玩具', num: 3 },
7     { id: 3, name: '书籍', num: 2 }
8   ])
9
10  // 由于这里只有原始数据, 没有直接给我们提供商品总数量
11  // 但是我们可以基于现有的 goodsList 计算得到总数量
12  // 那么推荐把总数量声明为 计算属性
13  const totalNum = computed(() => {
14    return goodsList.value.reduce((prev, item) => prev + item.num, 0)
15  })
16 </script>
17
18 <template>
19   <h3>比特人的礼物清单</h3>
20   <table>
21     <thead>
22       <tr>
23         <th>名字</th>
24         <th>数量</th>
25       </tr>
26     </thead>
27     <tbody>
28       <tr
29         v-for="item in goodsList"
30         :key="item.id">
31         <td>{{ item.name }}</td>
32         <td>{{ item.num }}</td>
33       </tr>
34     </tbody>
35   </table>
36   <p>礼物总数: {{ totalNum }} 个</p>
37 </template>
38
39 <style>
40   table {
41     width: 350px;
42     border: 1px solid #333;
43   }
44   table th,
```



```
45   table td {
46     border: 1px solid #333;
47   }
48   table td {
49     text-align: center;
50   }
51 </style>
```

## 1.6 注意

1. 计算属性必须有返回值
2. 使用和 ref/reactive 数据一样, 可用于插值, 也可配合指令

## 1.7 总结

1. 何时用计算属性?

答: 基于现在数据计算得到新数据的时候

2. 语法?

```
1 const 新数据 = computed(() => {
2   return 计算结果
3 })
```

## 2. 计算属性 VS 普通函数

### 2.1 计算属性

#### 2.1.1 作用

封装了一段对于基于现有数据, 计算求得一个**新数据**

#### 2.1.2 语法

1. 写在computed函数中, 必须返回
2. 作为属性, 直接使用
  - js中获取计算属性: 计算属性.value
  - 模板中使用计算属性: {{ 计算属性 }} 或 配合指令

### 2.2 普通函数

#### 2.2.1 作用

封装一段js代码, 用于调用以**处理业务逻辑**。

## 2.2.2 语法

### 1. 直接在script下定义

### 2. 作为函数调用

- js中调用：函数名(实参列表)
- 模板中调用 {{ 函数名 (实参列表) }} 或者 @事件名= “方法名(实参列表)”

### 3. 计算属性的优势

- 缓存特性(提升性能)，函数没有缓存特性
- 计算属性会对计算出来的结果缓存, 再次使用直接读取缓存
- 只有依赖项变了, 才会自动重新计算, 并再次缓存新数据

```
1 <script setup>
2   import { ref, computed } from 'vue'
3
4   // 商品列表(原始数据)
5   const goodsList = ref([
6     { id: 1, name: '篮球', num: 1 },
7     { id: 2, name: '玩具', num: 3 },
8     { id: 3, name: '书籍', num: 2 }
9   ])
10
11   // 计算总数
12   const totalNum = computed(() => {
13     return goodsList.value.reduce((prev, item) => prev+ item.num, 0)
14   })
15 </script>
16 <template>
17   <div id="app">
18     <h3>比特人的礼物清单</h3>
19     <table>
20       <tr>
21         <th>名字</th>
22         <th>数量</th>
23       </tr>
24       <tr
25         v-for="item in goodsList"
26         :key="item.id">
27         <td>{{ item.name }}</td>
28         <td>{{ item.num }}个</td>
29       </tr>
30     </table>
31
32     <!-- 目标：统计求和，求得礼物总数 -->
```

```
33     <p>礼物总数: {{ totalNum }} 个</p>
34 </div>
35 </template>
36
37 <style>
38   #app {
39     width: 240px;
40     margin: 100px auto;
41   }
42   table {
43     border: 1px solid #000;
44     text-align: center;
45     width: 240px;
46   }
47   th,
48   td {
49     border: 1px solid #000;
50   }
51   h3 {
52     position: relative;
53   }
54 </style>
```

## 2.3 总结

### 1. 计算属性与普通函数的区别？

- 答：1、计算属性基于依赖 **有缓存特性**，普通函数 **没有缓存**
- 2、当基于现有的数据计算得到新数据，推荐使用计算属性
  - 3、当处理业务逻辑时，推荐使用函数，比如点击事件的处理函数

## 3. 计算属性的完整写法

### 3.1 思考

既然计算属性也是属性，能访问，应该也能修改了？

1. 计算属性默认的写法，只能读(展示数据)，不能"改"
2. 如果要"改" → 需要用计算属性的完整写法

```
// 简易写法
const uname = computed(() => {
  // 计算得到新数据
  return '姬霓太美'
})
```

```
// 完整写法 = get + set
const uname = computed({
  get() {
    // 计算得到新数据
    return '姬霓太美'
  },
  set(val) {
    // 一段修改逻辑
    console.log(val)
  }
})
```

## 3.2 代码演示

```
1 <script setup>
2   import { computed } from 'vue'
3
4   // 完整写法 = get + set
5   const uname = computed({
6     // 使用计算属性时，自动触发get，get内部必须`return`计算结果`
7     get() {
8       // 计算得到新数据
9       return '姬霓太美'
10    },
11    // 给计算属性赋值时，自动触发set，并接收赋予的新值val
12    set(val) {
13      // 一段修改逻辑
14      console.log(val)
15    }
16  })
17 </script>
18 <template>
19   <input type="text" v-model="uname" />
20 </template>
```

## 3.3 总结

### 1. 何时使用计算属性完整写法？

答：当 `修改计算属性` 时，也就是当计算属性配合v-model的时候

### 2. 完整写法的是？

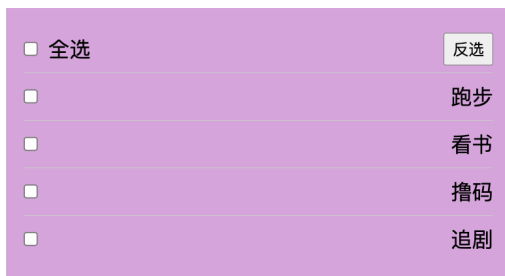
答： `get + set`

```
1 const xxx = computed({
2   get() {
3     return 计算结果
4   },
5   set(val) {
```

```
6
7   }
8 })
```

## 4. 案例 - 全返反选

### 4.1 效果



### 4.2 静态代码(可复制)

```
1 <script setup>
2   import { ref } from 'vue'
3
4   // 计划列表
5   const planList = ref([
6     { id: 12, name: '跑步', done: false },
7     { id: 76, name: '看书', done: false },
8     { id: 31, name: '撸码', done: false },
9     { id: 49, name: '追剧', done: false }
10  ])
11
12 </script>
13
14 <template>
15   <p>
16     <span>
17       <input type="checkbox" id="all" />
18       <label for="all">全选</label>
19     </span>
20     <button>反选</button>
21   </p>
22   <ul>
23     <li>
24       <input type="checkbox" />
25       <span>xxx</span>
26     </li>
27   </ul>
```

```
28 </template>
29
30 <style lang="scss">
31   * {
32     margin: 0;
33     padding: 0;
34   }
35   div {
36     width: 400px;
37     margin: 100px auto;
38     padding: 15px;
39     font-size: 18px;
40     background: plum;
41     p {
42       display: flex;
43       justify-content: space-between;
44       align-items: center;
45       height: 40px;
46       button {
47         padding: 3px 6px;
48       }
49     }
50   }
51
52   ul {
53     list-style: none;
54     li {
55       display: flex;
56       align-items: center;
57       justify-content: space-between;
58       height: 40px;
59       border-top: 1px solid #ccc;
60       span.completed {
61         color: #ddd;
62         text-decoration: line-through;
63       }
64     }
65   }
66
67   input {
68     margin-right: 8px;
69   }
70 </style>
```

## 4.3 完整代码

```
1 <script setup>
2   import { computed, ref } from 'vue'
3   // 计划列表
4   const planList = ref([
5     { id: 12, name: '跑步', done: false },
6     { id: 76, name: '看书', done: false },
7     { id: 31, name: '撸码', done: false },
8     { id: 49, name: '追剧', done: false }
9   ])
10  // 是否全部选中
11  const isAll = computed({
12    // 使用计算属性自动触发 get
13    get() {
14      // every: 检测每一个
15      return planList.value.every((item) => item.done)
16    },
17    // 修改计算属性自动触发 set
18    set(val) {
19      // val: 给计算属性赋予的新值, 在这里就是表全选复选框的状态
20
21      // 遍历 planList 数组, 把每个小选的 done 属性与 val 保持一致即可
22      planList.value.forEach((item) => (item.done = val))
23    }
24  })
25
26  // 反选
27  const onToggle = () => {
28    // 遍历 planList 数组, 对每个对象的 done 属性取反即可
29    planList.value.forEach((item) => (item.done = !item.done))
30  }
31 </script>
32 <template>
33   <p>
34     <span>
35       <input
36         v-model="isAll"
37         type="checkbox"
38         id="all" />
39       <label for="all">全选</label>
40     </span>
41     <button @click="onToggle">反选</button>
42   </p>
43   <ul>
44     <li
45       v-for="item in planList"
46       :key="item.id">
47     <input
```

```

48     type="checkbox"
49     v-model="item.done" />
50     <span :class="{ completed: item.done }">{{ item.name }}</span>
51   </li>
52 </ul>
53 </template>
54
55 <style lang="scss">
56   // npm i sass -D
57   // npm run dev
58   * {
59     margin: 0;
60     padding: 0;
61   }
62   ul {
63     list-style: none;
64   }
65   #app {
66     width: 400px;
67     margin: 100px auto;
68     padding: 15px 18px;
69     background: plum;
70     p {
71       display: flex;
72       justify-content: space-between;
73       align-items: center;
74       height: 40px;
75       border-bottom: 1px solid #ccc;
76       button {
77         padding: 3px 6px;
78       }
79       input {
80         margin-right: 8px;
81       }
82     }
83     ul {
84       li {
85         display: flex;
86         justify-content: space-between;
87         align-items: center;
88         height: 40px;
89         border-bottom: 1px solid #ccc;
90         span.completed {
91           color: #ddd;
92           text-decoration: line-through;
93         }
94       }

```



```
95     }
96   }
97 </style>
```

## 四、侦听器

### 1. 作用

监视数据变化, 执行DOM操作或异步操作

### 2. 语法

#### 1. 监视简单类型

```
1 <script setup>
2   import { ref, watch } from 'vue'
3
4   // 搜索框关键字
5   const keyword = ref('')
6
7   // 监视 keyword, 只要 keyword 的值变了, 就会执行回调函数
8   watch(keyword, (newVal, oldVal) => {
9     // newVal: 新值
10    // oldVal: 旧值
11  })
12
13 </script>
14 <template>
15   <div>
16     <input
17       type="text"
18       v-model="keyword"
19       placeholder="请输入关键字" />
20   </div>
21 </template>
```

#### 2. 监视复杂类型

```
1 <script setup>
2   import { reactive, watch } from 'vue'
3
4   // 表单对象
5   const obj = reactive({
```

```

6     username: '',
7     password: ''
8   })
9
10  // 监视 obj 的变化
11  watch(obj, (newVal, oldVal) => {
12    console.log(newVal, oldVal)
13  })
14 </script>
15 <template>
16   <div>
17     <input
18       type="text"
19       v-model="obj.username"
20       placeholder="请输入用户名" />
21     <br/>
22     <br/>
23     <input
24       type="text"
25       v-model="obj.password"
26       placeholder="请输入密码" />
27   </div>
28 </template>
29

```

### 3. 总结

#### 1. watch的作用是？

答：监视响应式数据的变化, 当数据变了, 针对性的DOM操作或异步操作

## 五、案例成绩管理

### 1. 效果

编号	科目	成绩	操作
1	语文	94	<a href="#">删除</a>
2	数学	57	<a href="#">删除</a>
3	英语	92	<a href="#">删除</a>
总分: 243		平均分: 81	

科目: 
  
分数:

### 2. 静态代码(可复制)

```

1 <script setup>
2   import { ref } from 'vue'
3

```

```

4 // 成绩列表
5 const scoreList = ref([
6   { id: 19, subject: '语文', score: 94 },
7   { id: 27, subject: '数学', score: 59 },
8   { id: 12, subject: '英语', score: 92 }
9 ])
10 </script>
11
12 <template>
13   <div class="score-case">
14     <div class="table">
15       <table>
16         <thead>
17           <tr>
18             <th>编号</th>
19             <th>科目</th>
20             <th>成绩</th>
21             <th>操作</th>
22           </tr>
23         </thead>
24         <tbody>
25           <tr>
26             <td>1</td>
27             <td>语文</td>
28             <td class="red">46</td>
29             <td><a href="#">删除</a></td>
30           </tr>
31           <tr>
32             <td>2</td>
33             <td>英语</td>
34             <td>80</td>
35             <td><a href="#">删除</a></td>
36           </tr>
37         </tbody>
38         <tbody>
39           <tr>
40             <td colspan="5">
41               <span class="none">暂无数据</span>
42             </td>
43           </tr>
44         </tbody>
45
46       <tfoot>
47         <tr>
48           <td colspan="5">
49             <span>总分：246</span>
50             <span style="margin-left: 50px">平均分：79</span>

```

```
51         </td>
52     </tr>
53 </tfoot>
54 </table>
55 </div>
56 <form class="form">
57     <div class="form-item">
58         <div class="label">科目: </div>
59         <div class="input">
60             <input
61                 type="text"
62                 placeholder="请输入科目" />
63         </div>
64     </div>
65     <div class="form-item">
66         <div class="label">分数: </div>
67         <div class="input">
68             <input
69                 type="text"
70                 placeholder="请输入分数" />
71         </div>
72     </div>
73     <div class="form-item">
74         <div class="label"></div>
75         <div class="input">
76             <button class="submit">添加</button>
77         </div>
78     </div>
79 </form>
80 </div>
81 </template>
82 <style>
83     .score-case {
84         width: 1000px;
85         margin: 50px auto;
86         display: flex;
87     }
88     .score-case .table {
89         flex: 4;
90     }
91     .score-case .table table {
92         width: 100%;
93         border-spacing: 0;
94         border-top: 1px solid #ccc;
95         border-left: 1px solid #ccc;
96     }
97     .score-case .table table th {
```

```
98     background: #f5f5f5;
99 }
100 .score-case .table table tr:hover td {
101     background: #f5f5f5;
102 }
103 .score-case .table table td,
104 .score-case .table table th {
105     border-bottom: 1px solid #ccc;
106     border-right: 1px solid #ccc;
107     text-align: center;
108     padding: 10px;
109 }
110 .score-case .table table td.red,
111 .score-case .table table th.red {
112     color: red;
113 }
114 .score-case .table .none {
115     height: 100px;
116     line-height: 100px;
117     color: #999;
118 }
119 .score-case .form {
120     flex: 1;
121     padding: 20px;
122 }
123 .score-case .form .form-item {
124     display: flex;
125     margin-bottom: 20px;
126     align-items: center;
127 }
128 .score-case .form .form-item .label {
129     width: 60px;
130     text-align: right;
131     font-size: 14px;
132 }
133 .score-case .form .form-item .input {
134     flex: 1;
135 }
136 .score-case .form .form-item input,
137 .score-case .form .form-item select {
138     appearance: none;
139     outline: none;
140     border: 1px solid #ccc;
141     width: 200px;
142     height: 40px;
143     box-sizing: border-box;
144     padding: 10px;
```

```
145     color: #666;
146   }
147   .score-case .form .form-item input::placeholder {
148     color: #666;
149   }
150   .score-case .form .form-item .cancel,
151   .score-case .form .form-item .submit {
152     appearance: none;
153     outline: none;
154     border: 1px solid #ccc;
155     border-radius: 4px;
156     padding: 4px 10px;
157     margin-right: 10px;
158     font-size: 12px;
159     background: #ccc;
160   }
161   .score-case .form .form-item .submit {
162     border-color: #069;
163     background: #069;
164     color: #fff;
165   }
166 </style>
```

### 3. 功能描述

#### 3.1 渲染功能

#### 3.2 添加功能

#### 3.3 删除功能

#### 3.4 统计总分，求平均分

#### 3.5 数据持久化

### 4. 思路分析

- 渲染功能 v-for :key v-bind:动态绑定class的样式
- 删除功能 v-on绑定事件，阻止a标签的默认行为
- v-model的修饰符 .trim、.number、判断数据是否为空后 再添加、添加后清空文本框的数据
- 使用计算属性computed 计算总分和平均分的值

### 5. 完整代码

```
1 <script setup>
2   import { computed, reactive, ref, watch } from 'vue'
3
4   // 本地存储 key
5   const SCORE_LIST_KEY = 'score-list-key'
6
7   // 成绩列表
8   const scoreList = ref(
9     // 从本地取值
10    // 有值就用，否则用默认的成绩列表
11    JSON.parse(localStorage.getItem(SCORE_LIST_KEY)) || [
12      { id: 19, subject: '语文', score: 94 },
13      { id: 27, subject: '数学', score: 57 },
14      { id: 12, subject: '英语', score: 92 }
15    ]
16  )
17
18  // 成绩表单
19  const scoreForm = reactive({
20    subject: '', // 科目
21    score: '' // 分数
22  })
23
24  // 总分
25  const totalScore = computed(() => {
26    // reduce 求和
27    return scoreList.value.reduce((prev, item) => prev + item.score, 0)
28  })
29  // 平均分
30  const avgScore = computed(() => {
31    return scoreList.value.length
32      ? totalScore.value / scoreList.value.length
33      : 0
34  })
35
36  // 删除
37  const onDel = (index) => {
38    // 弹窗确认
39    if (window.confirm('确认删除么?')) {
40      scoreList.value.splice(index, 1)
41    }
42  }
43
44  // 添加
45  const onAdd = () => {
46    // 非空校验
```

```

47     if (!scoreForm.subject || !scoreForm.score)
48         return alert('科目或分数不能为空')
49
50     // 添加至数组
51     scoreList.value.push({
52         ...scoreForm,
53         id: Date.now()
54     })
55     // 清空
56     scoreForm.subject = scoreForm.score = ''
57 }
58
59 // 监听 scoreList 的变化
60 watch(
61     scoreList,
62     (newVal) => {
63         // 存本地
64         localStorage.setItem(SCORE_LIST_KEY, JSON.stringify(newVal))
65     },
66     {
67         // 深度监视
68         deep: true
69     }
70 )
71 </script>
72
73 <template>
74     <div class="score-case">
75         <div class="table">
76             <table>
77                 <thead>
78                     <tr>
79                         <th>编号</th>
80                         <th>科目</th>
81                         <th>成绩</th>
82                         <th>操作</th>
83                     </tr>
84                 </thead>
85                 <tbody v-if="scoreList.length">
86                     <tr
87                         v-for="(item, index) in scoreList"
88                         :key="item.id">
89                         <td>{{ index + 1 }}</td>
90                         <td>{{ item.subject }}</td>
91                         <td :class="{ red: item.score < 60 }">{{ item.score }}</td>
92                         <td>
93                             <a

```



```
94         href="#"
95         @click="onDel(index)"
96         >删除</a>
97     >
98 </td>
99 </tr>
100 </tbody>
101 <tbody v-else>
102     <tr>
103         <td colspan="5">
104             <span class="none">暂无数据</span>
105         </td>
106     </tr>
107 </tbody>
108
109 <tfoot>
110     <tr>
111         <td colspan="5">
112             <span>总分: {{ totalScore }}</span>
113             <span style="margin-left: 50px">平均分: {{ avgScore }}</span>
114         </td>
115     </tr>
116 </tfoot>
117 </table>
118 </div>
119 <form class="form">
120     <div class="form-item">
121         <div class="label">科目: </div>
122         <div class="input">
123             <input
124                 type="text"
125                 placeholder="请输入科目"
126                 v-model.trim="scoreForm.subject" />
127         </div>
128     </div>
129     <div class="form-item">
130         <div class="label">分数: </div>
131         <div class="input">
132             <input
133                 type="number"
134                 placeholder="请输入分数"
135                 v-model.number="scoreForm.score" />
136         </div>
137     </div>
138     <div class="form-item">
139         <div class="label"></div>
140         <div class="input">
```

```
141         <button
142             class="submit"
143             @click.prevent="onAdd">
144             添加
145         </button>
146     </div>
147 </div>
148 </form>
149 </div>
150 </template>
151 <style>
152     .score-case {
153         width: 1000px;
154         margin: 50px auto;
155         display: flex;
156     }
157     .score-case .table {
158         flex: 4;
159     }
160     .score-case .table table {
161         width: 100%;
162         border-spacing: 0;
163         border-top: 1px solid #ccc;
164         border-left: 1px solid #ccc;
165     }
166     .score-case .table table th {
167         background: #f5f5f5;
168     }
169     .score-case .table table tr:hover td {
170         background: #f5f5f5;
171     }
172     .score-case .table table td,
173     .score-case .table table th {
174         border-bottom: 1px solid #ccc;
175         border-right: 1px solid #ccc;
176         text-align: center;
177         padding: 10px;
178     }
179     .score-case .table table td.red,
180     .score-case .table table th.red {
181         color: red;
182     }
183     .score-case .table .none {
184         height: 100px;
185         line-height: 100px;
186         color: #999;
187     }
```

```
188 .score-case .form {
189     flex: 1;
190     padding: 20px;
191 }
192 .score-case .form .form-item {
193     display: flex;
194     margin-bottom: 20px;
195     align-items: center;
196 }
197 .score-case .form .form-item .label {
198     width: 60px;
199     text-align: right;
200     font-size: 14px;
201 }
202 .score-case .form .form-item .input {
203     flex: 1;
204 }
205 .score-case .form .form-item input,
206 .score-case .form .form-item select {
207     appearance: none;
208     outline: none;
209     border: 1px solid #ccc;
210     width: 200px;
211     height: 40px;
212     box-sizing: border-box;
213     padding: 10px;
214     color: #666;
215 }
216 .score-case .form .form-item input::placeholder {
217     color: #666;
218 }
219 .score-case .form .form-item .cancel,
220 .score-case .form .form-item .submit {
221     appearance: none;
222     outline: none;
223     border: 1px solid #ccc;
224     border-radius: 4px;
225     padding: 4px 10px;
226     margin-right: 10px;
227     font-size: 12px;
228     background: #ccc;
229 }
230 .score-case .form .form-item .submit {
231     border-color: #069;
232     background: #069;
233     color: #fff;
234 }
```

比特就业课