

07-自定义指令+插槽+商品列表案例

一、自定义指令

1. 基本使用

1.1 指令介绍

- 内置指令：v-model、v-for、v-bind、v-on... 这都是Vue给咱们内置的一些指令，可以直接使用
- 自定义指令：同时Vue也支持让开发者，自己注册一些指令。这些指令被称为自定义指令

1.2 作用

封装一段 **公共的DOM操作** 代码，便于复用

1.3 语法

1. 注册

```
1 // main.js 中
2 app.directive('指令名', {
3   // 元素挂载后(成为真实DOM) 自动触发一次
4   // 自动执行
5   mounted(el) {
6     // el: 指令所在的DOM元素
7   }
8 })
```

2. 使用

```
1 <p v-指令名></p>
```

1.4 代码示例

需求：当页面加载时, 让元素获取焦点

```
1 // main.js
2
3 // 注册全局指令
```

```

4 app.directive('focus', {
5   mounted(el) {
6     console.log(el) // input 元素
7     // 聚焦
8     el.focus()
9   }
10 })

```

App.vue

```

1 <script setup></script>
2
3 <template>
4   <div class="app">
5     <input type="text" v-focus />
6   </div>
7 </template>

```

1.5 总结

1. 自定义指令的作用是什么？

答：封装一段公共的 `DOM操作` 的代码

2. 使用自定义指令的步骤是哪两步？

答：先注册、后使用

3. 指令配置选项中的 `mounted` 钩子何时执行？

答：元素 `挂载后` (成为DOM树的一部分时) 自动执行

2. 绑定数据

2.1 需求

实现一个 `color` 指令：传入不同的颜色, 给标签设置文字颜色

2.2 语法

1.在绑定指令时，可以通过“等号”的形式为指令 绑定 具体的参数值

```

1 <div v-color="colorStr">Some Text</div>

```

2.通过 `binding.value` 可以拿到指令值，指令值修改会 触发 **updated** 钩子

```
1 app.directive('指令名', {
2   // 挂载后自动触发一次
3   mounted(el, binding) { },
4   // 数据更新, 每次都会执行
5   updated(el, binding) { }
6 })
```

2.3 代码示例

```
1 // main.js
2 app.directive('color', {
3   mounted(el, binding) {
4     el.style.color = binding.value
5   },
6   updated(el, binding) {
7     el.style.color = binding.value
8   }
9 })
```

App.vue

```
1 <script setup>
2   import { ref } from 'vue'
3   // 颜色
4   const colorStr = ref('red')
5 </script>
6
7 <template>
8   <p v-color="colorStr"></p>
9 </template>
```

2.4 简化形式

对于自定义指令来说, 一个很常见的情况是仅仅需要在 `mounted` 和 `updated` 上实现相同的行
为。这种情况下我们可以直接用一个函数来定义指令, 如下所示:

```
1 app.directive('color', (el, binding) => {
2   // 这会在 `mounted` 和 `updated` 时都调用
3   el.style.color = binding.value
4 })
```

3. 封装v-loading指令

3.1 场景

实际开发过程中，发送请求需要时间，在请求的数据未回来时，页面会处于**空白状态**，用户体验不好

3.2 解决方案

封装一个 v-loading 指令，实现加载中的效果

3.3 分析

1. 本质 loading 效果就是一个蒙层，盖在了盒子上
2. 数据请求中，开启 loading 状态，添加蒙层
3. 数据请求完毕，关闭 loading 状态，移除蒙层

3.4 实现

1. 准备一个 loading 类，通过伪元素定位，设置宽高，实现蒙层
2. 开启关闭 loading 状态（添加移除蒙层），本质只需要添加移除类即可
3. 结合自定义指令的语法进行封装复用

3.5 准备代码

App.vue

```
1 <script setup>
2   import axios from 'axios'
3   import { ref } from 'vue'
4
5   // 新闻列表
6   const newsList = ref([])
7
8   getNewsData()
9
10  // 获取新闻列表
11  async function getNewsData() {
12    // 请求新闻数据
13    const resp = await axios.get('http://localhost:4000/api/news')
14    // 保存数据
15    newsList.value = resp.data.data
16  }
17 </script>
18
19 <template>
```

```
20 <div class="box">
21   <ul>
22     <li
23       v-for="item in newsList"
24       :key="item.id"
25       class="news">
26       <div class="left">
27         <div class="title">{{ item.title }}</div>
28         <div class="info">
29           <span>{{ item.source }}</span>
30           <span>{{ item.time }}</span>
31         </div>
32       </div>
33       <div class="right">
34         
37       </div>
38     </li>
39   </ul>
40 </div>
41 </template>
42
43 <style>
44   .loading:before {
45     z-index: 999;
46     content: '';
47     position: absolute;
48     left: 0;
49     top: 0;
50     width: 100%;
51     height: 100%;
52     background: #fff url('./assets/loading.gif') no-repeat center;
53     background-size: auto;
54   }
55
56   .box {
57     position: relative;
58     width: 800px;
59     min-height: 600px;
60     margin: 10px auto;
61     border-radius: 5px;
62   }
63
64   .news {
65     display: flex;
66     margin: 0 auto;
67     padding: 20px 0;
```

```
67     cursor: pointer;
68   }
69   .news .left {
70     flex: 1;
71     display: flex;
72     flex-direction: column;
73     justify-content: space-between;
74     padding-right: 10px;
75   }
76   .news .left .title {
77     font-size: 20px;
78   }
79   .news .left .info {
80     color: #999999;
81   }
82   .news .left .info span {
83     margin-right: 20px;
84   }
85   .news .right {
86     width: 160px;
87     height: 120px;
88   }
89   .news .right img {
90     width: 100%;
91     height: 100%;
92     object-fit: cover;
93   }
94 </style>
```

```
1 // main.js
2 app.directive('loading', (el, binding) => {
3   if (binding.value) {
4     el.classList.add('loading')
5   } else {
6     el.classList.remove('loading')
7   }
8 })
```

二、插槽

插槽分类

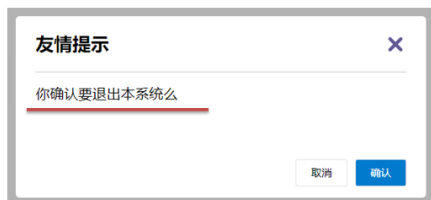
- 默认插槽
- 具名插槽

- 作用域插槽

1. 默认插槽

1.1 作用

让组件内部的一些 **结构** 支持 **自定义**



1.2 需求

将需要多次显示的对话框,封装成一个组件

1.3 问题

组件的内容部分, **不希望写死**, 希望能使用的时候**自定义**。怎么办

1.4 插槽的基本语法

1. 组件内需要定制的结构部分, 改用 `<slot></slot>` 占位
2. 使用组件时, `<MyDialog></MyDialog>` 写成双标签, 包裹结构, 传入替换slot



1.5 代码示例

MyDialog.vue

```
1 <script setup>
2
3 </script>
4 <template>
5   <div class="dialog">
6     <div class="dialog-header">
7       <h3>友情提示</h3>
```

```
8      <span class="close">✕</span>
9    </div>
10
11    <div class="dialog-content">您确定要进行删除操作吗? </div>
12    <div class="dialog-footer">
13      <button>取消</button>
14      <button>确认</button>
15    </div>
16  </div>
17 </template>
18
19 <style scoped>
20 * {
21   margin: 0;
22   padding: 0;
23 }
24 .dialog {
25   width: 470px;
26   height: 230px;
27   padding: 0 25px;
28   background-color: #ffffff;
29   margin: 40px auto;
30   border-radius: 5px;
31 }
32 .dialog-header {
33   height: 70px;
34   line-height: 70px;
35   font-size: 20px;
36   border-bottom: 1px solid #ccc;
37   position: relative;
38 }
39 .dialog-header .close {
40   position: absolute;
41   right: 0px;
42   top: 0px;
43   cursor: pointer;
44 }
45 .dialog-content {
46   height: 80px;
47   font-size: 18px;
48   padding: 15px 0;
49 }
50 .dialog-footer {
51   display: flex;
52   justify-content: flex-end;
53 }
54 .dialog-footer button {
```



```
55     width: 65px;
56     height: 35px;
57     background-color: #ffffff;
58     border: 1px solid #e1e3e9;
59     cursor: pointer;
60     outline: none;
61     margin-left: 10px;
62     border-radius: 3px;
63   }
64   .dialog-footer button:last-child {
65     background-color: #007acc;
66     color: #fff;
67   }
68 </style>
```

App.vue

```
1 <script setup>
2   import MyDialog from './components/MyDialog.vue'
3 </script>
4 <template>
5   <MyDialog />
6 </template>
7
8 <style>
9   body {
10     background-color: #b3b3b3;
11   }
12 </style>
```

1.6 总结

1. 组件内某一部分结构不确定，想要自定义怎么办？

答：使用 插槽 (技术)

2. 插槽的步骤分为哪几步？

答：两步；先占位、后传入

2. 插槽默认值

2.1 问题

通过插槽完成了内容的定制，传什么显示什么, 但是如果不传，则是空白



能否给插槽设置 默认显示内容 呢?

2.2 解决方案

封装组件时, 可以为 `<slot></slot>` 提供默认内容

2.3 语法

在 `<slot></slot>` 标签内, 放置内容, 作为默认内容

```
<div class="dialog">
  <div class="dialog-header">
    <h3>友情提示</h3>
    <span class="close">✕</span>
  </div>

  <div class="dialog-content">
    <!-- 1. 占位, 并提供默认内容 -->
    <slot>我是默认内容</slot>
  </div>

  <div class="dialog-footer">
    <button>取消</button>
    <button>确认</button>
  </div>
</div>
```

2.4 效果

- 使用组件时, 不传, 则会显示slot的默认内容

```
<MyDialog />
```

- 使用组件时, 传了, 则slot整体会被换掉, 从而显示传入的

```
<MyDialog>
  <h4>确定删除么?</h4>
</MyDialog>
```

2.5 代码示例

App.vue

```
1 <script setup>
2   import MyDialog from './components/MyDialog.vue'
3 </script>
4
5 <template>
```

```

6    <!-- 1. 不传 -->
7    <MyDialog />
8
9    <!-- 2. 传了 -->
10   <MyDialog>
11       <h4>确定删除么?</h4>
12   </MyDialog>
13
14 </template>
15
16 <style>
17   body {
18     background-color: #b3b3b3;
19   }
20 </style>

```

MyDialog.vue

```

1  <script setup>
2
3  </script>
4  <template>
5    <div class="dialog">
6      <div class="dialog-header">
7        <h3>友情提示</h3>
8        <span class="close">✕</span>
9      </div>
10
11     <div class="dialog-content">
12       <!-- 插槽默认内容 -->
13       <slot>我是默认内容</slot>
14     </div>
15     <div class="dialog-footer">
16       <button>取消</button>
17       <button>确认</button>
18     </div>
19   </div>
20 </template>
21
22 <style scoped>
23   * {
24     margin: 0;
25     padding: 0;
26   }
27   .dialog {

```

```
28     width: 470px;
29     height: 230px;
30     padding: 0 25px;
31     background-color: #ffffff;
32     margin: 40px auto;
33     border-radius: 5px;
34 }
35 .dialog-header {
36     height: 70px;
37     line-height: 70px;
38     font-size: 20px;
39     border-bottom: 1px solid #ccc;
40     position: relative;
41 }
42 .dialog-header .close {
43     position: absolute;
44     right: 0px;
45     top: 0px;
46     cursor: pointer;
47 }
48 .dialog-content {
49     height: 80px;
50     font-size: 18px;
51     padding: 15px 0;
52 }
53 .dialog-footer {
54     display: flex;
55     justify-content: flex-end;
56 }
57 .dialog-footer button {
58     width: 65px;
59     height: 35px;
60     background-color: #ffffff;
61     border: 1px solid #e1e3e9;
62     cursor: pointer;
63     outline: none;
64     margin-left: 10px;
65     border-radius: 3px;
66 }
67 .dialog-footer button:last-child {
68     background-color: #007acc;
69     color: #fff;
70 }
71 </style>
```

2.6 总结

1. 插槽默认内容有什么用？

答：使用组件不传内容时, 防止出现空白

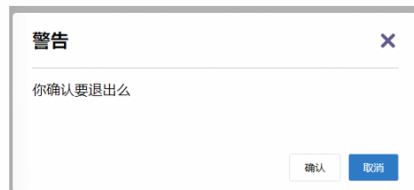
2. 默认内容何时显示？何时不显示？

答：使用组件时, 不传内容,则显示默认内容; 否则显示传递的内容

3. 具名插槽

3.1 需求

一个组件内有多处结构，需要外部传入标签，进行定制



上面的弹框中有**三处不同**，但是**默认插槽只能定制一处内容**，这时怎么办？

3.2 具名插槽语法

- 多个slot使用name属性区分名字
- template配合v-slot:名字 来分发对应标签

3.3 v-slot的简写

v-slot写起来太长，vue给我们提供一个简单写法 **v-slot:** 直接简写为 **#**

3.4 代码示例

MyDialog.vue

```
1 <script setup></script>
2
3 <template>
4   <div class="dialog">
5     <div class="dialog-header">
6       <slot name="header">
7         <h3>温馨提示</h3>
8       </slot>
9       <span class="close">✕</span>
10    </div>
11
12    <div class="dialog-content">
13      <slot>我是主体内容</slot>
14    </div>
```

```
15     <div class="dialog-footer">
16         <slot name="footer">
17             <button>取消</button>
18             <button>确认</button>
19         </slot>
20     </div>
21 </div>
22 </template>
23
24 <style>
25 * {
26     margin: 0;
27     padding: 0;
28 }
29 .dialog {
30     width: 470px;
31     height: 230px;
32     padding: 0 25px;
33     background-color: #ffffff;
34     margin: 40px auto;
35     border-radius: 5px;
36 }
37 .dialog-header {
38     height: 70px;
39     line-height: 70px;
40     font-size: 20px;
41     border-bottom: 1px solid #ccc;
42     position: relative;
43 }
44 .dialog-header .close {
45     position: absolute;
46     right: 0px;
47     top: 0px;
48     cursor: pointer;
49 }
50 .dialog-content {
51     height: 80px;
52     font-size: 18px;
53     padding: 15px 0;
54 }
55 .dialog-footer {
56     display: flex;
57     justify-content: flex-end;
58 }
59 .dialog-footer button {
60     width: 65px;
61     height: 35px;
```

```

62     background-color: #ffffff;
63     border: 1px solid #e1e3e9;
64     cursor: pointer;
65     outline: none;
66     margin-left: 10px;
67     border-radius: 3px;
68 }
69 .dialog-footer button:last-child {
70     background-color: #007acc;
71     color: #fff;
72 }
73 </style>

```

App.vue

```

1  <script setup>
2    import MyDialog from './components/MyDialog.vue'
3  </script>
4
5  <template>
6    <!-- 1. 不传、都显示插槽默认内容 -->
7    <MyDialog />
8
9    <!-- 2. 传了 -->
10   <MyDialog>
11     <template v-slot:header>
12       <h3>友情提示</h3>
13     </template>
14     <template v-slot:default> 请输入正确手机号 </template>
15     <template v-slot:footer>
16       <button>关闭</button>
17     </template>
18   </MyDialog>
19
20   <!-- 3. 传了 -->
21   <MyDialog>
22     <template v-slot:header>
23       <h3>警告</h3>
24     </template>
25     <template v-slot:default> 你确认要退出么? </template>
26     <template v-slot:footer>
27       <button>确认</button>
28       <button>取消</button>
29     </template>
30   </MyDialog>

```

```
31 </template>
32
33 <style>
34   body {
35     background-color: #b3b3b3;
36   }
37 </style>
```

3.5 总结

1. 组件内有多处不确定的结构 怎么办？

答：具名插槽

2. 具名插槽的使用语法是什么？

答：1、 `<slot name="名字"> 默认内容 </slot>`

2、 `<template #名字> 要展示的内容 </template>`

4. 作用域插槽

4.1 作用

带数据的插槽，可以让组件功能更强大、更灵活、复用性更高；用 slot 占位的同时，还可以给 slot 绑定数据，

将来使用组件时，不仅可以传内容，还能使用 slot 带来的数据

4.2 场景

封装表格组件

序号	姓名	年纪	操作
1	狗蛋	19	删除
2	大锤	17	删除
3	铁棍	18	删除

序号	姓名	年纪	操作
1	Jack	18	查看
2	Rose	19	查看
3	Henry	17	查看

4.3 使用步骤

1. 给 slot 标签, 以添加属性的方式传值

```
1 <slot a="hello" :b="666"></slot>
```


2. 所有添加的属性, 都会被收集到一个对象中

```
1 { a: 'hello', b: 666 }
```

3. 在template中, 通过 #插槽名= "obj" 接收, 默认插槽名为 default

```
1 <!-- obj会收集 slot 上绑定的所有自定义属性 -->
2 <template #default="obj">
3   {{ obj }}
4 </template>
```

4.4 静态代码

components/MyTable.vue

```
1 <script setup>
2
3 </script>
4
5 <template>
6   <table class="my-table">
7     <thead>
8       <tr>
9         <th>序号</th>
10        <th>姓名</th>
11        <th>年纪</th>
12        <th>操作</th>
13      </tr>
14    </thead>
15    <tbody>
16      <tr>
17        <td>1</td>
18        <td>赵小云</td>
19        <td>19</td>
20        <td>
21          <button>查看</button>
22        </td>
23      </tr>
24      <tr>
25        <td>1</td>
26        <td>张小花</td>
27        <td>19</td>
```

```
28         <td>
29             <button>查看</button>
30         </td>
31     </tr>
32     <tr>
33         <td>1</td>
34         <td>孙大明</td>
35         <td>19</td>
36         <td>
37             <button>查看</button>
38         </td>
39     </tr>
40 </tbody>
41 </table>
42 </template>
43
44 <style>
45 .my-table {
46     width: 450px;
47     text-align: center;
48     border: 1px solid #ccc;
49     font-size: 24px;
50     margin: 30px auto;
51 }
52 .my-table thead {
53     background-color: #1f74ff;
54     color: #fff;
55 }
56 .my-table thead th {
57     font-weight: normal;
58 }
59 .my-table thead tr {
60     line-height: 40px;
61 }
62 .my-table th,
63 .my-table td {
64     border-bottom: 1px solid #ccc;
65     border-right: 1px solid #ccc;
66 }
67 .my-table td:last-child {
68     border-right: none;
69 }
70 .my-table tr:last-child td {
71     border-bottom: none;
72 }
73 .my-table button {
74     width: 65px;
```

```
75 height: 35px;
76 font-size: 18px;
77 border: 1px solid #ccc;
78 outline: none;
79 border-radius: 3px;
80 cursor: pointer;
81 background-color: #ffffff;
82 margin-left: 5px;
83 }
84 </style>
```

App.vue

```
1 <script setup>
2   import { ref } from 'vue'
3
4   import MyTable from './components/MyTable.vue'
5
6   const tableData1 = ref([
7     { id: 11, name: '狗蛋', age: 18 },
8     { id: 22, name: '大锤', age: 19 },
9     { id: 33, name: '铁棍', age: 17 }
10  ])
11
12  const tableData2 = ref([
13    { id: 21, name: 'Jack', age: 18 },
14    { id: 32, name: 'Rose', age: 19 },
15    { id: 43, name: 'Henry', age: 17 }
16  ])
17 </script>
18 <template>
19   <MyTable />
20
21   <MyTable />
22 </template>
23
24 <style>
25   body {
26     background-color: #fff;
27   }
28 </style>
```

4.5 完整代码

```
1 <script setup>
2   const props = defineProps({
3     // 数据源
4     data: {
5       type: Array,
6       default: () => []
7     }
8   })
9 </script>
10
11 <template>
12   <table class="my-table">
13     <thead>
14       <tr>
15         <th>序号</th>
16         <th>姓名</th>
17         <th>年纪</th>
18         <th>操作</th>
19       </tr>
20     </thead>
21     <tbody>
22       <tr
23         v-for="(item, index) in props.data"
24         :key="item.id">
25         <td>{{ index + 1 }}</td>
26         <td>{{ item.name }}</td>
27         <td>{{ item.age }}</td>
28         <td>
29           <slot :i="index"></slot>
30         </td>
31       </tr>
32     </tbody>
33   </table>
34 </template>
35
36 <style>
37   .my-table {
38     width: 450px;
39     text-align: center;
40     border: 1px solid #ccc;
41     font-size: 24px;
42     margin: 30px auto;
43   }
44   .my-table thead {
```

```
45     background-color: #1f74ff;
46     color: #fff;
47   }
48   .my-table thead th {
49     font-weight: normal;
50   }
51   .my-table thead tr {
52     line-height: 40px;
53   }
54   .my-table th,
55   .my-table td {
56     border-bottom: 1px solid #ccc;
57     border-right: 1px solid #ccc;
58   }
59   .my-table td:last-child {
60     border-right: none;
61   }
62   .my-table tr:last-child td {
63     border-bottom: none;
64   }
65   .my-table button {
66     width: 65px;
67     height: 35px;
68     font-size: 18px;
69     border: 1px solid #ccc;
70     outline: none;
71     border-radius: 3px;
72     cursor: pointer;
73     background-color: #ffffff;
74     margin-left: 5px;
75   }
76 </style>
77
```

App.vue

```
1 <script setup>
2   import { ref } from 'vue'
3
4   import MyTable from './components/MyTable.vue'
5
6   const tableData1 = ref([
7     { id: 11, name: '狗蛋', age: 19 },
8     { id: 22, name: '大锤', age: 17 },
9     { id: 33, name: '铁棍', age: 18 }
```

```

10  })
11
12  const tableData2 = ref([
13    { id: 21, name: 'Jack', age: 18 },
14    { id: 32, name: 'Rose', age: 19 },
15    { id: 43, name: 'Henry', age: 17 }
16  ])
17
18  // 删除
19  const onDel = (i) => {
20    if (window.confirm('确认删除么?')) {
21      tableData1.value.splice(i, 1)
22    }
23  }
24
25  // 查看
26  const onInspect = (i) => {
27    alert(JSON.stringify(tableData2.value[i]))
28  }
29 </script>
30 <template>
31   <MyTable :data="tableData1">
32     <template #default="{ i }">
33       <button @click="onDel(i)">删除</button>
34     </template>
35   </MyTable>
36
37   <MyTable :data="tableData2">
38     <template #default="{ i }">
39       <button @click="onInspect(i)">查看</button>
40     </template>
41   </MyTable>
42 </template>
43
44 <style>
45   body {
46     background-color: #fff;
47   }
48 </style>

```

4.6 总结

1. 作用域插槽的作用是什么？

答：让插槽带数据, 使得组件功能更强大、更灵活

2. 作用域插槽的使用步骤是什么？

答：1、 slot上绑定数据

2、 <template #名字="{ 数据 }"> </template>

三、综合案例

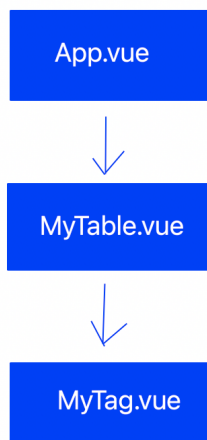
1. 整体效果和分析

1.1 整体效果

编号	封面	名字	操作
101		梨皮朱泥三热清代小品圆鼓典雅紫砂壶	茶具
102		全防水HABU旋钮牛皮户外徒步鞋山字泰抗鼠	男鞋
103		毛茸茸小熊出没，儿童羊羔绒背心73-90cm	儿童服饰
104		基础百搭，儿童宽头针织毛衣1-2岁	儿童服饰



1.2 结构分析



1.3 需求说明

1、my-table 表格组件封装

- 动态传递表格数据渲染
- 表头支持用户自定义
- 主体支持用户自定义

2、my-tag 标签组件封装

- 双击显示输入框，输入框获取焦点
- 失去焦点，隐藏输入框
- 回显标签信息
- 内容修改, 回车修改标签信息

2. 封装 MyTable 并渲染

2.1 静态代码

components/MyTable.vue

```
1 <script setup>
2 </script>
3
4 <template>
5   <table class="my-table">
6     <thead>
7       <tr>
8         <th>编号</th>
9         <th>名称</th>
10        <th>图片</th>
11        <th width="100px">标签</th>
12      </tr>
13    </thead>
14    <tbody>
15      <tr>
16        <td>1</td>
17        <td>毛茸茸小熊出没，儿童羊羔绒背心73-90cm</td>
18        <td>
19          
20        </td>
21        <td>
22          标签内容1
23        </td>
24      </tr>
25      <tr>
26        <td>2</td>
27        <td>毛茸茸小熊出没，儿童羊羔绒背心73-90cm</td>
28        <td>
29          
30        </td>
31        <td>
32          标签内容2
33        </td>
34      </tr>
35    </tbody>
36  </table>
37 </template>
38
```



```

39 <style lang="scss" scoped>
40 .my-table {
41   width: 100%;
42   border-spacing: 0;
43   img {
44     width: 100px;
45     height: 100px;
46     object-fit: contain;
47     vertical-align: middle;
48   }
49   th {
50     background: #f5f5f5;
51     border-bottom: 2px solid #069;
52   }
53   td {
54     border-bottom: 1px dashed #ccc;
55   }
56   td,
57   th {
58     text-align: center;
59     padding: 10px;
60     transition: all .5s;
61     &.red {
62       color: red;
63     }
64   }
65   .none {
66     height: 100px;
67     line-height: 100px;
68     color: #999;
69   }
70 }
71
72 </style>

```

App.vue

```

1 <script setup>
2   import { ref } from 'vue'
3   import MyTable from './components/MyTable.vue'
4
5   // 商品列表
6   const goodsList = ref([
7     {
8       id: 101,

```

```
9     picture:
10       'https://yanxuan-
item.nosdn.127.net/f8c37ffa41ab1eb84bff499e1f6acfc7.jpg',
11     name: '梨皮朱泥三绝清代小品壶经典款紫砂壶',
12     tag: '茶具'
13   },
14   {
15     id: 102,
16     picture:
17       'https://yanxuan-
item.nosdn.127.net/221317c85274a188174352474b859d7b.jpg',
18     name: '全防水HABU旋钮牛皮户外徒步鞋山宁泰抗菌',
19     tag: '男鞋'
20   },
21   {
22     id: 103,
23     picture:
24       'https://yanxuan-
item.nosdn.127.net/cd4b840751ef4f7505c85004f0bebc5.png',
25     name: '毛茸茸小熊出没, 儿童羊羔绒背心73-90cm',
26     tag: '儿童服饰'
27   },
28   {
29     id: 104,
30     picture:
31       'https://yanxuan-
item.nosdn.127.net/56eb25a38d7a630e76a608a9360eec6b.jpg',
32     name: '基础百搭, 儿童套头针织毛衣1-9岁',
33     tag: '儿童服饰'
34   }
35 ])
36 </script>
37 <template>
38   <MyTable />
39 </template>
40
41 <style lang="scss">
42   #app {
43     width: 1000px;
44     margin: 50px auto;
45     img {
46       width: 100px;
47       height: 100px;
48       object-fit: contain;
49       vertical-align: middle;
50     }
51   }
```

2.2 完整代码

components/MyTable.vue

```
1 <script setup>
2   const props = defineProps({
3     // 数据源
4     data: {
5       type: Array,
6       default: () => []
7     }
8   })
9 </script>
10
11 <template>
12   <table class="my-table">
13     <thead>
14       <tr>
15         <th>编号</th>
16         <th>名称</th>
17         <th>图片</th>
18         <th width="100px">标签</th>
19       </tr>
20     </thead>
21     <tbody>
22       <tr v-for="(item, index) in props.data" :key="item.id">
23         <td>{{ index + 1 }}</td>
24         <td>{{ item.name }}</td>
25         <td>
26           
29         </td>
30         <td>
31           标签内容
32         </td>
33       </tr>
34     </tbody>
35   </table>
36 </template>
37
38 <style lang="scss">
39
40 .my-table {
```

```

41 width: 100%;
42 border-spacing: 0;
43 img {
44   width: 100px;
45   height: 100px;
46   object-fit: contain;
47   vertical-align: middle;
48 }
49 th {
50   background: #f5f5f5;
51   border-bottom: 2px solid #069;
52 }
53 td {
54   border-bottom: 1px dashed #ccc;
55 }
56 td,
57 th {
58   text-align: center;
59   padding: 10px;
60   transition: all .5s;
61   &.red {
62     color: red;
63   }
64 }
65 .none {
66   height: 100px;
67   line-height: 100px;
68   color: #999;
69 }
70 }
71
72 </style>

```

App.vue

```

1 <script setup>
2   import { ref } from 'vue'
3   import MyTable from './components/MyTable.vue'
4
5   // 商品列表
6   const goodsList = ref([
7     {
8       id: 101,
9       picture:

```

```
10     'https://yanxuan-
item.nosdn.127.net/f8c37ffa41ab1eb84bffa499e1f6acfc7.jpg',
11     name: '梨皮朱泥三绝清代小品壶经典款紫砂壶',
12     tag: '茶具'
13 },
14 {
15     id: 102,
16     picture:
17     'https://yanxuan-
item.nosdn.127.net/221317c85274a188174352474b859d7b.jpg',
18     name: '全防水HABU旋钮牛皮户外徒步鞋山宁泰抗菌',
19     tag: '男鞋'
20 },
21 {
22     id: 103,
23     picture:
24     'https://yanxuan-
item.nosdn.127.net/cd4b840751ef4f7505c85004f0bebc5.png',
25     name: '毛茸茸小熊出没，儿童羊羔绒背心73-90cm',
26     tag: '儿童服饰'
27 },
28 {
29     id: 104,
30     picture:
31     'https://yanxuan-
item.nosdn.127.net/56eb25a38d7a630e76a608a9360eec6b.jpg',
32     name: '基础百搭，儿童套头针织毛衣1-9岁',
33     tag: '儿童服饰'
34 }
35 ])
36 </script>
37 <template>
38     <MyTable :data="goodsList" />
39 </template>
40
41 <style lang="scss">
42     #app {
43         width: 1000px;
44         margin: 50px auto;
45         img {
46             width: 100px;
47             height: 100px;
48             object-fit: contain;
49             vertical-align: middle;
50         }
51         td:last-child {
52             width: 150px;
```

```
53     }  
54   }  
55 </style>
```

3. MyTable插槽自定义

components/MyTable.vue

```
1 <template>  
2   <table class="my-table">  
3     <thead>  
4       <tr>  
5         <!-- 表头具名插槽 -->  
6         <slot name="thead"></slot>  
7       </tr>  
8     </thead>  
9     <tbody>  
10      <tr  
11        v-for="(item, index) in props.data"  
12        :key="item.id">  
13        <!-- 表体默认插槽 -->  
14        <slot  
15          :row="item"  
16          :i="index"></slot>  
17        </tr>  
18      </tbody>  
19    </table>  
20 </template>
```

App.vue

```
1 <template>  
2   <MyTable :data="goodsList">  
3     <!-- 表头插槽自定义 -->  
4     <template #thead>  
5       <th>序号</th>  
6       <th>封面</th>  
7       <th>名字</th>  
8       <th>操作</th>  
9     </template>  
10    <!-- 表体作用域插槽自定义 -->  
11    <template #default="{ row, i }">  
12      <td>{{ i + 1 }}</td>
```

```
13     <td>
14         
17     </td>
18     <td>{{ row.name }}</td>
19     <td>xxx</td>
20 </template>
21 </MyTable>
22 </template>
```

4. 封装MyTag组件

MyTag.vue

```
1 <script setup>
2
3 </script>
4
5 <template>
6     <div class="my-tag">
7         <input
8             class="input"
9             type="text"
10             placeholder="输入标签"
11         />
12         <div class="text">
13             茶具
14         </div>
15     </div>
16 </template>
17
18
19
20 <style lang="scss" scoped>
21 .my-tag {
22     cursor: pointer;
23     .input {
24         appearance: none;
25         outline: none;
26         border: 1px solid #ccc;
27         width: 100px;
28         height: 40px;
29         box-sizing: border-box;
30         padding: 10px;
```

```
31     color: #666;
32     &::placeholder {
33         color: #666;
34     }
35 }
36 }
37 </style>
```

App.vue

```
1 <script setup>
2   import MyTag from './components/MyTag.vue'
3 </script>
4
5 <template>
6   <td>
7     <MyTag />
8   </td>
9 </template>
```

5. 控制MyTag交互

MyTag.vue

```
1 <script setup>
2   import { ref } from 'vue'
3   const isEdit = ref(false)
4
5   // 双击
6   const onDbClick = () => {
7     isEdit.value = true
8   }
9 </script>
10
11 <template>
12   <div class="my-tag">
13     <input
14       v-if="isEdit"
15       v-focus
16       class="input"
17       type="text"
18       placeholder="输入标签"
19       @blur="isEdit = false">
```



```
20     />
21     <div
22       v-else
23       @dblclick="onDbClick"
24       class="text">
25       茶具
26     </div>
27   </div>
28 </template>
```

```
1 // main.js
2
3 // 注册全局指令: focus
4 app.directive('focus',{
5   mounted(el) {
6     el.focus()
7   }
8 })
```

6. v-model优化MyTag

App.vue

```
1 <MyTag v-model="row.tag" />
```

MyTag.vue

```
1 <script setup>
2   const model= defineModel()
3
4   // 回车
5   const onEnter = (e) => {
6     const value = e.target.value.trim()
7     if (value) {
8       model.value = value
9     }
10    isEdit.value = false
11  }
12 </script>
```

```
13
14 <template>
15   <div class="my-tag">
16     <input
17       v-if="isEdit"
18       v-focus
19       class="input"
20       type="text"
21       placeholder="输入标签"
22       :value="model"
23       @blur="isEdit = false"
24       @keyup.enter="onEnter"
25     />
26     <div
27       v-else
28       @dblclick="onDbClick"
29       class="text">
30       {{ tagName }}
31     </div>
32   </div>
33 </template>
```