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

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

2. 使用

```
<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

```
// main.js

// 注册全局指令
```

```
app.directive('focus', {
  mounted(el) {
    console.log(el)// input 元素
    // 聚焦
    el.focus()
  }
})
```

App.vue

```
<script setup></script>

<template>
  <div class="app">
    <input type="text" v-focus />
  </div>
</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

```
<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**

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

2.3 代码示例

```
// main.js
app.directive('color', {
  mounted(el, binding) {
    el.style.color = binding.value
  },
  updated(el, binding) {
    el.style.color = binding.value
  }
})
```

App.vue

```
<script setup>
  import { ref } from 'vue'
  // 颜色
  const colorStr = ref('red')
</script>

<template>
   <p v-color="colorStr"></p>
</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：

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

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

```vue
<script setup>
  import axios from 'axios'
  import { ref } from 'vue'

  // 新闻列表
  const newsList = ref([])

  getNewsData()

  // 获取新闻列表
  async function getNewsData() {
    // 请求新闻数据
    const resp = await axios.get('http://localhost:4000/api/news')
    // 保存数据
    newsList.value = resp.data.data
  }
</script>

<template>
```

```
  <div class="box">
    <ul>
      <li
        v-for="item in newsList"
        :key="item.id"
        class="news">
        <div class="left">
          <div class="title">{{ item.title }}</div>
          <div class="info">
            <span>{{ item.source }}</span>
            <span>{{ item.time }}</span>
          </div>
        </div>
        <div class="right">
          <img
            :src="item.img"
            alt="img" />
        </div>
      </li>
    </ul>
  </div>
</template>

<style>
  .loading:before {
    z-index: 999;
    content: '';
    position: absolute;
    left: 0;
    top: 0;
    width: 100%;
    height: 100%;
    background: #fff url('./assets/loading.gif') no-repeat center;
    background-size: auto;
  }

  .box {
    position: relative;
    width: 800px;
    min-height: 600px;
    margin: 10px auto;
    border-radius: 5px;
  }
  .news {
    display: flex;
    margin: 0 auto;
    padding: 20px 0;
```

```
    cursor: pointer;
  }
  .news .left {
    flex: 1;
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    padding-right: 10px;
  }
  .news .left .title {
    font-size: 20px;
  }
  .news .left .info {
    color: #999999;
  }
  .news .left .info span {
    margin-right: 20px;
  }
  .news .right {
    width: 160px;
    height: 120px;
  }
  .news .right img {
    width: 100%;
    height: 100%;
    object-fit: cover;
  }
</style>
```

```
// main.js
app.directive('loading', (el, binding) => {
  if (binding.value) {
    el.classList.add('loading')
  } else {
    el.classList.remove('loading')
  }
})
```


二、插槽

插槽分类

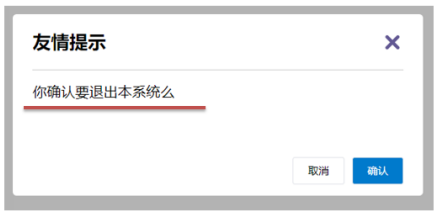
- 默认插槽
- 具名插槽

- 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

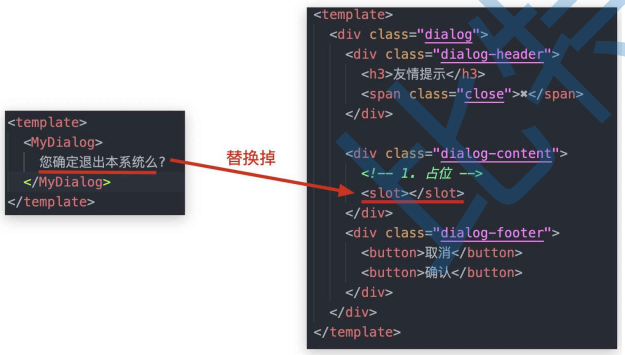
将需要多次显示的对话框,封装成一个组件

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

```vue
<script setup>

</script>
<template>
  <div class="dialog">
    <div class="dialog-header">
      <h3>友情提示</h3>
```

```
      <span class="close">✖</span>
    </div>

    <div class="dialog-content">您确定要进行删除操作吗? </div>
    <div class="dialog-footer">
      <button>取消</button>
      <button>确认</button>
    </div>
  </div>
</template>

<style scoped>
  * {
    margin: 0;
    padding: 0;
  }
  .dialog {
    width: 470px;
    height: 230px;
    padding: 0 25px;
    background-color: #ffffff;
    margin: 40px auto;
    border-radius: 5px;
  }
  .dialog-header {
    height: 70px;
    line-height: 70px;
    font-size: 20px;
    border-bottom: 1px solid #ccc;
    position: relative;
  }
  .dialog-header .close {
    position: absolute;
    right: 0px;
    top: 0px;
    cursor: pointer;
  }
  .dialog-content {
    height: 80px;
    font-size: 18px;
    padding: 15px 0;
  }
  .dialog-footer {
    display: flex;
    justify-content: flex-end;
  }
  .dialog-footer button {
```

```
    width: 65px;
    height: 35px;
    background-color: #ffffff;
    border: 1px solid #e1e3e9;
    cursor: pointer;
    outline: none;
    margin-left: 10px;
    border-radius: 3px;
  }
  .dialog-footer button:last-child {
    background-color: #007acc;
    color: #fff;
  }
</style>
```

App.vue

```
<script setup>
  import MyDialog from './components/MyDialog.vue'
</script>
<template>
  <MyDialog />
</template>

<style>
  body {
    background-color: #b3b3b3;
  }
</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

```vue
<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

```vue
<MyDialog />
```

- 使用组件时，传了，则slot整体会被换掉, 从而显示传入的

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

2.5 代码示例

App.vue

```vue
<script setup>
  import MyDialog from './components/MyDialog.vue'
</script>

<template>
```

```vue
  <!-- 1. 不传 -->
  <MyDialog />

  <!-- 2. 传了 -->
  <MyDialog>
    <h4>确定删除么?</h4>
  </MyDialog>

</template>

<style>
 body {
   background-color: #b3b3b3;
 }
</style>
```

MyDialog.vue

```vue
<script setup>

</script>
<template>
  <div class="dialog">
    <div class="dialog-header">
      <h3>友情提示</h3>
      <span class="close">✖</span>
    </div>

    <div class="dialog-content">
      <!-- 插槽默认内容 -->
      <slot>我是默认内容</slot>
    </div>
    <div class="dialog-footer">
      <button>取消</button>
      <button>确认</button>
    </div>
  </div>
</template>

<style scoped>
  * {
    margin: 0;
    padding: 0;
  }
  .dialog {
```

```
    width: 470px;
    height: 230px;
    padding: 0 25px;
    background-color: #ffffff;
    margin: 40px auto;
    border-radius: 5px;
  }
  .dialog-header {
    height: 70px;
    line-height: 70px;
    font-size: 20px;
    border-bottom: 1px solid #ccc;
    position: relative;
  }
  .dialog-header .close {
    position: absolute;
    right: 0px;
    top: 0px;
    cursor: pointer;
  }
  .dialog-content {
    height: 80px;
    font-size: 18px;
    padding: 15px 0;
  }
  .dialog-footer {
    display: flex;
    justify-content: flex-end;
  }
  .dialog-footer button {
    width: 65px;
    height: 35px;
    background-color: #ffffff;
    border: 1px solid #e1e3e9;
    cursor: pointer;
    outline: none;
    margin-left: 10px;
    border-radius: 3px;
  }
  .dialog-footer button:last-child {
    background-color: #007acc;
    color: #fff;
  }
</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

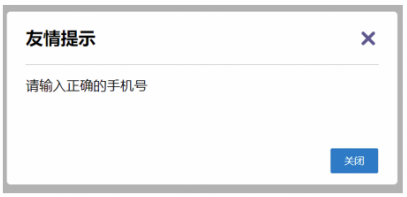
2. 默认内容何时显示？何时不显示?

   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

```vue
<script setup></script>

<template>
  <div class="dialog">
    <div class="dialog-header">
      <slot name="header">
        <h3>温馨提示</h3>
      </slot>
      <span class="close">❌</span>
    </div>

    <div class="dialog-content">
      <slot>我是主体内容</slot>
    </div>
```

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

<style>
  * {
    margin: 0;
    padding: 0;
  }
  .dialog {
    width: 470px;
    height: 230px;
    padding: 0 25px;
    background-color: #ffffff;
    margin: 40px auto;
    border-radius: 5px;
  }
  .dialog-header {
    height: 70px;
    line-height: 70px;
    font-size: 20px;
    border-bottom: 1px solid #ccc;
    position: relative;
  }
  .dialog-header .close {
    position: absolute;
    right: 0px;
    top: 0px;
    cursor: pointer;
  }
  .dialog-content {
    height: 80px;
    font-size: 18px;
    padding: 15px 0;
  }
  .dialog-footer {
    display: flex;
    justify-content: flex-end;
  }
  .dialog-footer button {
    width: 65px;
    height: 35px;
```

```
    background-color: #ffffff;
    border: 1px solid #e1e3e9;
    cursor: pointer;
    outline: none;
    margin-left: 10px;
    border-radius: 3px;
  }
  .dialog-footer button:last-child {
    background-color: #007acc;
    color: #fff;
  }
</style>
```

App.vue

```
<script setup>
  import MyDialog from './components/MyDialog.vue'
</script>

<template>
  <!-- 1. 不传、都显示插槽默认内容 -->
  <MyDialog />

  <!-- 2. 传了 -->
  <MyDialog>
    <template v-slot:header>
      <h3>友情提示</h3>
    </template>
    <template v-slot:default> 请输入正确手机号 </template>
    <template v-slot:footer>
      <button>关闭</button>
    </template>
  </MyDialog>

  <!-- 3. 传了 -->
  <MyDialog>
    <template v-slot:header>
      <h3>警告</h3>
    </template>
    <template v-slot:default> 你确认要退出么? </template>
    <template v-slot:footer>
      <button>确认</button>
      <button>取消</button>
    </template>
  </MyDialog>
```

```
</template>

<style>
  body {
    background-color: #b3b3b3;
  }
</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

```
<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

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

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

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

4.4 静态代码

`components/MyTable.vue`

```
<script setup>

</script>

<template>
  <table class="my-table">
    <thead>
      <tr>
        <th>序号</th>
        <th>姓名</th>
        <th>年纪</th>
        <th>操作</th>
      </tr>
    </thead>
    <tbody>
      <tr>
        <td>1</td>
        <td>赵小云</td>
        <td>19</td>
        <td>
          <button>查看</button>
        </td>
      </tr>
        <tr>
        <td>1</td>
        <td>张小花</td>
        <td>19</td>
```

```
        <td>
          <button>查看</button>
        </td>
      </tr>
        <tr>
        <td>1</td>
        <td>孙大明</td>
        <td>19</td>
        <td>
          <button>查看</button>
        </td>
      </tr>
    </tbody>
  </table>
</template>

<style>
.my-table {
  width: 450px;
  text-align: center;
  border: 1px solid #ccc;
  font-size: 24px;
  margin: 30px auto;
}
.my-table thead {
  background-color: #1f74ff;
  color: #fff;
}
.my-table thead th {
  font-weight: normal;
}
.my-table thead tr {
  line-height: 40px;
}
.my-table th,
.my-table td {
  border-bottom: 1px solid #ccc;
  border-right: 1px solid #ccc;
}
.my-table td:last-child {
  border-right: none;
}
.my-table tr:last-child td {
  border-bottom: none;
}
.my-table button {
  width: 65px;
```

```
  height: 35px;
  font-size: 18px;
  border: 1px solid #ccc;
  outline: none;
  border-radius: 3px;
  cursor: pointer;
  background-color: #ffffff;
  margin-left: 5px;
}
</style>
```

App.vue

```
<script setup>
  import { ref } from 'vue'

  import MyTable from './components/MyTable.vue'

  const tableData1 = ref([
    { id: 11, name: '狗蛋', age: 18 },
    { id: 22, name: '大锤', age: 19 },
    { id: 33, name: '铁棍', age: 17 }
  ])

  const tableData2 = ref([
    { id: 21, name: 'Jack', age: 18 },
    { id: 32, name: 'Rose', age: 19 },
    { id: 43, name: 'Henry', age: 17 }
  ])
</script>
<template>
  <MyTable />

  <MyTable />
</template>

<style>
  body {
    background-color: #fff;
  }
</style>
```


4.5 完整代码

components/MyTable.vue

```vue
<script setup>
  const props = defineProps({
    // 数据源
    data: {
      type: Array,
      default: () => []
    }
  })
</script>

<template>
  <table class="my-table">
    <thead>
      <tr>
        <th>序号</th>
        <th>姓名</th>
        <th>年纪</th>
        <th>操作</th>
      </tr>
    </thead>
    <tbody>
      <tr
        v-for="(item, index) in props.data"
        :key="item.id">
        <td>{{ index + 1 }}</td>
        <td>{{ item.name }}</td>
        <td>{{ item.age }}</td>
        <td>
          <slot :i="index"></slot>
        </td>
      </tr>
    </tbody>
  </table>
</template>

<style>
  .my-table {
    width: 450px;
    text-align: center;
    border: 1px solid #ccc;
    font-size: 24px;
    margin: 30px auto;
  }
  .my-table thead {
```

```
    background-color: #1f74ff;
    color: #fff;
  }
  .my-table thead th {
    font-weight: normal;
  }
  .my-table thead tr {
    line-height: 40px;
  }
  .my-table th,
  .my-table td {
    border-bottom: 1px solid #ccc;
    border-right: 1px solid #ccc;
  }
  .my-table td:last-child {
    border-right: none;
  }
  .my-table tr:last-child td {
    border-bottom: none;
  }
  .my-table button {
    width: 65px;
    height: 35px;
    font-size: 18px;
    border: 1px solid #ccc;
    outline: none;
    border-radius: 3px;
    cursor: pointer;
    background-color: #ffffff;
    margin-left: 5px;
  }
</style>

```

App.vue

```
<script setup>
  import { ref } from 'vue'

  import MyTable from './components/MyTable.vue'

  const tableData1 = ref([
    { id: 11, name: '狗蛋', age: 19 },
    { id: 22, name: '大锤', age: 17 },
    { id: 33, name: '铁棍', age: 18 }
```

```
  ])

  const tableData2 = ref([
    { id: 21, name: 'Jack', age: 18 },
    { id: 32, name: 'Rose', age: 19 },
    { id: 43, name: 'Henry', age: 17 }
  ])

  // 删除
  const onDel = (i) => {
    if (window.confirm('确认删除么?')) {
      tableData1.value.splice(i, 1)
    }
  }

  // 查看
  const onInspect = (i) => {
    alert(JSON.stringify(tableData2.value[i]))
  }
</script>
<template>
  <MyTable :data="tableData1">
    <template #default="{ i }">
      <button @click="onDel(i)">删除</button>
    </template>
  </MyTable>

  <MyTable :data="tableData2">
    <template #default="{ i }">
      <button @click="onInspect(i)">查看</button>
    </template>
  </MyTable>
</template>

<style>
  body {
    background-color: #fff;
  }
</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`

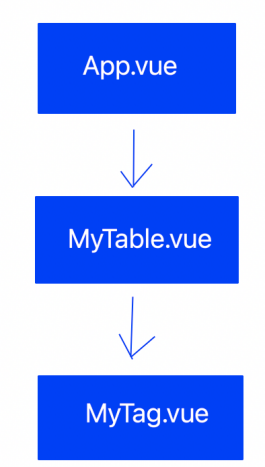
2、 `<template #名字="{ 数据 }"> </template>`

三、综合案例

1. 整体效果和分析

1.1 整体效果

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`

```vue
<script setup>
</script>

<template>
  <table class="my-table">
    <thead>
      <tr>
        <th>编号</th>
        <th>名称</th>
        <th>图片</th>
        <th width="100px">标签</th>
      </tr>
    </thead>
    <tbody>
      <tr>
       <td>1</td>
       <td>毛茸茸小熊出没，儿童羊羔绒背心73-90cm</td>
        <td>
          <img src="https://yanxuan-item.nosdn.127.net/cd4b840751ef4f7505c85004f0bebcb5.png" />
        </td>
        <td>
          标签内容1
        </td>
      </tr>
      <tr>
       <td>2</td>
       <td>毛茸茸小熊出没，儿童羊羔绒背心73-90cm</td>
        <td>
          <img src="https://yanxuan-item.nosdn.127.net/cd4b840751ef4f7505c85004f0bebcb5.png" />
        </td>
        <td>
          标签内容2
        </td>
      </tr>
    </tbody>
  </table>
</template>

```

```scss
<style lang="scss" scoped>
.my-table {
  width: 100%;
  border-spacing: 0;
  img {
    width: 100px;
    height: 100px;
    object-fit: contain;
    vertical-align: middle;
  }
  th {
    background: #f5f5f5;
    border-bottom: 2px solid #069;
  }
  td {
    border-bottom: 1px dashed #ccc;
  }
  td,
  th {
    text-align: center;
    padding: 10px;
    transition: all .5s;
    &.red {
      color: red;
    }
  }
  .none {
    height: 100px;
    line-height: 100px;
    color: #999;
  }
}

</style>
```

App.vue

```vue
<script setup>
  import { ref } from 'vue'
  import MyTable from './components/MyTable.vue'

  // 商品列表
  const goodsList = ref([
    {
      id: 101,
```

```
      picture:
        'https://yanxuan-item.nosdn.127.net/f8c37ffa41ab1eb84bff499e1f6acfc7.jpg',
      name: '梨皮朱泥三绝清代小品壶经典款紫砂壶',
      tag: '茶具'
    },
    {
      id: 102,
      picture:
        'https://yanxuan-item.nosdn.127.net/221317c85274a188174352474b859d7b.jpg',
      name: '全防水HABU旋钮牛皮户外徒步鞋山宁泰抗菌',
      tag: '男鞋'
    },
    {
      id: 103,
      picture:
        'https://yanxuan-item.nosdn.127.net/cd4b840751ef4f7505c85004f0bebcb5.png',
      name: '毛茸茸小熊出没，儿童羊羔绒背心73-90cm',
      tag: '儿童服饰'
    },
    {
      id: 104,
      picture:
        'https://yanxuan-item.nosdn.127.net/56eb25a38d7a630e76a608a9360eec6b.jpg',
      name: '基础百搭，儿童套头针织毛衣1-9岁',
      tag: '儿童服饰'
    }
  ])
</script>
<template>
  <MyTable />
</template>

<style lang="scss">
  #app {
    width: 1000px;
    margin: 50px auto;
    img {
      width: 100px;
      height: 100px;
      object-fit: contain;
      vertical-align: middle;
    }
  }
```

```
</style>
```

2.2 完整代码

components/MyTable.vue

```vue
<script setup>
  const props = defineProps({
    // 数据源
    data: {
      type: Array,
      default: () => []
    }
  })
</script>

<template>
  <table class="my-table">
    <thead>
      <tr>
        <th>编号</th>
        <th>名称</th>
        <th>图片</th>
        <th width="100px">标签</th>
      </tr>
    </thead>
    <tbody>
      <tr v-for="(item, index) in props.data" :key="item.id">
       <td>{{ index + 1 }}</td>
        <td>{{ item.name }}</td>
        <td>
          <img
            :src="item.picture"
          />
        </td>
        <td>
          标签内容
        </td>
      </tr>
    </tbody>
  </table>
</template>

<style lang="scss">

.my-table {
```

```
  width: 100%;
  border-spacing: 0;
  img {
    width: 100px;
    height: 100px;
    object-fit: contain;
    vertical-align: middle;
  }
  th {
    background: #f5f5f5;
    border-bottom: 2px solid #069;
  }
  td {
    border-bottom: 1px dashed #ccc;
  }
  td,
  th {
    text-align: center;
    padding: 10px;
    transition: all .5s;
    &.red {
      color: red;
    }
  }
  .none {
    height: 100px;
    line-height: 100px;
    color: #999;
  }
}

</style>
```

App.vue

```
<script setup>
  import { ref } from 'vue'
  import MyTable from './components/MyTable.vue'

  // 商品列表
  const goodsList = ref([
    {
      id: 101,
      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/cd4b840751ef4f7505c85004f0bebcb5.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;
```

```
    }
  }
</style>
```

3. MyTable插槽自定义

`components/MyTable.vue`

```
<template>
  <table class="my-table">
    <thead>
      <tr>
        <!-- 表头具名插槽 -->
        <slot name="thead"></slot>
      </tr>
    </thead>
    <tbody>
      <tr
        v-for="(item, index) in props.data"
        :key="item.id">
        <!-- 表体默认插槽 -->
        <slot
          :row="item"
          :i="index"></slot>
      </tr>
    </tbody>
  </table>
</template>
```

`App.vue`

```
<template>
  <MyTable :data="goodsList">
    <!-- 表头插槽自定义 -->
    <template #thead>
      <th>序号</th>
      <th>封面</th>
      <th>名字</th>
      <th>操作</th>
    </template>
    <!-- 表体作用域插槽自定义 -->
    <template #default="{ row, i }">
      <td>{{ i + 1 }}</td>
```

```
      <td>
        <img
          :src="row.picture"
          alt="picture" />
      </td>
      <td>{{ row.name }}</td>
      <td>xxx</td>
    </template>
  </MyTable>
</template>
```

4. 封装MyTag组件

MyTag.vue

```
<script setup>

</script>

<template>
  <div class="my-tag">
    <input
      class="input"
      type="text"
      placeholder="输入标签"
    />
    <div class="text">
       茶具
    </div>
  </div>
</template>



<style lang="scss" scoped>
.my-tag {
  cursor: pointer;
  .input {
    appearance: none;
    outline: none;
    border: 1px solid #ccc;
    width: 100px;
    height: 40px;
    box-sizing: border-box;
    padding: 10px;
```

```
    color: #666;
    &::placeholder {
      color: #666;
    }
  }
}
</style>
```

App.vue

```
<script setup>
  import MyTag from './components/MyTag.vue'
</script>

<template>
  <td>
    <MyTag />
  </td>
</template>
```

5. 控制MyTag交互

MyTag.vue

```
<script setup>
  import { ref } from 'vue'
  const isEdit = ref(false)

  // 双击
  const onDblClick = () => {
    isEdit.value = true
  }
</script>

<template>
  <div class="my-tag">
    <input
      v-if="isEdit"
      v-focus
      class="input"
      type="text"
      placeholder="输入标签"
      @blur="isEdit = false"
```

```
    />
    <div
      v-else
      @dblclick="onDblClick"
      class="text">
       茶具
    </div>
  </div>
</template>
```

```
// main.js

// 注册全局指令: focus
app.directive('focus',{
  mounted(el) {
    el.focus()
  }
})
```

6. v-model优化MyTag

App.vue

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

MyTag.vue

```
<script setup>
  const model= defineModel()

  // 回车
  const onEnter = (e) => {
    const value = e.target.value.trim()
    if (value) {
      model.value = value
    }
    isEdit.value = false
  }
</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="onDblClick"
29       class="text">
30       {{ tagName }}
31     </div>
32   </div>
33 </template>
```