项目五 表单输入绑定

本章内容

- 5.1 双向绑定
- 5.2 基本用法
- 5.3 值绑定
- 5.4 修饰符
- 5.5 案例实战

5.1 双向绑定

- 对于Vue来说，使用v-bind并不能解决表单域对象双向绑定的需求，所谓双向绑定，就是无论是通过input还是Vue对象，都能修改绑定的数据对象的值。Vue提供了v-model进行双向绑定。

5.1 双向绑定

- 对于数据的绑定，不管是使用**插值表达式（{{}}）**还是**v-text**指令，对于**数据间的交互都是单向的**，只能将Vue实例里的值传递给页面，页面对数据值的任何操作却无法传递给model。

- MVVM模式最重要的一个特性，可以说是数据的双向绑定，而Vue作为一个MVVM框架，肯定也实现了数据的双向绑定。**在Vue中使用内置的v-model指令完成数据在View与Model间的双向绑定。**

5.1 双向绑定

- 可以用v-model指令在**表单<input>、<textarea>及<select>元素**上创建双向数据绑定。它会根据控件类型自动选取正确的方法来更新元素。尽管有些神奇，但v-model本质上不过是语法糖。**它负责监听用户的输入事件以更新数据，并对一些极端场景进行一些特殊处理。**

5.1 双向绑定

- v-model会**忽略**所有表单元素的value、checked、selected特性的**初始值**，而总是将Vue实例的数据作为数据来源。我们应该通过JavaScript在组件的data选项中声明初始值。
- 提示：表单元素可以与用户进行交互，所以使用v-model指令在表单控件或者组件上创建双向绑定。

5.2 基本用法

- 5.2.1 文本
- 5.2.2 多行文本
- 5.2.3 复选框
- 5.2.4 单选按钮
- 5.2.5 选择框

5.2.1 文本

- 最常用的就是文本的绑定了。在下面的示例中，绑定了name和age两个属性。
- 例5.1 绑定单行文本示例

```
<div id="app">
    <label for="name">姓名：</label>
    <input v-model="name" type="text" id="name">
    <p>{{name}}</p>
    <label for="age">年龄：</label>
    <input v-model="age" type="text" id="age">
    <p>{{age}}</p>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            name: '金城武',
            age: '35'
        }
    })
</script>
```

5.2.2 多行文本

- 把上面示例中的p标签换成textarea标签，即可实现多行文本的绑定。
- 例5.2 绑定多行文本示例

```
<div id="app">
        <span>第一章内容:</span>
        <p style="white-space: pre-line;">{{ message }}</p>
        <textarea v-model="message"></textarea>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                message: ""
            }
        })
    </script>
```

5.2.3 复选框

- 单个复选框，绑定到布尔值。
- 例5.3 绑定单个复选框示例

```
<div id="app">
        <input type="checkbox" id="checkbox" v-model="checked">
        <label for="checkbox">{{ checked }}</label>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                checked: true
            }
        })
    </script>
```

5.2.3 复选框

- 多个复选框，绑定到同一个数组，被选中的添加到数组中
- 例5.4 绑定多个复选框示例

```
<div id="app">
        <input type="checkbox" id="name1" value="马云" v-model="checkedNames">
        <label for="name1">马云</label>
        <input type="checkbox" id="name2" value="马化腾" v-model="checkedNames">
        <label for="name2">马化腾</label>
        <input type="checkbox" id="name3" value="马明哲" v-model="checkedNames">
        <label for="name3">马明哲</label>
        <p><span>哪些人做过首富: {{ checkedNames }}</span></p>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                checkedNames: [],   //定义空数组
            }
        })
    </script>
```

5.2.4 单选按钮

- 单选按钮一般都有多个条件可供选择，既然是单选按钮，自然希望实现互斥效果，可以使用v-model指令配合单选框的value来实现。

5.2.4 单选按钮

- 例5.5 绑定单选按钮示例

```
<div id="app">
        <h3>单选题</h3>
        <input type="radio" id="one" value="A" v-model="picked">
        <label for="one">A</label><br />
        <input type="radio" id="two" value="B" v-model="picked">
        <label for="two">B</label><br />
        <input type="radio" id="three" value="C" v-model="picked">
        <label for="three">C</label><br />
        <input type="radio" id="four" value="D" v-model="picked">
        <label for="four">D</label>
        <p><span>选择: {{ picked }}</span></p>
    </div>
    <script>
        new Vue({
            el: '#app',
            data: {
                picked: "C"
            }
        })
    </script>
```

5.2.5 选择框

- **1.单选框**
- 例5.6 绑定单选框示例

```
<div id="app">
    <h3>选择你最喜欢吃的水果</h3>
    <select v-model="selected">
        <option disabled value="">可以选择的水果如下</option>
        <option>苹果</option>
        <option>香蕉</option>
        <option>橘子</option>
    </select>
    <span>选择结果: {{ selected }}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            selected: ' '
        }
    })
</script>
```

5.2.5 选择框

- **2.多选框(绑定到一个数组)**
- 为<select>标签添加multiple属性，即可实现多选。
- 例5.7 绑定多选框示例

```
<div id="app">
    <h3>选择你喜欢吃的水果</h3>
    <select v-model="selected" multiple style="height: 100px">
        <option disabled value="">可以选择的水果如下</option>
        <option>苹果</option>
        <option>香蕉</option>
        <option>橘子</option>
        <option>草莓</option>
    </select>
    <span>选择结果: {{ selected }}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            selected: ["苹果", "香蕉"],
        }
    })
</script>
```

5.2.5 选择框

- **3.用v-for渲染的动态选项**
- 在实际应用场景中，<select>标签中的<option>一般是通过v-for指令动态输出的，其中每一项的value或text都可以使用v-bind动态输出。

5.2.5 选择框

- **3.用v-for渲染的动态选项**
- 例5.8 用v-for渲染的动态选项示例

```
<div id="app">
    <select v-model="selected">
        <option v-for="option in options" v-bind:value="option.value">
            {{option.text}}
        </option>
    </select>
    <span>选择结果: {{ selected }}</span>
</div>
<script>
    var vm = new Vue({
        el: '#app',
        data: {
            selected: '苹果',
            options: [
                { text: 'One', value: '苹果' },
                { text: 'Two', value: '香蕉' },
                { text: 'Three', value: '芒果' }
            ]
        }
    })
</script>
```

5.3 值绑定

- 5.3.1 绑定复选框
- 5.3.2 绑定单选按钮
- 5.3.3 绑定选择框

5.3.1 绑定复选框

- 在下面示例中，true-value和false-value特性并不会影响输入控件的value特性，因为浏览器在提交表单时并不会包含未被选中的复选框。
- 如果要确保表单中这两个值中的一个能够被提交，（例如“yes”或“no”），请换用单选按钮。

5.3.1 绑定复选框

- 例5.9 动态绑定复选框示例

```
<div id="app">
    <input type="checkbox" v-model="toggle" true-value="yes" false-value="no">
    <span>{{toggle}}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            toggle:'',
        }
    })
</script>
```

5.3.2 单选按钮

- 首先为单选按钮绑定一个属性a，定义属性值为“苹果”；然后使用v-model指令为单选按钮绑定pick属性，当单选按钮选中后，pick的值等于a的属性值。

5.3.2 单选按钮

- 例5.10 动态绑定单选按钮的值

```
<div id="app">
    <input type="radio" v-model="pick" v-bind:value="a">
    <span>{{ pick}}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            a:'该单选按钮已被选中',
            pick:'',
        }
    })
</script>
```

5.3.3 绑定选择框

- 在下面的示例中，定义了4个option选项，使用v-bind进行绑定。

5.3.3 绑定选择框

- 例5.11 动态绑定选择框的选项

```
<div id="app">
    <select v-model="selected" multiple>
        <option v-bind:value="{ number: 1 }">A</option>
        <option v-bind:value="{ number: 2 }">B</option>
        <option v-bind:value="{ number: 3 }">C</option>
        <option v-bind:value="{ number: 4 }">D</option>
    </select>
    <p><span>{{ selected }}</span></p>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            selected:[],
        }
    })
</script>
```

5.4 修饰符

- 5.4.1 lazy修饰符
- 5.4.2 number修饰符
- 5.4.3 trim修饰符

5.4 修饰符

- 对于v-model指令，有3个常用的修饰符
  - lazy修饰符
  - number修饰符
  - trim修饰符

5.4.1 lazy

- 在默认情况下，v-model在<input>和<textarea>中通常是用户输入数据时绑定数据。如果在v-model后添加一个修饰符.lazy，它就会转变为在change事件中绑定数据。

5.4.1 lazy修饰符

- 例5.12 lazy修饰符示例

```
<div id="app">
    <input v-model.lazy="message">
    <span>{{ message }}</span>
</div>
<script>
    new Vue({
        el: '#app',
        data: {
            message:'abc',
        }
    })
</script>
```

5.4.2 number修饰符

- number修饰符可以将输入的值转化为Number类型，否则虽然输入的是数字但它的类型其实是String，此修饰符在**数字输入框**中比较有用。
- 如果想自动将用户的输入值转为数值类型，可以给v-model添加number修饰符。因为即使在 type="number"时，HTML输入元素的值也总会返回字符串。如果这个值无法被parseFloat()解析，则会返回原始的值。

5.4.2 number

- 例5.13 number修饰符示例

```
<div id="app">
    <p>.number修饰符</p>
    <input type="number" v-model.number="val">
    <p>数据类型是：{{ typeof(val) }}</p>
</div>
<script>
    new Vue({
        el: '#app',
        data:{
            val:'',
        }
    })
</script>
```

5.4.3 trim

- 如果要自动过滤用户输入的首尾空格，可以给v-model添加trim修饰符。
- 例5.14 trim修饰符示例

```
<div id="app">
    <p>.trim修饰符</p>
    <input type="text" v-model.trim="val">
    <p>val的长度是：{{ val.length }}</p>
</div>
<script>
    new Vue({
        el: '#app',
        data:{
            val:'',
        }
    })
</script>
```

5.5 案例实战

- 5.5.1 破坏瓶子小游戏
- 5.5.2 动态表格

5.5.1 破坏瓶子小游戏

- 本案例应用前面所学的知识，来编写一个简单的破坏瓶子的小游戏。小游戏：其实是一张图片，对应一些按钮，通过不断地单击按钮，当单击一定次数后，用一张新图片替换原来的图片。

```
<style>
    #bottle{
        width:150px;                /*定义宽度*/
        height: 500px;              /*定义高度*/
        margin: 0 auto;             /*定义外边距*/
        background: url(001.png) center no-repeat;   /* 定义图片 居中 不平铺*/
        background-size: 80%;       /*定义背景图片尺寸*/
    }
    #bottle.burst{
        background-image:url(002.png);     /* 定义背景图片*/
    }
    #state{
        color: red;                 /*定义字体颜色*/
        text-align: center;             /*定义水平居中*/
    }
    #bottle-health{
        width:200px;                /*定义宽度*/
        border: 2px solid #000;         /*定义边框*/
        margin: 0 auto 20px auto;       /*定义外边距*/
    }
    #bottle-health div{
        height:10px;                /*定义高度*/
        background: #dc2b57;            /*定义背景颜色*/
    }
    #controls{
        width: 200px;               /*定义宽度*/
        margin: 0 auto;             /*定义外边距*/
    }
    #controls button{
        margin-left: 20px;              /*定义左侧外边距*/
    }
  </style>
```

```
<div id="app">
    <!--图片-->
    <div id="bottle" v-bind:class="{ burst:boole }"></div>
    <!--提示破碎的信息-->
    <div id="state">{{ state }}</div>
    <!--破坏进度情况-->
    <div id="bottle-health">
        <div v-bind:style="{width:health + '%' }"></div>
    </div>
    <!--控制按钮-->
    <div id="controls">
        <button v-on:click="beat" v-show="!boole">敲瓶子</button>
        <button v-on:click="restart">重新开始</button>
    </div>
</div>
```

```
<script>
    new Vue({
        el: '#app',
        data:{
            health:100,  //定义破坏的进度条
            boole:false,
            state:""
        },
        methods:{
            // 破坏瓶子
            beat:function () {
                this.health-=10;              //每次单击按钮触发beat方法，health宽度减小10%
                if (this.health<=0){
                    this.boole=true;          //当this.health<=0时，this. boole的值true，应用burst类更换背景图片
                    this.state="瓶子被敲碎了"  //提示“瓶子被敲碎”信息
                }
            },
            //重新开始
            restart:function(){
                this.health=100;      //进度条恢复100
                this.boole=false;     //this.boole回到定义时的值（false），显示瓶子没破坏前图片
                this.state=""         //提示“瓶子被敲碎”信息
            }
        }
    })
</script>
```

5.5.2 动态表格

- 本案例设计了一个动态表格，可以手动增加、删除编辑和更新数据。数据的添加和更新使用双向数据绑定来实现。效果如下图所示

```
<div id="table">
    <div class="add">
        <input type="text" v-model="addDetail.title" name="title" placeholder="发布内容" />
        <input type="text" v-model="addDetail.user" name="user" placeholder="发布人" />
        <input type="date" v-model="addDetail.dates" name="date" placeholder="发布时间" />
        <button @click="add">新增</button>
    </div>
    <table cellpadding="0" cellspacing="0">
        <thead>
            <tr>
                <th>序号</th>
                <th>标题</th>
                <th>发布人</th>
                <th>发布时间</th>
                <th>操作</th>
            </tr>
        </thead>
        <tbody>
            <tr v-for="(item,index) in newsList">
                <td width="10%">{{index+1}}</td>
                <td>{{item.title}}</td>
                <td width="15%">{{item.user}}</td>
                <td width="20%">{{item.dates}}</td>
                <td width="15%">
                    <span @click="deletelist(item.id,index)" class="delete">删除</span>
                    <span class="edit" @click="edit(item)">编辑</span>
                </td>
            </tr>
        </tbody>
    </table>
    <div id="mask" v-if="editlist">
        <div class="mask">
            <div class="title">
                编辑
                <span @click="editlist=false">x</span>
            </div>
            <div class="content">
                <input type="text" v-model="editDetail.title" name="title" value="" placeholder="标题" />
                <input type="text" v-model="editDetail.user" name="user" value="" placeholder="发布人" />
                <input type="date" v-model="editDetail.dates" name="date" value="" placeholder="发布时间" />
                <button @click="update">更新</button>
                <button @click="editlist=false">取消</button>
            </div>
        </div>
    </div>
</div>
```

```
<script>
    var app = new Vue({
        el: '#table',
        data: {
            addDetail: {},
            editlist: false,
            editDetail: {},
            newsList: [{
                title: '招聘前端工程师',
                user: '张凯',
                dates: '2020-08-10',
                id: "10001"
            }, {
                title: '招聘PHP工程师',
                user: '王军',
                dates: '2020-08-15',
                id: "10002"
            }],
            editid: ''
        },
        mounted() { },
        methods: {
            //新增功能
            add: function () {
                this.newsList.push({
                    title: this.addDetail.title,
                    user: this.addDetail.user,
                    dates: this.addDetail.dates,
                })
            },
            //删除功能
            deletelist: function (id, i) {
                this.newsList.splice(i, 1);
            },
            //编辑功能
            edit: function (item) {
                this.editDetail = {
                    title: item.title,
                    user: item.user,
                    dates: item.dates,
                    id: item.id
                }
                this.editlist = true
                this.editid = item.id
            },
```

```
203                 //确认更新
204                 update: function () {
205                     let _this = this
206                     for (let i = 0; i < _this.newsList.length; i++) {
207                         if (_this.newsList[i].id == this.editid) {
208                             _this.newsList[i] = {
209                                 title: _this.editDetail.title,
210                                 user: _this.editDetail.user,
211                                 dates: _this.editDetail.dates,
212                                 id: this.editid
213                             }
214                             this.editlist = false;
215                         }
216                     }
217                 }
218             }
219         })
220     </script>
```