

你可以用 `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。

`v-model` 会忽略所有表单元素的 `value`、`checked`、`selected` attribute 的初始值而总是将 Vue 实例的数据作为数据来源。你应该通过 JavaScript 在组件的 `data` 选项中声明初始值。

`v-model` 在内部为不同的输入元素使用不同的 property 并抛出不同的事件：

- `text` 和 `textarea` 元素使用 `value` property 和 `input` 事件；
- `checkbox` 和 `radio` 使用 `checked` property 和 `change` 事件；
- `select` 字段将 `value` 作为 prop 并将 `change` 作为事件。

对于需要使用 (如中文、日文、韩文等) 的语言，你会发现 `v-model` 不会在输入法组合文字过程中得到更新。如果你也想处理这个过程，请使用 `input` 事件。

文本

```
<input v-model="message" placeholder="edit me"> <p>Message is: {{ message }}</p>
```

Message is:

多行文本

```
<span>Multiline message is:</span> <p style="white-space: pre-line;">{{ message }}</p>
```

```
<br> <textarea v-model="message" placeholder="add multiple lines"></textarea>
```

Multiline message is:

在文本区域插值 (`<textarea>{{text}}</textarea>`) 并不会生效，应用 `v-model` 来代替。

复选框

单个复选框，绑定到布尔值：

```
<input type="checkbox" id="checkbox" v-model="checked"> <label for="checkbox">{{ checked }}</label>
```

false

多个复选框，绑定到同一个数组：

```
<input type="checkbox" id="jack" value="Jack" v-model="checkedNames"> <label for="jack">Jack</label> <input type="checkbox" id="john" value="John" v-model="checkedNames"> <label for="john">John</label> <input type="checkbox" id="mike" value="Mike" v-model="checkedNames"> <label for="mike">Mike</label> <br> <span>Checked names: {{ checkedNames }}</span>
```

```
new Vue({ el: '...', data: { checkedNames: [] } })
```

Jack John Mike

Checked names: []

单选按钮

```
<div id="example-4"> <input type="radio" id="one" value="One" v-model="picked"> <label for="one">One</label> <br> <input type="radio" id="two" value="Two" v-model="picked"> <label for="two">Two</label> <br> <span>Picked: {{ picked }}</span> </div>
```

```
new Vue({ el: '#example-4', data: { picked: '' } })
```

One

Two

Picked:

选择框

单选时:

```
<div id="example-5"> <select v-model="selected"> <option disabled value="">请选择
</option> <option>A</option> <option>B</option> <option>C</option> </select>
<span>Selected: {{ selected }}</span> </div>
```

```
new Vue({ el: '...', data: { selected: '' } })
```

Selected:

如果 `v-model` 表达式的初始值未能匹配任何选项, `<select>` 元素将被渲染为“未选中”状态。在 iOS 中, 这会使用户无法选择第一个选项。因为这样的情况下, iOS 不会触发 `change` 事件。因此, 更推荐像上面这样提供一个值为空的禁用选项。

多选时 (绑定到一个数组):

```
<div id="example-6"> <select v-model="selected" multiple style="width: 50px;">
<option>A</option> <option>B</option> <option>C</option> </select> <br> <span>Selected:
{{ selected }}</span> </div>
```

```
new Vue({ el: '#example-6', data: { selected: [] } })
```

Selected: []

用 `v-for` 渲染的动态选项:

```
<select v-model="selected"> <option v-for="option in options" v-
bind:value="option.value"> {{ option.text }} </option> </select> <span>Selected: {{
selected }}</span>
```

```
new Vue({ el: '...', data: { selected: 'A', options: [ { text: 'One', value: 'A' }, {
text: 'Two', value: 'B' }, { text: 'Three', value: 'C' } ] } })
```

Selected: A

值绑定

对于单选按钮, 复选框及选择框的选项, `v-model` 绑定的值通常是静态字符串 (对于复选框也可以是布尔值):

```
<!-- 当选中时, `picked` 为字符串 "a" --> <input type="radio" v-model="picked" value="a">
<!-- `toggle` 为 true 或 false --> <input type="checkbox" v-model="toggle"> <!-- 当选中第
一个选项时, `selected` 为字符串 "abc" --> <select v-model="selected"> <option
value="abc">ABC</option> </select>
```

但是有时我们可能想把值绑定到 Vue 实例的一个动态 property 上, 这时可以用 `v-bind` 实现, 并且这个 property 的值可以不是字符串。

## 复选框

```
<input type="checkbox" v-model="toggle" true-value="yes" false-value="no" >
```

```
// 当选中时 vm.toggle === 'yes' // 当没有选中时 vm.toggle === 'no'
```

这里的 `true-value` 和 `false-value` attribute 并不会影响输入控件的 `value` attribute，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，(即“yes”或“no”)，请换用单选按钮。

## 单选按钮

```
<input type="radio" v-model="pick" v-bind:value="a">
```

```
// 当选中时 vm.pick === vm.a
```

## 选择框的选项

```
<select v-model="selected"> <!-- 内联对象字面量 --> <option v-bind:value="{ number: 123 }">123</option> </select>
```

```
// 当选中时 typeof vm.selected // => 'object' vm.selected.number // => 123
```

## 修饰符

### .lazy

在默认情况下，`v-model` 在每次 `input` 事件触发后将输入框的值与数据进行同步 (除了上述输入法组合文字时)。你可以添加 `lazy` 修饰符，从而转为在 `change` 事件\_之后\_进行同步：

```
<!-- 在“change”时而非“input”时更新 --> <input v-model.lazy="msg">
```

### .number

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：

```
<input v-model.number="age" type="number">
```

这通常很有用，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。

### .trim

如果要自动过滤用户输入的首尾空白字符，可以给 `v-model` 添加 `trim` 修饰符：

```
<input v-model.trim="msg">
```

## 在组件上使用 v-model

如果你还不熟悉 Vue 的组件，可以暂且跳过这里。

HTML 原生的输入元素类型并不总能满足需求。幸好，Vue 的组件系统允许你创建具有完全自定义行为且可复用的输入组件。这些输入组件甚至可以和 `v-model` 一起使用！

要了解更多，请参阅组件指南中的[自定义输入组件](#)。