Работы с элементами управления

ДВУСТОРОННЯЯ ПРИВЯЗКА

Двунаправленное связывание используется с элементами управления и позволяет синхронизировать данные приложения с пользовательским интерфейсом.

ДВУСТОРОННЯЯ ПРИВЯЗКА

Для создания двусторонней привязки используется директива v-model:

```
<div id="app">
    <input type="text" v-model="name" /><br>
    <p>Name: {{name}}</p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {name: 'Tom'}
    });
</script>
```

ДВУСТОРОННЯЯ ПРИВЯЗКА

Двусторонняя привязки (директива v-model) облегчает написание и восприятие разметки:

```
<div id="app">
    <input type="text" v-model="number" /><br><br>
    <input type="text" v-bind:value="number"
        v-on:input="number=$event.target.value"/>
    <p>Факториал числа {{number}} равен {{factorial(number)}}</p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {number:1},
        methods:{
            factorial: function(n){
                let result = 1;
                for(let i=1; i<=n; i++) result *=i;
                return result;
            }
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Пример с textarea:

```
<div id="app">
    <label>Введите комментарий
        <textarea v-model="comment"></textarea>
    </label>
    <br>
    <div>
        <h3>Комментарий</h3>
        <p style="white-space: pre-line">{{comment}}</p>
    </div>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            comment:'Комментарий....'
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Пример с input checkbox :

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

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Можно заменить true и false на другие значения:

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

!!! Атрибуты true-value и false-value не влияют на атрибут value тега input (браузеры пропускают невыбранные чекбоксы при отправке форм)

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Группа флажков может быть привязана к одному массиву. В этом случае при отметке флажка его значение попадает в массив:

```
<div id="app">
    <input type="checkbox" value="Tom" v-model="selectedUsers">
    <label>C#</label><br>
      <input type="checkbox" value="Bob" v-model="selectedUsers">
      <label>Bob</label><br>
      <input type="checkbox" value="Sam" v-model="selectedUsers">
      <label>Sam</label><br>
      <input type="checkbox" value="Alice" v-model="selectedUsers">
      <label>Alice</label><br>
      <span>Выбрано: {{selectedUsers}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            selectedUsers:[]
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Переключатели привязываются к одному свойству. При выборе переключателя связанное свойство получает выбранное значение:

```
<div id="app">
    <input type="radio" value="Tom" v-model="user">
    <label>Tom</label><br>
    <input type="radio" value="Bob" v-model="user">
    <label>Bob</label><br>
    <input type="radio" value="Sam" v-model="user">
    <label>Sam</label><br>
    <span>Выбрано: {{ user }}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            user:''
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Пример связывания значения value с динамическим свойством экземпляра Vue:

```
<div id="app">
    <input type="radio" v-bind:value="variantName1"
        v-model="user">
    <label>{{variantName1}}</label><br>
    <input type="radio" v-bind:value="variantName2"
        v-model="user">
    <label>{{variantName2}}</label><br>
    <span>Выбрано: {{ user }}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            user:'', variantName1: 'Tom', variantName2: 'Bob'
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Каждый из переключателей может быть привязан к сложному объекту:

```
<div id="app">
    <input type="radio" v-bind:value="{name:'Tom', age:22}"
v-model="user">
    <label>Tom</label><br>
    <input type="radio" v-bind:value="{name:'Bob', age:25}"
v-model="user">
    <label>Bob</label><br>
    <input type="radio" v-bind:value="{name:'Sam', age:28}"
v-model="user">
    <label>Sam</label><br>
    <span>Выбрано: {{user.name}} - {{user.age}}</span>
</div>
<script>
    var app = new Vue({
        el: '#app',
        data: {
            user:{}
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Выпадающий список с одиночным выбором можно привязать к свойству Vue, которое хранит выбранный элемент:

```
<div id="app">
    <select v-model="user">
        <option>Tom</option>
        <option>Bob</option>
        <option>Sam</option>
    </select>
    <span>Выбрано: {{user}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            user:''
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Список select с множественным выбором можно связать с массивом, в который попадут все выбранные значения:

```
<div id="app">
    <select v-model="users" multiple>
        <option>Tom</option>
        <option>Bob</option>
        <option>Sam</option>
        <option>Alice</option>
    </select>
    <span>Выбрано: {{users}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users:[]
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Пример связывания значения value с динамическим свойством экземпляра Vue:

```
<div id="app">
    <select v-model="selectedUnit">
        <option v-bind:value="metre">м</option>
        <option v-bind:value="kilogram">кг</option>
        <option v-bind:value="second">с</option>
    </select>
    <span>Выбрано: {{user}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            selectedUnit:'', metre: 'метр',
            kilogram: 'килограмм', second: 'секунда'
        }
    });
</script>
```

ДИРЕКТИВА V-FOR

Для рендеринга коллекций предназначена директива **v-for**. Она имеет следующий синтаксис: `v-for="item in items"`

Пример:

```
<div id="app">
    <ul>
        <li v-for="phone in phones">{{phone}}</li>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            phones: ['iPhone 7', 'Galaxy S8', 'Nokia N8',
'Xiaomi Mi6']
        }
    });
</script>
```

ДИРЕКТИВА V-FOR

Подобным образом можно выводить и более сложные объекты из массива:

```
<div id="app">
    <ul>
    <li v-for="user in users"><p>Имя: {{user.name}}</p>
                              <p>Возраст: {{user.age}}</p></li>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users: [
                    {name:'Tom', age:22},
                    {name:'Bob', age:31},
                    {name: 'Sam', age:28}
            ]
        }
    });
</script>
```

ДИРЕКТИВА V-FOR

При переборе элементов с помощью дополнительного параметра в v-for мы можем получить индекс элемента в массиве, используя следующий синтаксис: v-for="(element, index) in array"

```
<div id="app">
    <ul>
        <li v-for="(phone, index) in phones">{{index + 1}}.
{{phone}}</li>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            phones: ['iPhone 7', 'Galaxy S8', 'Nokia N8',
'Xiaomi Mi6']
        }
    });
</script>
```

ДИРЕКТИВА V-FOR

Подобно мы можем перебирать и все свойства одного объекта с помощью синтаксиса: v-for="(value, property) in obj"

```
<div id="app">
    <ul>
      <li v-for="user in users">
         <p v-for="(value,key) in user">{{key}}: {{value}}</p>
      </li>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users: [{name:'Tom', age:22},
                    {name:'Bob', age:31},
                    {name: 'Sam', age:28}]
        }
    });
</script>
```

ДИРЕКТИВА V-FOR

Если необходимо, чтобы для каждого объекта из массива создавалось несколько html-элементов, то блок этих элементов следует обертывать элементом template:

```
<div id="app">
    <ul>
      <template v-for="user in users">
        <li>Name: {{user.name}}</li><li>Age: {{user.age}}</li>
      </template>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users: [{name:'Tom', age:22},
                    {name:'Bob', age:31},
                    {name: 'Sam', age:28}]
        }
    });
</script>
```

ДИРЕКТИВА V-FOR

С помощью v-for можно перебрать все числа от 1 до определенного значения. Например, перебор всех чисел от 1 до 10:

```
<div id="app">
  <span v-for="n in 10">{{ n }} </span>
</div>
<script>
    let app = new Vue({
        el: '#app'
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Флажок может быть привязан к целому объекту:

```
<div id="app" class="col-md-4">
 <template v-for="user in users">
  <input type="checkbox" v-bind:value="user" v-model="selectedUsers">
  <label>{{user.name}}</label><br>
 </template>
 <ul>
  <li v-for="user in selectedUsers">{{user.name}} - {{user.age}}</li>
 </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users:[ {name:'Tom', age:22}, {name:'Bob', age:25},
                {name:'Sam', age:28}, {name:'Alice', age:26}],
            selectedUsers:[]
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Пример хранения данных в массиве и динамического создания по элементам массива группу радиокнопок:

```
<div id="app" class="col-md-4">
    <template v-for="user in users">
        <input type="radio" v-bind:value="user" v-
model="selectedUser">
        <label>{{user.name}}</label><br>
    </template>
    <span v-if="selectedUser!==null">Выбрано: {{selectedUser.name}} -
{{selectedUser.age}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users:[ {name:'Tom', age:22}, {name:'Bob', age:25},
                {name:'Sam', age:28}, {name:'Alice', age:26}],
            selectedUser:null
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Для динамического создания списка из массива применяется директива v-for, а для привязки значения к элементу списка - атрибут v-bind:value:

```
<div id="app">
    <select v-model="selectedUser">
        <option v-for="user in users"
v-bind:value="user.name">{{user.name}}</option>
    </select>
    <span>Выбрано: {{selectedUser}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users:[{name:'Tom', age:22}, {name:'Bob', age:25},
              {name:'Sam', age:28}, {name:'Alice', age:26}],
            selectedUser:''
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Можем выбирать и весь объект в целом:

```
<div id="app">
    <select v-model="selectedUser">
        <option v-for="user in users" v-
bind:value="user">{{user.name}}</option>
    </select>
    <span v-if="selectedUser!==null">Выбрано:
{{selectedUser.name}} - {{selectedUser.age}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users:[{name:'Tom', age:22}, {name:'Bob', age:25},
              {name:'Sam', age:28}, {name:'Alice', age:26} ],
            selectedUser:null
        }
    });
</script>
```

РАБОТА С ЭЛЕМЕНТАМИ ВВОДА

Можно установить привязку к объектам в списке с множественным выбором:

```
<div id="app">
    <select v-model="selectedUsers" multiple>
        <option v-for="user in users"
v-bind:value="user">{{user.name}}</option>
    </select>
    <h3>Выбранные объекты</h3>
    <ul><li v-for="user in selectedUsers">{{user.name}} -
{{user.age}}</li></ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            users:[{name:'Tom', age:22}, {name:'Bob', age:25},
              {name:'Sam', age:28}, {name:'Alice', age:26} ],
            selectedUsers:[]
        }
    });
</script>
```

МОДИФИКАТОРЫ

По умолчанию директива **v-model** обновляет связанные с полем ввода данные при каждом вызове события **input** (ввод символа). Вместо этого мы можем вызывать обновление не после события **input**, а после события **change** (потеря фокуса). Для этого применяется модификатор **.lazy**:

```
<div id="app" class="col-md-4">
    <input type="text" v-model.lazy="name"/>
    <span>Введено: {{name}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            name:''
        }
    });
</script>
```

МОДИФИКАТОРЫ

Модификатор **.number** позволяет автоматически конвертировать при вводе вводимое значение в число:

```
<div id="app">
    <input v-model.number="age" type="number">
    <span>Введено: {{age}}</span>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            age: 0
        }
    });
</script>
```

МОДИФИКАТОРЫ

Модификатор **.trim** автоматически обрезает начальные и концевые пробелы при вводе строки:

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

!!! Можно объединять модификаторы.

СПЕЦИАЛЬНЫЙ АТРИБУТ KEY

Атрибут key помогает алгоритму работы с виртуальным DOM определить, какие ноды соответствует какой строке данных. Если этот атрибут не указан, Vue использует алгоритм, минимизирующий перемещение элементов и старается обновить и повторно использовать как можно больше элементов правильного типа «на местах». А если ключ указан, Vue изменяет порядок следования элементов, основываясь на изменении порядка ключей. Элементы, соответствующие ключам, которых больше нет, Vue всегда уничтожает.

!!! Потомки одного и того же общего родителя должны иметь уникальные ключи. Повторяющиеся ключи приведут к ошибкам при отрисовке

СПЕЦИАЛЬНЫЙ АТРИБУТ KEY

Пример использования:

```
<div id="app">
    <ul>
        <li v-for="item in items"
            :key="item.id">{{item.text}}</li>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            items:[{id:1, text:'First'},
                {id:2, text:'Second'}]
        }
    });
</script>
```

ДОПОЛНИТЕЛЬНЫЕ ССЫЛКИ

Официальное руководство по Vue.js:
https://ru.vuejs.org/v2/guide/

Руководство по Vue.js (онлайн учебник)
https://metanit.com/web/vuejs/