Особенности работы с массивами данных

ДИРЕКТИВА V-IF

С помощью директивы **v-else-if** к **v-if** можно добавить обработку дополнительных условий:

```
<div id="app">
    <input type="number" v-model="number" />
    <p v-if="number==1">Один</p>
    <p v-else-if="number==2">Два</p>
    <p v-else-if="number>2 && number<7"> Несколько</p>
    <p v-else>Много</p>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            number: 1
        }
    });
</script>
```

ПРИВЯЗКА КЛАССОВ

С помощью привязки атрибута class во Vue.js можно динамически управлять классами элементов. Для условной привязки классов определим следующую страницу:

```
<div id="app">
    <div class="region" v-bind:class="{active:
isActive}" v-on:click="isActive=!isActive"></div>
    <div class="region"></div>
</div>
<script>
    let app = new Vue({
        el: "#app",
        data: {isActive : false}
    });
</script>
```

ПРИВЯЗКА КЛАССОВ

Установка множества классов через один вычисляемый объект:

```
<div id="app">
    <div class="region" v-bind:class="classObj"></div>
</div>
<script>
    let app = new Vue({
        el: "#app",
        data: {
            isActive : false,
            isBounded: false
        },
        computed: {
            classObj: function () {
                return { active: this.isActive,
                    bounded: this.isBounded  }
            }
        }
    });
</script>
```

ПРИВЯЗКА КЛАССОВ

Можно выполнить простую привязку к свойству, которое хранит название класса:

```
<div id="app">
    <div class="region" v-bind:class="color"></div>
    <input type="text" v-model="color" />
</div>
<script>
    let app = new Vue({
        el: "#app",
        data: {
            color: ""
        }
    });
</script>
```

ПРИВЯЗКА КЛАССОВ

Если надо установить привязку сразу к нескольким классом, то их можно перечислить в квадратных скобках:

```
<div id="app">
    <div class="region" v-bind:class="[color, {bounded: 
isBounded}, size]" v-on:click="isBounded=!isBounded"></div>
    <br>
    <input type="text" v-model="color" />
    <br>
    <input type="text" v-model="size" />
</div>
<script>
    let app = new Vue({
        el: "#app",
        data: {
            color: "red",
            size: "small",
            isBounded: false
        }
    });
</script>
```

ПРИВЯЗКА К СТИЛЯМ

Для привязки к стилям атрибуту style можно передать объект, который содержит стилевые свойства и их значения:

```
<div id="app">
    <div class="region" v-bind:style="{'background-color':
color, height: height + 'px', width: width+'px'}"></div>
</div><br><br>
    <input type="text" v-model="color" /><br><br>
    <input type="number" v-model="width" /><br><br>
    <input type="number" v-model="height" />
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            color: 'red',
            width: 75,
            height: 75
        }
    });
</script>
```

ПРИВЯЗКА К СТИЛЯМ

Отдельное вычисляемое свойство для задания стилей:

```
<div id="app">
    <div class="region" v-bind:style="styleObj" ></div><br><br>
    <input type="text" v-model="color" /><br><br>
    <input type="number" v-model="width" /><br><br>
    <input type="number" v-model="height" />
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: { color: 'red', width: 75, height: 75 },
        computed:{
            styleObj: function(){
                return {
                    'background-color': this.color,
                    height: this.height + 'px',
                    width: this.width+'px'
                }
    }   }   });
</script>
```

ПРИВЯЗКА К СТИЛЯМ

Если необходимо установить для элемента сразу несколько подобных стилей, то они передаются в квадратные скобки:

```
<div id="app">
    <div class="region" v-bind:style="[colorStyle, sizeStyle]"></div><br><br>
    <input type="text" v-model="color" /><br><br>
    <input type="number" v-model="width" /><br><br>
    <input type="number" v-model="height" />
</div>
<script>
    var app = new Vue({
        el: '#app',
        data: { color: 'red', width: 75, height: 75 },
        computed:{
            colorStyle: function(){
                return { 'background-color': this.color }
            },
            sizeStyle: function(){
                return {
                    height: this.height + 'px',
                    width: this.width+'px'
                }
    }   }   });
</script>
```

УПРАВЛЕНИЕ МАССИВАМИ

Во Vue.js для каждого массива определено ряд методов, с помощью которых можно управлять элементами массива:

- push(),
- pop(),
- shift(),
- unshift(),
- splice(),
- sort(),
- reverse().

Эти методы являются обертками над одноименными стандартными методами javascript информирующие Vue о изменении массива.

УПРАВЛЕНИЕ МАССИВАМИ (ДОБАВЛЕНИЕ И УДАЛЕНИЕ)

```
<div id="app">
    <p>
        <input type="text" v-model="newPhone" />
        <button v-on:click="phones.push(newPhone)">Добавить</button>
    </p>
    <ul>
      <li v-for="(phone, index) in phones">
      <p>{{ phone }} <button v-on:click="phones.splice(index, 1)">
Удалить</button></p>
    </ul>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            newPhone: '',
            phones: ['iPhone 7', 'Galaxy S8', 'Nokia N8', 'Xiaomi
Mi6']
        }
    });
</script>
```

УПРАВЛЕНИЕ МАССИВАМИ

Будьте осторожны: Vue не способен отследить некоторые изменения. Это относится, например,
к прямому присваиванию значений
`this.array[index] = newValue`
и обновлению длинны
`this.array.length = newlength`

```
let vm = new Vue({
  data: {
    items: ['a', 'b', 'c']
  }
})
vm.items[1] = 'x'; // НЕ РЕАКТИВНО
vm.items.length = 2; // НЕ РЕАКТИВНО
```

УПРАВЛЕНИЕ МАССИВАМИ (УСТАНОВКА ЭЛЕМЕНТА В МАССИВЕ)

Для установки значения нам надо использовать метод **Vue.set()**:
Vue.set(массив, индекс_элемента, новое_значение);

```
<div id="app">
    <ul>
      <li v-for="phone in phones">
      <p>{{ phone }}</p>
    </ul>
    <button v-on:click="updateList">Обновить</button>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            newPhone: '',
            phones: ['iPhone 7', 'Galaxy S8', 'Nokia N8', 'Xiaomi Mi6']
        },
        methods:{
            updateList: function(){
                Vue.set(this.phones, 1, 'Samsung Galaxy S8');
            }
        }
    });
</script>
```

УПРАВЛЕНИЕ МАССИВАМИ (ВОЗВРАЩЕНИЕ НОВОГО МАССИВА)

```
<div id="app">
    <ul>
      <li v-for="phone in phones">
      <p>{{ phone }}</p>
    </ul>
    <button v-on:click="updateList">Обновить</button>
</div>
<script>
    let app = new Vue({
        el: '#app',
        data: {
            newPhone: '',
            phones: ['iPhone 7', 'Galaxy S8', 'Nokia N8', 'Xiaomi Mi6']
        },
        methods:{
            updateList: function(){
                this.phones = this.phones.slice(1,3);
            }
        }
    });
</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/