

Циклы

- Циклы и одномерные массивы
- Циклы и многомерные массивы
- Циклы и объекты
- Циклы и объекты в многомерных массивах
- Вложенные циклы
- Директива v-for и тег template
- Директива v-for совместно с директивой v-if
- Директива v-for с диапазоном
- Поддержание состояния

Циклы и одномерные массивы

Для отображения списка элементов на основе массива данных во Vue используется директива **v-for**.

Директива требует специального синтаксиса в форме **elem in array**, где **array** является исходным массивом данных и **elem** является псевдонимом для элемента массива, по которому выполняется итерация.

Пусть в **data** у нас есть элемент **array**, который содержит внутри себя массив с некоторыми данными:

```
data() {  
  return {  
    // массив данных  
    array: ["PHP", "JavaScript", "Python", "Perl", "Java"]  
  }  
}
```

Построим список `ul`, внутри которого разместим каждый элемент нашего массива **array** в отдельный пункт списка **li**. Очевидно, что для выполнения подобных действий необходимо использовать цикл.

Во Vue такой цикл делается следующим образом:

```
<li v-for="elem in array">
```

Происходит здесь следующее. Атрибут **v-for** запускает цикл, строка **elem in array** указывает на то, что мы будем перебирать массив **array**, а **каждый элемент этого массива будет записываться в переменную elem**.

В результате строка: `<li v-for="elem in array">` создаст столько тегов **li**, сколько раз прокрутится цикл.

Внутри **li** будет доступна переменная **elem**, в которой каждый раз будет содержаться новый элемент массива. Мы можем вывести ее содержимое с помощью фигурных скобок: `{{ elem }}`. Эта переменная может иметь любое придуманное нами имя, как назовем - так и будем пользоваться (`elem` это не обязательное имя переменной).

Обычно я делаю так. Если значение массива простой тип данных, переменную цикла я называю **el[em]** или **val[ue]**. Если значение массива сложный тип данных (другой массив или объект) переменную я называю **item**.

Итого, полный код вывода списка будет выглядеть следующим образом:

```
<ul>
  <li v-for="elem in array">
    {{ elem }}
  </li>
</ul>
```

где:

- **array** - исходный массив;
- **elem** - ссылка на текущий элемент массива.

Пример вывода одномерного массива в выпадающий список **select** представлен в следующем демонстрационном примере.

example_1. Цикл для одномерного массива

```
<div id="app">

  <h2>Выбор языка программирования:</h2>

  <select>

    <!-- v-for -->

    <option v-for="elem in langs">{{ elem }}</option>

    <!-- end v-for -->

  </select>

</div>

<script>

  let app = Vue.createApp({

    data() {

      return {

        // одномерный массив данных

        langs: ["PHP", "JavaScript", "Python", "Perl",

          "Java", "C++"]

      }

    }

  }).mount("#app");

</script>
```

Иногда кроме элементов массива, мы хотели бы получить доступ еще и к индексам массива. Для этого вместо **elem in array** следует использовать следующую структуру: **(elem, index) in array**, тогда в переменную **elem** будут попадать элементы массива, а в **index** - их индексы.

Вместо **index** вы можете использовать любое другое имя, как вам будет удобно (например **key**).

example_2. Цикл с доступом к индексам массива

```
<div id="app">

  <h2>Я обязательно стану Гуру:</h2>

  <!-- v-for -->

  <span v-for="(elem, index) in langs">

    <!-- сделаем инкремент для индекса -->

    {{ ++index }} - {{ elem }} <br>

  </span>

  <!-- end v-for -->

</div>

<script>

  let app = Vue.createApp({

    data() {

      return {

        // одномерный массив данных

        langs: ["PHP", "JavaScript", "Python", "Perl",

          "Java", "C++"]

      }

    }

  }).mount("#app");

</script>
```

Вместо **in** разделителем допускается использовать **of**, как в итераторах JavaScript:

```
<div v-for="elem of array"></div>
```

Циклы и многомерные массивы

Одномерные массивы в веб-разработке чаще всего представляют собой **объект предметной области** и выводятся в документ в виде **HTML-карточки** или **строки таблицы**. А вот многомерный массив – это то, с чем приходится работать чаще всего. Ничего сложного, только помните – на каждой итерации получаем ссылку на **вложенный массив**.

example_3. Цикл для многомерного массива

```
<div id="app">

  <h2>Автомобили для Гуру:</h2>

  <table id="table">

    <tr>

      <th>№</th>

      <th>ID</th>

      <th>Модель</th>

      <th>Бренд</th>

      <th>Фото</th>

    </tr>

    <!-- v-for -->

    <!-- перебираем элементы двумерного массива -->

    <tr v-for="item, index in cars">

      <td>{{ ++index }}</td>

      <td>{{ item[0] }}</td>

      <td>{{ item[2] }}</td>

      <td>{{ item[1] }}</td>

      <td></td>

    </tr>

    <!-- end v-for -->

  </table>

</div>
```

```
</table>

</div>

<script>

  let app = Vue.createApp({

    data() {

      return {

        // подключаемый двумерный массив данных

        cars: cars

      }

    }

  });

  // прикручиваем шаблон к компоненту

  const vm = app.mount("#app");

</script>
```

Циклы и объекты

Директиву **v-for** также легко использовать для перебора свойств **объекта**.

Синтаксис директивы при этом не изменится:

```
<li v-for="elem in obj">
```

Можно также указать ссылку для **имени свойства** объекта (ключа):

```
<li v-for="(elem, key) in obj">
```

example_4. Цикл для объекта

```
<div id="app">

  <ul>

    <!-- получаем ссылку на пару: ключ - значение -->
```

```
<li v-for="(elem, key) in car">
    {{ key }} - {{ elem }}
</li>
</ul>
</div>
<script>
    let app = Vue.createApp({
        data() {
            return {
                // подключаемый объект car
                car: car
            }
        }
    }).mount("#app");
</script>
```

Циклы и объекты в многомерных массивах

Если говорить про многомерные массивы, то наиболее часто используемая форма представления многомерного массива – **массив вложенных объектов**.

Образное представление такого массива – обычная таблица, где:

- таблица – **внешний массив**. Где элементами массива являются строки таблицы;
- строка – **вложенный массив**. Где элементами массива является набор именованных ячеек строки.

Можно указать ссылку не только для **ключа**, но и для **индекса** элемента:

```
<li v-for="(elem, key, index) in obj">
```

example_5. Цикл для объектов в многомерном массиве

```
<div id="app">

  <div v-for="(item, key) in cars">

    id: {{ item.id }} <br>

    Бренд : <b>{{ item.brand }}</b> <br>

    Модель : {{ item.model }} <br>

    Двигатель : {{ item.engine }} <br>

    Мощность двигателя : {{ item.power }} <br>

    Тип : {{ item.cylinder }} <br>

    Объем двигателя : {{ item.engineSize }} <br>

    Количество цилиндров : {{ item.numberCylinders }}

    <p></p>

    <hr>

  </div>

</div>

<script>

  let app = Vue.createApp({

    data() {

      return {

        // подключаемый массив объектов cars

        cars: cars

      }

    }

  }).mount("#app");

</script>
```




До сих пор значениями ключа объекта были **простые типы данных** (числа или строки). Рассмотрим пример, когда значением ключа является **массив**.

Протестируем реализацию **вложенных циклов**.

В целях предотвращения избыточного загромождения кода я не стал применять дополнительное форматирование к элементам вывода.

example_6. Вложенные циклы

```
<div id="app">

  <ul>

    <template v-for="(elem, key) in person">

      <li>

        <span v-if="key == 'education'">

          <template v-for="el in elem">

            {{ el.institution }} <br>

          </template>

        </span>

        <span v-else>

          ({{ key }}) {{ elem }}

        </span>

      </li>

    </template>

  </ul>

</div> <!-- // #app -->

<script>

  let app = Vue.createApp({

    data() {
```

```
return {  
    // подключаемый объект  
    person: personnels  
}  
}  
  
}).mount("#app");  
</script>
```

Примечание. Со временем задачу реализации вложенных циклов мы решим более изящно.

Директива v-for и тег template

Аналогично директиве **v-if** для предотвращения появления дополнительных HTML-узлов в разметке документа **можно применить псевдоэлемент <template>**.

Псевдоэлемент <template> служит невидимой обёрткой для других элементов и сам в результатах отрисовки не появляется.

example_7. Цикл с использованием template

```
<div id="app">  
    <h2>Наши преподаватели</h2>  
    <ol>  
        <!-- используем в качестве разделителя of -->  
        <template v-for="person of personnels">  
            <li>  
                <span class='name'> {{ person.surname }} {{  
                    person.name }} </span><br>  
                <span class='add'> {{ person.post }} / Категория:
```

```
        {{ person.category }} </span><br>

        <span class='add'> {{ person.email ? "email: " +
        person.email : "" }} </span>

    </li>

</template>

</ol>

</div>

<script>

    let app = Vue.createApp({
        data() {
            return {
                // подключаемый массив объектов
                personnels: personnels
            }
        }
    });

    const vm = app.mount("#app");

</script>
```

Директива v-for совместно с директивой v-if

Ревьюируем код примера **example_7** таким образом, чтобы в браузер выводились только актуальные преподаватели (actual = 1).

При использовании директив цикла и условий на одном узле, директива **v-if** имеет более высокий приоритет, чем **v-for**. Это означает, что условие **v-if** не будет иметь доступа к переменным цикла из области **v-for**.

На этом основании приведенный ниже код примера **example_8** работать **не будет**.

example_8. Так работать не будет

```
<div id="app">

  <h2>Наши преподаватели</h2>

  <ol>

    <!-- используем в качестве разделителя of -->

    <!--

    пытаемся вывести только актуальных преподавателей

    ! так работать не будет !

    -->

    <template v-for="person of personnels" v-
    if="person.actual">

      <li>

        <!-- код не изменился -->

      </li>

    </template>

  </ol>

</div>

<script>

  let app = Vue.createApp({

    data() {

      return {

        // подключаемый массив объектов

        personnels: personnels

      }

    }

  }).mount("#app");

</script>
```

Ситуацию можно исправить так, как продемонстрировано в следующем примере.

example_9. Цикл совместно с директивой условия v-if

```
<div id="app">

  <h2>Наши преподаватели</h2>

  <ol>

    <!-- используем в качестве разделителя of -->

    <!-- выводим только актуальных преподавателей -->

    <template v-for="person of personnels">

      <li v-if="person.actual">

        <!-- код не изменился -->

      </li>

    </template>

  </ol>

</div>

<script>

  let app = Vue.createApp({

    data() {

      return {

        // подключаемый массив объектов

        personnels: personnels

      }

    }

  }).mount("#app");

</script>
```

Примечание. Из-за существующего приоритета не рекомендуется

использовать директивы **v-if** и **v-for** на одном элементе.

Переменные цикла можно использовать не только в директиве **v-if**. В следующем примере используем переменные для вывода значений **атрибутов** дочерних элементов.

example_10. Переменные цикла в атрибутах дочерних элементов

```
<div id="app">

  <h2>Long Live Rock'N'Roll!</h2>

  <ol>

    <template v-for="item in teams">

      <li>

        <a

          v-bind:id="'id-' + item[0]"

          v-bind:href="item[7]"

          v-bind:title="item[4]"

          target="_blank">

            {{ item[1] }}

          </a>

          <ul>

            <li>Страна: {{ item[3] }}</li>

            <li>Дата основания {{ item[5] }}</li>

            <li>Стиль: {{ item[6] }}</li>

          </ul>

        </li>

      </template>

    </ol>

  </div>
```

```
<script>

let app = Vue.createApp({
  data() { return { teams: teams } }
}).mount("#app");

</script>
```

Директива v-for с диапазоном

Директива **v-for** может принимать целое число. В таком случае шаблон выполнит количество итераций, основываясь на диапазоне **1 ... n**.

```
<li v-for="n in 10"> {{ n }} </li>
```

Необходимо обратить внимание, что в данной конструкции **начальное значение n** начинается не с **0**, а с **1**.

Смотрите пример.

example_11. Вывод диапазона в цикле

```
<div id="app">

  <h2>Формирование диапазона v-for</h2>

  <ul>

    <li v-for="n in 11"> {{ n }} </li>

  </ul>

</div>

<script>

  let app = Vue.createApp().mount("#app");

</script>
```

Поддержание состояния

Когда Vue обновляет список элементов, отрисованных с помощью **v-for**, по умолчанию он использует стратегию **исправления на месте**.

Если порядок элементов данных изменился вместо того, чтобы перемещать элементы DOM в соответствии с порядком элементов, Vue исправит каждый элемент на месте и убедится, что он отражает то, что должно отображаться по этому конкретному индексу.

Чтобы Vue мог отслеживать идентификацию каждого узла и, таким образом, повторно использовать и изменять порядок существующих элементов, вам необходимо предоставить **уникальный key атрибут для каждого элемента**:

```
<span v-for="(item, index) in catalogs" :key="item.id">
  <!-- /// -->
</span>
```

В конце урока, как обычно, задачи для **самостоятельного решения**.

Задача 1

В файле **task_1.html** раздаточного материала вам предложен шаблон сценария. Изучите структуру массива данных, расположенного в файле **teams.js**. Напишите сценарий, выведите данные массива в браузер.

Задача 2

В файле **task_2.html** раздаточного материала вам предложен шаблон сценария. Подключите фреймворк **Vue**. Подключите файл данных **personnels.js**, изучите структуру массива данных. Напишите сценарий вывода: Фамилии, Имени, Отчества преподавателя в HTML-элемент **выпадающий**

список select.

Задача 3

В файле **task_3.html** раздаточного материала вам предложен шаблон сценария. Изучите структуру массива данных, расположенного в файле **personnels.js**. Обратите внимание, массив содержит **список объектов**, у которых в качестве значения ключа **courses** – сложная структура данных (массив).

Напишите сценарий, выведите данные массива в браузер. Для вывода информации о курсах преподавателя используйте **вложенный цикл**.

Примечание. Циклическим выводом сложных массивов данных будем заниматься на протяжении всего курса.

P.S.

Для отработки и закрепления учебного курса **донам** группы предоставляется следующий раздаточный материал.

К каждому уроку курса:

- Файлы **демонстрационного кода** (example);
- **Задачи** с решениями в контексте рассматриваемых вопросов урока (task).

К каждой теме курса:

- **Практические работы**.