

Министерство транспорта Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования

«Российский университет транспорта (МИИТ)» (РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа №5

«Программная инженерия»

На тему:

«Циклы, наблюдатели, витки»

Выполнил: ст. гр. ТУУ-171

Дудкин А.В.

Вариант №4

Проверил: доц. Сафронов А. И.

Москва – 2025

Содержание

| | |
|---|----|
| Цель работы | 2 |
| Описание задачи..... | 2 |
| Спецификация оборудования | 3 |
| Сведения о браузере | 4 |
| Технология локального подключения фреймворка Vue.js | 4 |
| Схема жизненного цикла Vue.js | 7 |
| Таблица соответствия переменных | 8 |
| Содержательная часть | 10 |
| Витки жизненного цикла компонента Vue.js..... | 10 |
| Списки, переборы, v-for | 18 |
| Наблюдатели watch..... | 28 |
| Вывод..... | 35 |

Цель работы

Улучшить навыки работы с фреймворком Vue.js. Получить опыт работы с хуками жизненного цикла компонента Vue, с циклами (Директива v-for) и наблюдателями (watch).

Описание задачи

Разработать локальные одностраничные *web*-приложение (*LSPWA*) под управлением фреймворка *Vue.js* на языке *JavaScript* в соответствии с указаниями вариантов индивидуального задания.

Витки жизненного цикла

Продумать схему тестирования и демонстрации работы всех витков / хуков / методов жизненного цикла фреймворка *Vue.js* в формате одностраничного *web*-приложения, отличающегося от рассмотренного в лекционном материале курса «*Web*-программирование». Реализовать схему тестирования строго под *Vue.js 3.x*

Списки, перебор v-for

Дан кубический массив размерности, указываемой пользователем в `<input>`. Массив заполняется и перезаполняется псевдослучайным образом каждый раз, как только меняется значение в `<input>`, но только после потери этим элементом фокуса. Все генерируемые значения лежат в диапазоне не более, чем с трёхразрядными целыми десятичными значениями. Продумать способ вывода этих значений в обрамлённую таблицу по спирали. Под таблицей для контроля её заполнения выводить те же значения друг за другом в абзац текста `<p></p>` в порядке увеличения индексов в традиционной последовательности чтения книги: столбец, строка, слой. Через разделитель «\$».

Наблюдатели watch

Подобрать шесть идентичных по ширине и высоте иллюстраций, выводимых в `` фиксированного размера. Седьмое изображение тех же размеров содержит надпись «изображение отсутствует». Вводить в `<input>` название (без расширения) иллюстрации. Если оно соответствует одному из шести имеющихся наименований – выводить соответствующую иллюстрацию, а если нет – выводить иллюстрацию с надписью «изображение отсутствует».

Спецификация оборудования

Спецификация оборудования, на котором выполнялись практические задания, представлена в таблице 1.

Таблица 1. Спецификация оборудования

| № п/п | Характеристика | Сведения |
|-------|----------------------|-----------------------------|
| 1 | Тип устройства | ПК |
| 2 | Модель | Пользовательская сборка |
| 3 | Операционная система | <i>Microsoft Windows 11</i> |
| 4 | Процессор | <i>Intel Core i9-12900K</i> |
| 5 | Оперативная память | 32 Гб |
| 6 | Объём жёсткого диска | 2 Тб |
| 7 | Видеокарта | 4070 ti |

Сведения о браузере

Информация об используемом браузере представлена на рисунке 1.

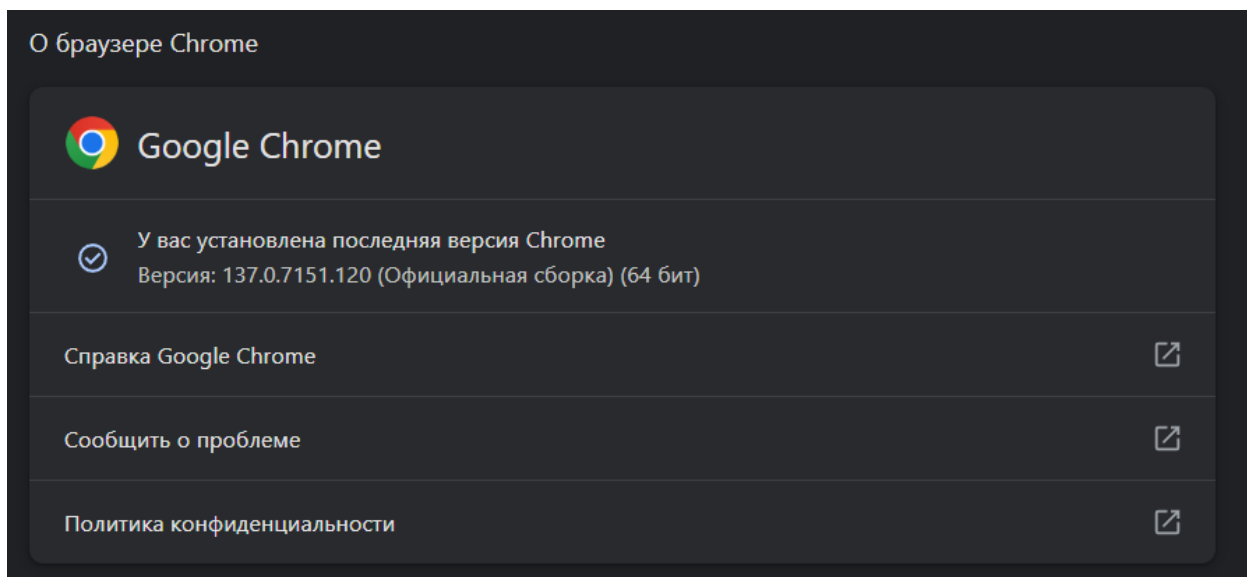


Рисунок 1 – Сведения о браузере Google chrome

Технология локального подключения фреймворка Vue.js

Технология локального подключения фреймворка Vue.js заключается в переносе содержимого фреймворка Vue.js в локальный файл сценария (.js). Для этого необходимо:

1. Перейти к официальной документации [Vue.js](https://vuejs.org/) и нужно нажать на кнопку Install.
2. Перейти к разделу подключения фреймворка (см. рисунок 2)

Using Vue from CDN

You can use Vue directly from a CDN via a script tag:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

Here we are using `unpkg`, but you can also use any CDN that serves npm packages, for example `jsdelivr` or `cdnjs`. Of course, you can also download this file and serve it yourself.

When using Vue from a CDN, there is no "build step" involved. This makes the setup a lot simpler, and is suitable for enhancing static HTML or integrating with a backend framework. However, you won't be able to use the Single-File Component (SFC) syntax.

Рисунок 2 – Подключение фреймворка через CDN (Content Delivery Network)

3. Перейти по ссылке, по которой подключается фреймворк к приложению (см. рисунок 3)

```

/**
 * vue v3.5.16
 * (c) 2018-present Yuxi (Evan) You and Vue contributors
 * @license MIT
 */
var Vue = (function (exports) {
  'use strict';

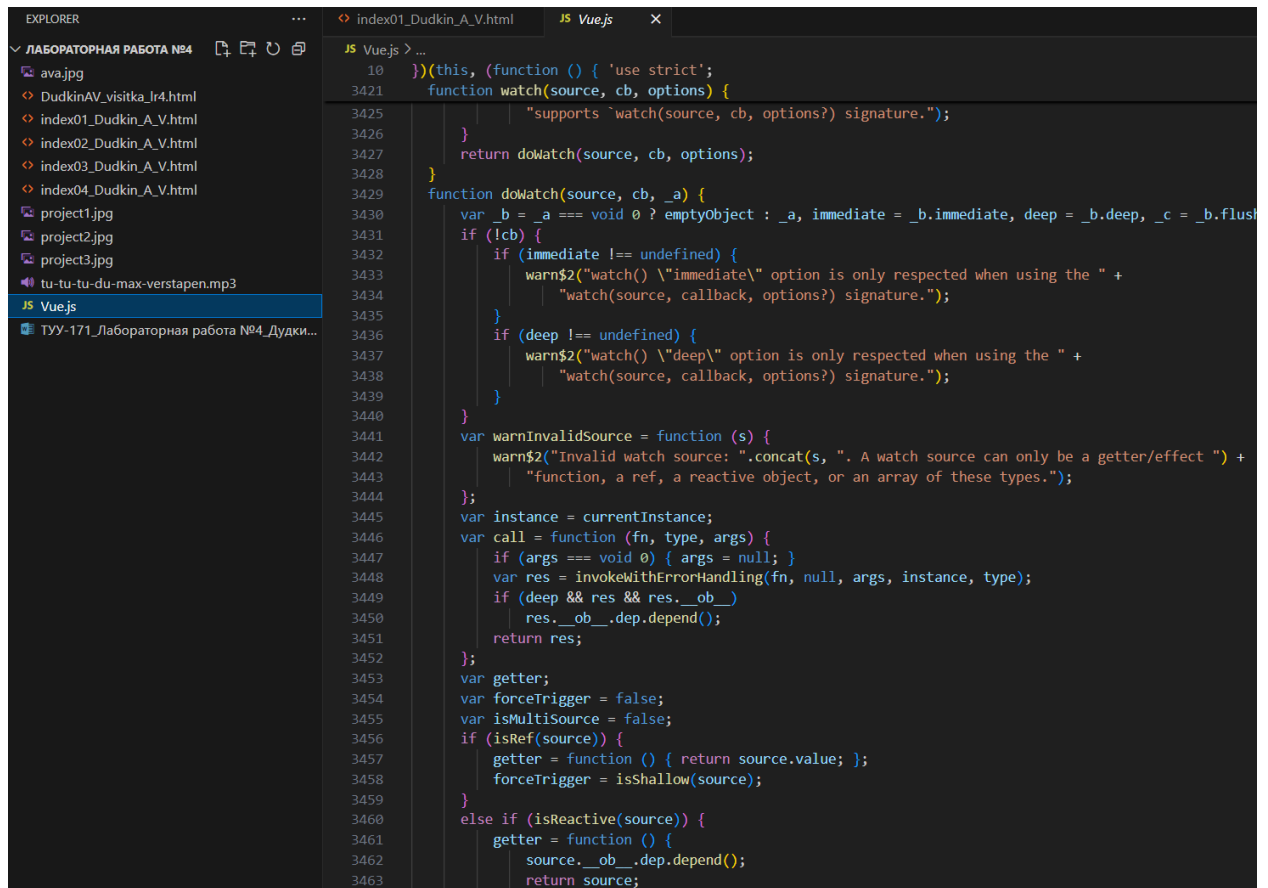
  /!* #__NO_SIDE_EFFECTS__ */
  // @__NO_SIDE_EFFECTS__
  function makeMap(str) {
    const map = /* @__PURE__ */ Object.create(null);
    for (const key of str.split(",")) map[key] = 1;
    return (val) => val in map;
  }

  const EMPTY_OBJ = Object.freeze({}) ;
  const EMPTY_ARR = Object.freeze([]) ;
  const NOOP = () => {}
};
const NO = () => false;
const isOn = (key) => key.charCodeAt(0) === 111 && key.charCodeAt(1) === 110 && // uppercase letter
(key.charCodeAt(2) > 122 || key.charCodeAt(2) < 97);
const isModelListener = (key) => key.startsWith("onUpdate:");
const extend = Object.assign;
const remove = (arr, el) => {
  const i = arr.indexOf(el);
  if (i > -1) {
    arr.splice(i, 1);
  }
};
const hasOwnProperty$1 = Object.prototype.hasOwnProperty;
const hasOwn = (val, key) => hasOwnProperty$1.call(val, key);
const isArray = Array.isArray;
const isMap = (val) => toString(val) === "[object Map]";
const isSet = (val) => toString(val) === "[object Set]";
const isDate = (val) => toString(val) === "[object Date]";
const isRegExp = (val) => toString(val) === "[object RegExp]";
const isFunction = (val) => typeof val === "function";
const isString = (val) => typeof val === "string";
const isSymbol = (val) => typeof val === "symbol";
const isObject = (val) => val !== null && typeof val === "object";
const isPromise = (val) => {
  return (isObject(val) || isFunction(val)) && isFunction(val.then) && isFunction(val.catch);
};
const objectToString = Object.prototype.toString;

```

Рисунок 3 – Страница с содержимым фреймворка

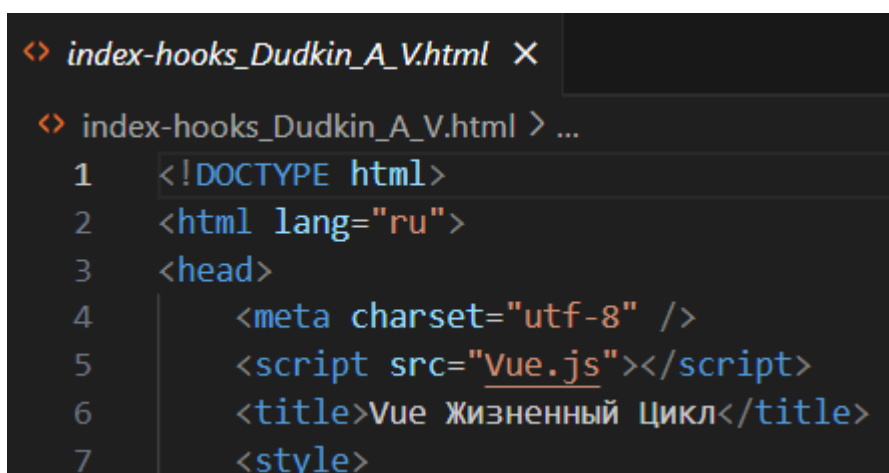
4. Скопировать содержимое страницы в отдельный файл сценария (см. рисунок 4)



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like 'ava.jpg', 'DudkinAV_visitka_ltr4.html', 'index01_Dudkin_A_V.html', 'index02_Dudkin_A_V.html', 'index03_Dudkin_A_V.html', 'index04_Dudkin_A_V.html', 'project1.jpg', 'project2.jpg', 'project3.jpg', 'tu-tu-tu-du-max-verstapen.mp3', and 'JS Vue.js'. The code editor shows the content of 'index01_Dudkin_A_V.html', which is a JavaScript file containing the Vue.js source code. The code is written in a dark theme and includes comments and function definitions.

Рисунок 4 – Локальный файл сценария фреймворка Vue.js

5. В заголовке разметки подключить файл сценария с фреймворком (см. рисунок 5)



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with files like 'index-hooks_Dudkin_A_V.html'. The code editor shows the content of 'index-hooks_Dudkin_A_V.html', which is an HTML file. The code is written in a dark theme and includes the HTML header with the Vue.js script tag.

Рисунок 5 – Локальное подключение фреймворка Vue.js

Схема жизненного цикла Vue.js

Схема жизненного цикла Vue.js представлен на рисунке 6.

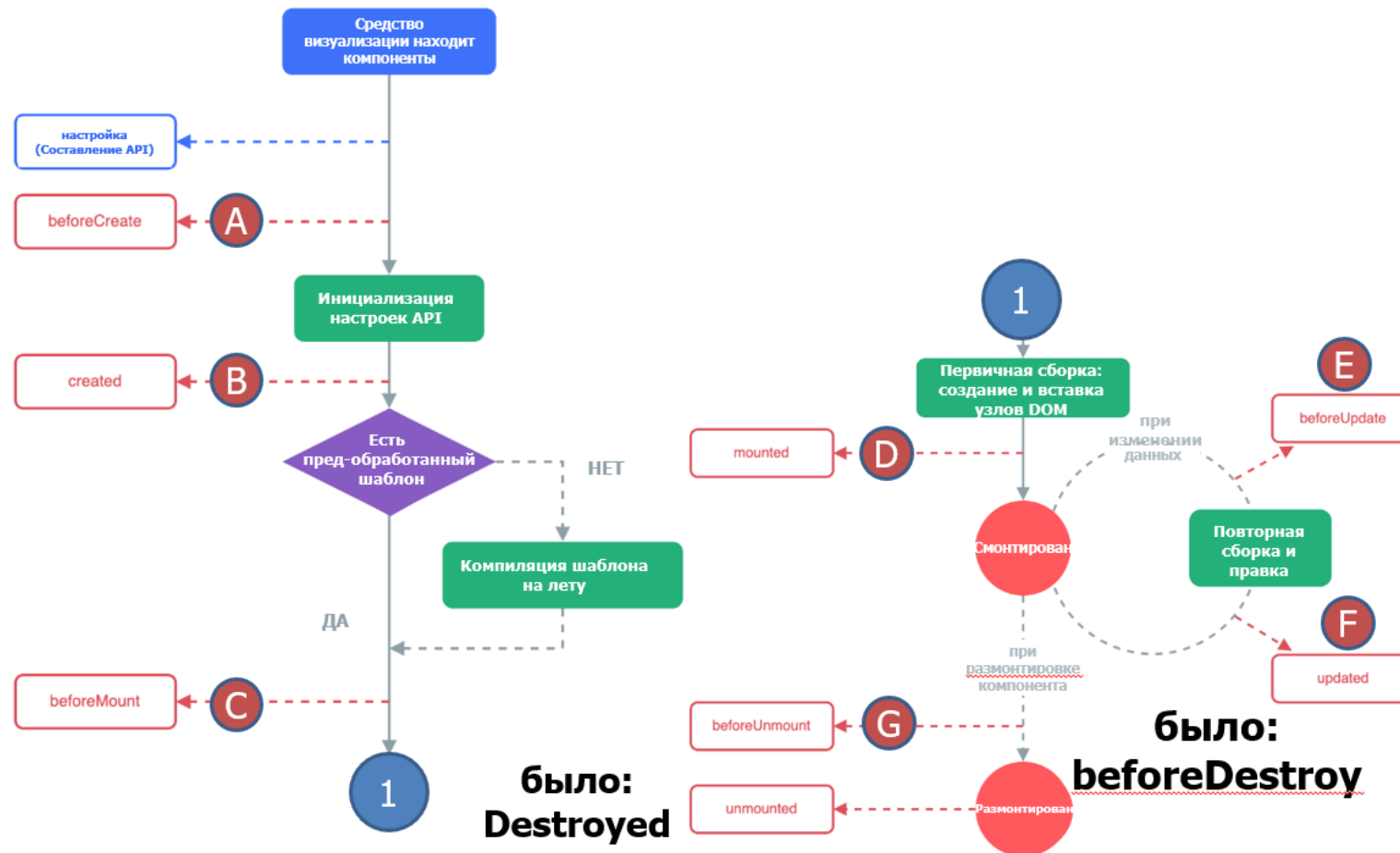


Рисунок 6 – Схема жизненного цикла компонента Vue.js

Таблица соответствия переменных

В таблице 2 представлены переменные, используемые в веб-приложениях.

Таблица 2. Переменные, используемые в веб-приложениях

| Веб-приложение | Методы & Переменные | Описание |
|------------------------|-----------------------|---|
| Витки жизненного цикла | isVisible | Реактивная переменная, определяющая видимость дочернего компонента (LifecycleDemo). |
| | toggleComponent() | Метод, переключающий значение isVisible (показывает/скрывает компонент). |
| | handleAddItem() | Метод, вызываемый при событии @add-item из дочернего компонента (выводит alert). |
| | items (data) | Массив строк, хранящий элементы списка (изначально содержит 'Первый элемент'). |
| | addItem() | Метод, добавляющий новый элемент ('Новый элемент') в массив items. |
| Списки, перебор, v-for | generateRandomValue() | Генерирует случайное число от 100 до 999. |
| | dimension | Реактивная переменная, хранящая размерность 3D-массива (N×N×N). По умолчанию: 2. |
| | cubicArray | Трёхмерный массив (3D), заполненный случайными числами от 100 до 999. |
| | spiralView | Двумерный массив (2D), представляющий спиральное отображение cubicArray. |
| | currentPage | Текущий отображаемый слой 3D-массива (индекс). |
| | currentLayer | Возвращает слой 3D-массива (cubicArray), соответствующий currentPage. |
| | calculateTableSize(n) | Вычисляет размер таблицы для спирального представления. |

Таблица 2. Продолжение

| Веб-приложение | Методы & Переменные | Описание |
|------------------------|---------------------|--|
| Списки, перебор, v-for | generateArray() | Заполняет cubicArray случайными значениями и обновляет spiralView. |
| | updateSpiralView() | Преобразует 3D-массив в спиральное 2D-представление. |
| | nextPage() | Увеличивает currentPage (переход к следующему слою). |
| | prevPage() | Уменьшает currentPage (переход к предыдущему слою). |
| Наблюдатели watch | imageName | Реактивная переменная, хранящая введённое пользователем название изображения. |
| | currentImage | URL текущего отображаемого изображения. |
| | currentImageAlt | Альтернативный текст для текущего изображения. |
| | Images | Массив доступных изображений (URL или пути к файлам). |
| | notFoundImage | URL изображения, которое отображается при отсутствии результата поиска. |
| | imageMapping | Объект, сопоставляющий текстовые запросы (image1, image2 и т.д.) с индексами массива Images. |

Содержательная часть

Витки жизненного цикла компонента Vue.js

Листинг веб-приложения 1 задания представлен:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="utf-8" />
  <script src="Vue.js"></script>
  <title>Vue Жизненный Цикл</title>
  <style>
    body {
      font-family: 'Segoe UI', Tahoma, sans-serif;
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
      background-color: #f5f5f5;
    }
    .container {
      background: white;
      padding: 20px;
      border-radius: 8px;
      box-shadow: 0 2px 10px rgba(0,0,0,0.1);
    }
    button {
      background: #42b983;
      color: white;
      border: none;
      padding: 10px 15px;
      border-radius: 4px;
      cursor: pointer;
      font-size: 16px;
      transition: background 0.3s;
    }
    button:hover {
      background: #369f6e;
    }
    .component-box {
      margin-top: 20px;
      padding: 15px;
      border: 1px dashed #42b983;
      border-radius: 4px;
    }
  </style>
</head>
<body>
  <div class="container">
    <div class="component-box">
      <h1>Витки жизненного цикла компонента Vue.js</h1>
    </div>
  </div>
</body>
</html>
```

```

</style>
</head>
<body>
  <div id="app">
    <div class="container">
      <h1>Демонстрация жизненного цикла Vue</h1>
      <button @click="toggleComponent">
        {{ isVisible ? 'Скрыть' : 'Показать' }} компонент
      </button>

      <div class="component-box" v-if="isVisible">
        <lifecycle-demo @add-item="handleAddItem"></lifecycle-demo>
      </div>
    </div>
  </div>
</div>

<script>
  const { createApp, ref, computed } = Vue;

  const LifecycleDemo = {
    template: `
      <div>
        <button @click="addItem">Добавить элемент</button>
        <div style="margin-top: 15px;">
          <h3>Список элементов:</h3>
          <p v-for="(item, index) in items" :key="index">
            {{ item }} #{{ index + 1 }}
          </p>
        </div>
      </div>
    `,
    data() {
      return {
        items: ['Первый элемент']
      };
    },
    methods: {
      addItem() {
        this.items.push('Новый элемент');
      }
    },
    beforeCreate() {
      alert('1. beforeCreate: Данные ещё не инициализированы');
    },
  }

```

```

created() {
  alert('2. created: Данные доступны, но DOM ещё нет');
},
beforeMount() {
  alert('3. beforeMount: Компонент готов к монтированию');
},
mounted() {
  alert('4. mounted: Компонент добавлен в DOM');
},
beforeUpdate() {
  alert('5. beforeUpdate: Перед обновлением DOM');
},
updated() {
  alert('6. updated: DOM обновлён');
},
beforeUnmount() {
  alert('7. beforeUnmount: Перед удалением компонента');
},
unmounted() {
  alert('8. unmounted: Компонент удалён');
}
};

createApp({
  components: {
    LifecycleDemo
  },
  setup() {
    const isVisible = ref(false);

    const toggleComponent = () => {
      isVisible.value = !isVisible.value;
    };

    const handleAddItem = () => {
      alert('Элемент добавлен в дочернем компоненте');
    };

    return {
      isVisible,
      toggleComponent,
      handleAddItem
    };
  }
})

```

```
    }).mount('#app');  
</script>  
</body>  
</html>
```

При запуске приложения последовательно вызываются следующие витки жизненного цикла Vue.js.

1. beforeCreate
2. created
3. beforeMount
4. mounted

Хуки beforeCreate и created вызываются перед и после инициализации данных, методов и реактивных свойств компонента (см. рисунки 7 и 8).

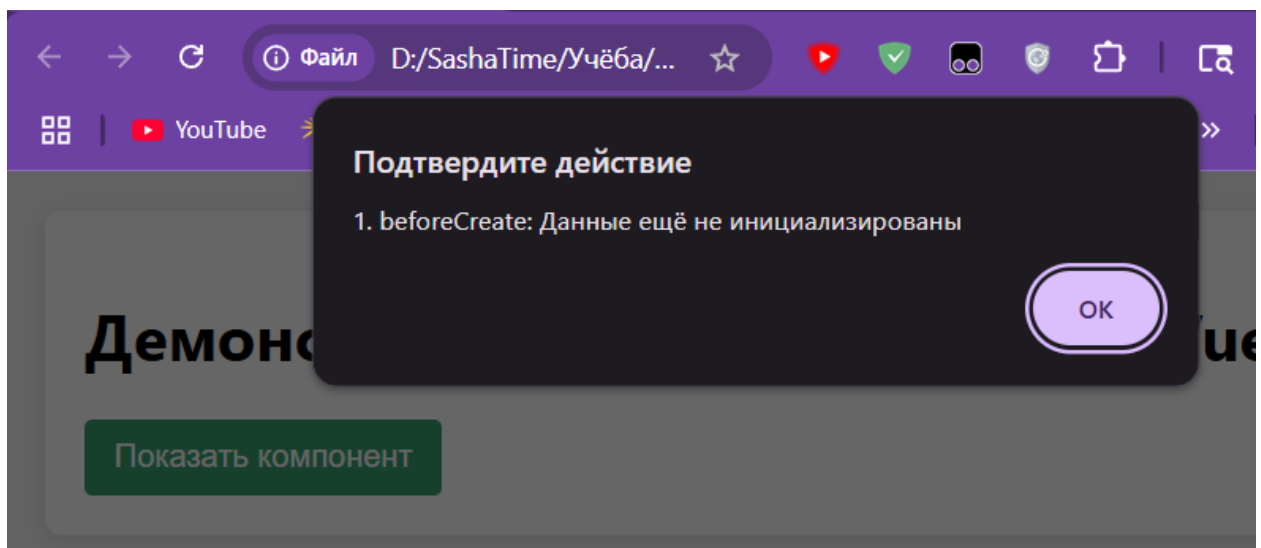


Рисунок 7 – Вызов витка beforeCreate жизненного цикла Vue.js

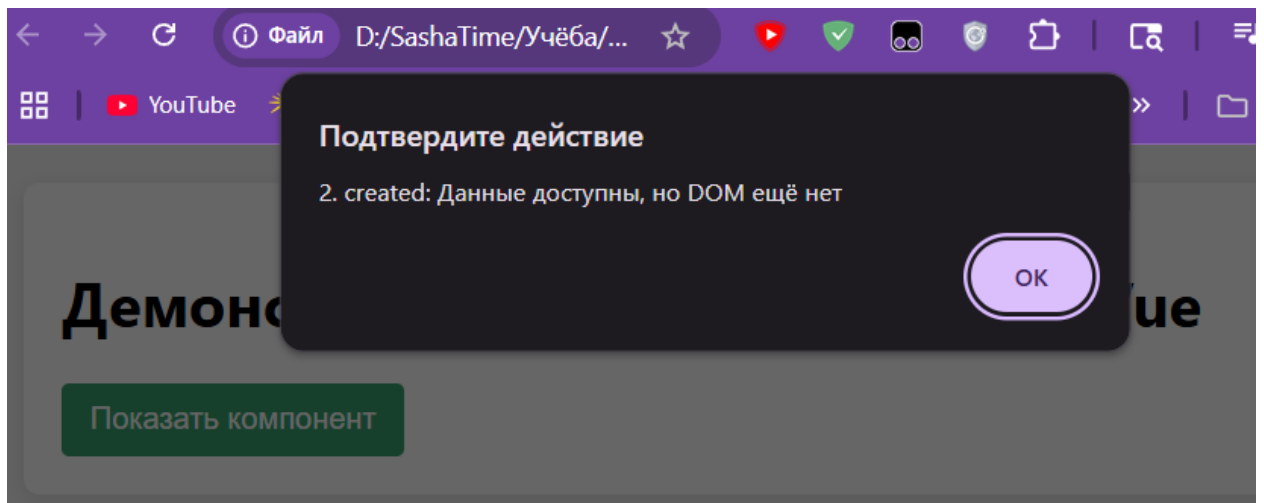


Рисунок 8 – Вызов витка created жизненного цикла Vue.js

Хуки **beforeMount** и **mounted** вызываются перед и после «прикрепления» разметки компонента к DOM-дереву (см. рисунки 9 и 10).

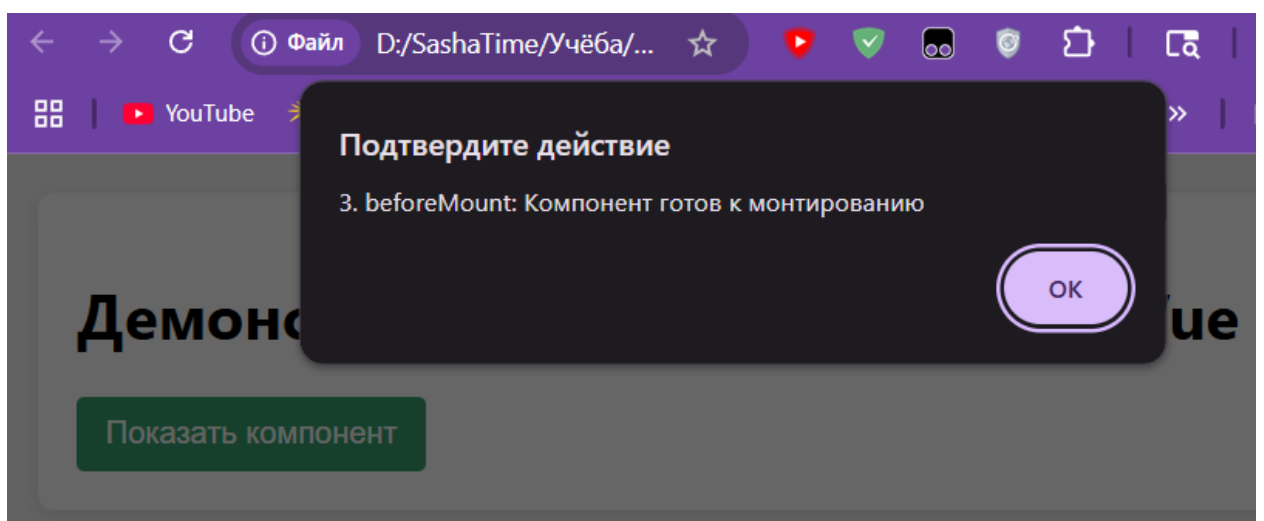


Рисунок 9 – Вызов витка beforeMount жизненного цикла Vue.js

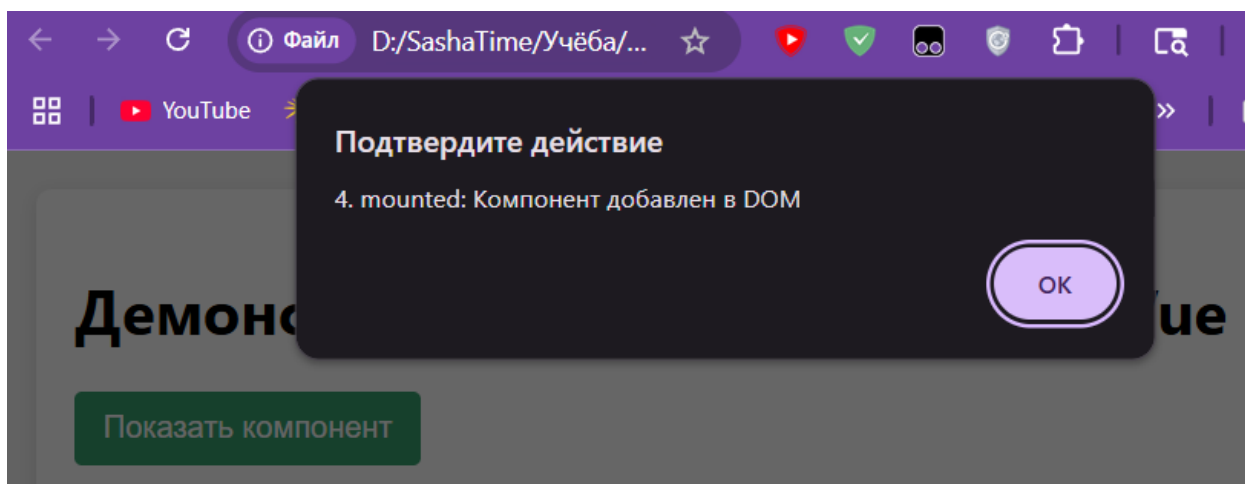


Рисунок 10 – Вызов витка mounted жизненного цикла Vue.js

После отработки перечисленных витков жизненного цикла Vue.js на странице отображается DOM-дерево (см. рисунок 11)

Демонстрация жизненного цикла Vue

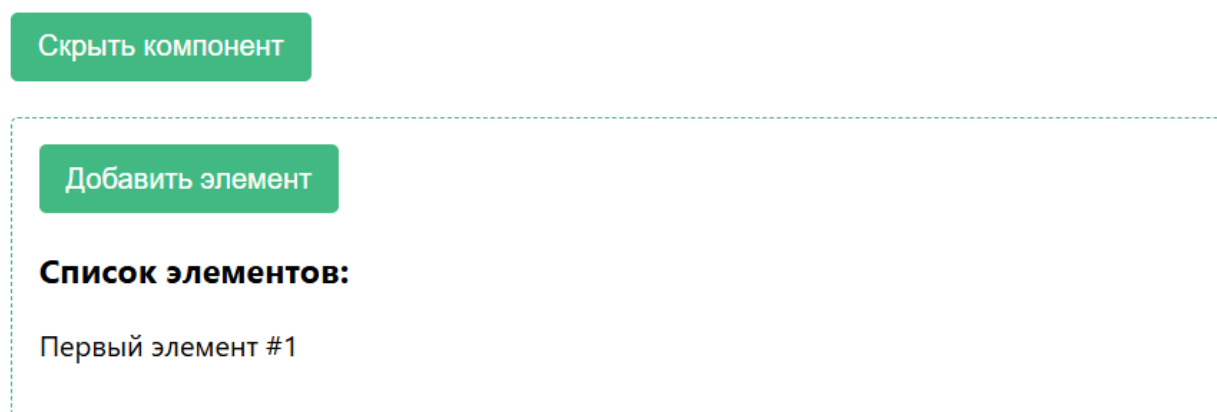


Рисунок 11 – Исходное состояние веб-приложения демонстрации хуков жизненного цикла Vue.js

Для вызова витков beforeUpdate и Updated нажмём на кнопку «Добавить элемент» компонент Vue» (см. рисунки 12 и 13) – оно добавит в разметку компонента слово. Во время обновления строится виртуальное DOM-дерево и сравнивается с реальным – в случае, если между ними есть различия, в обновлённую часть DOM-дерева внесутся изменения.

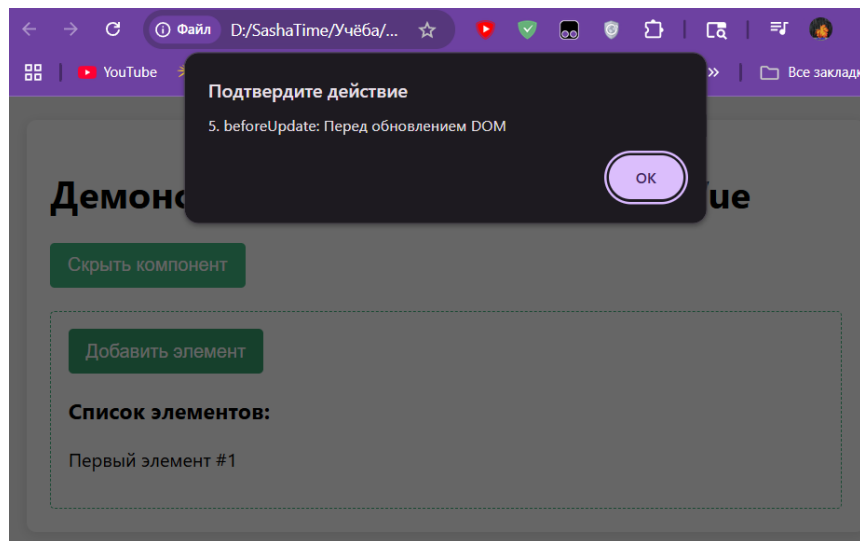


Рисунок 12 – Вызов витка beforeUpdate жизненного цикла Vue.js

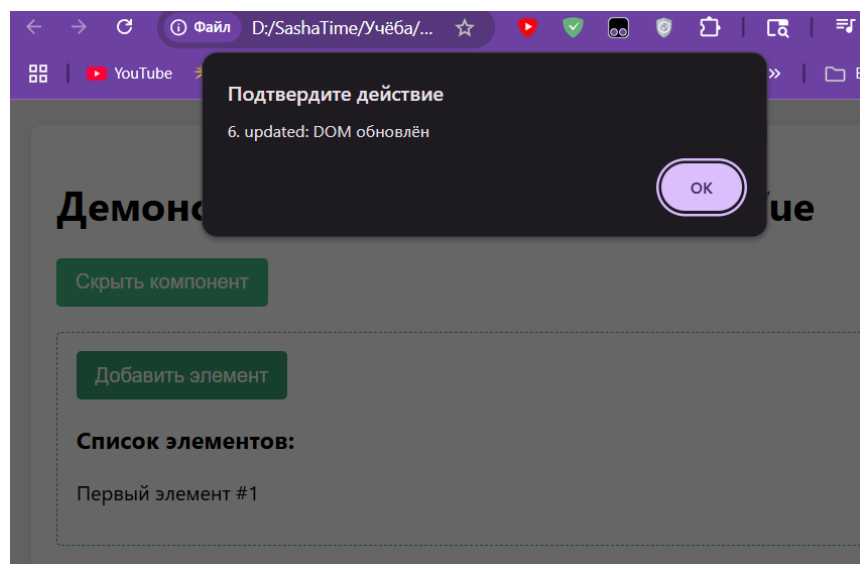


Рисунок 13 – Вызов витка Updated жизненного цикла Vue.js

После отработки хуков обновления компонента Vue на странице будет отображено DOM-дерево с добавленным словом (см. рисунок 14).

Демонстрация жизненного цикла Vue

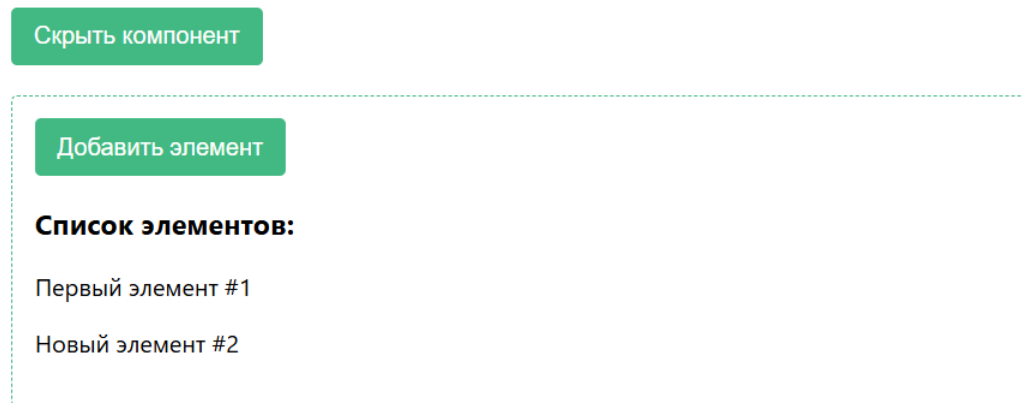


Рисунок 14 – Отображение веб-приложения демонстрации хуков жизненного цикла Vue.js с добавленным словом

Для вызова витков `beforeUnmount` и `Unmounted` нажмём на кнопку «Скрыть компонент Vue» (см. рисунки 15 и 16) – оно исключит из DOM-дерева разметку компонента Vue. Во Vue.js 3-й версии это можно реализовать через условную директиву `v-if` – в отличие от `v-show`, которая манипулирует стилями для отображения разметки компонента, она полностью удаляет компонент из DOM-дерева.

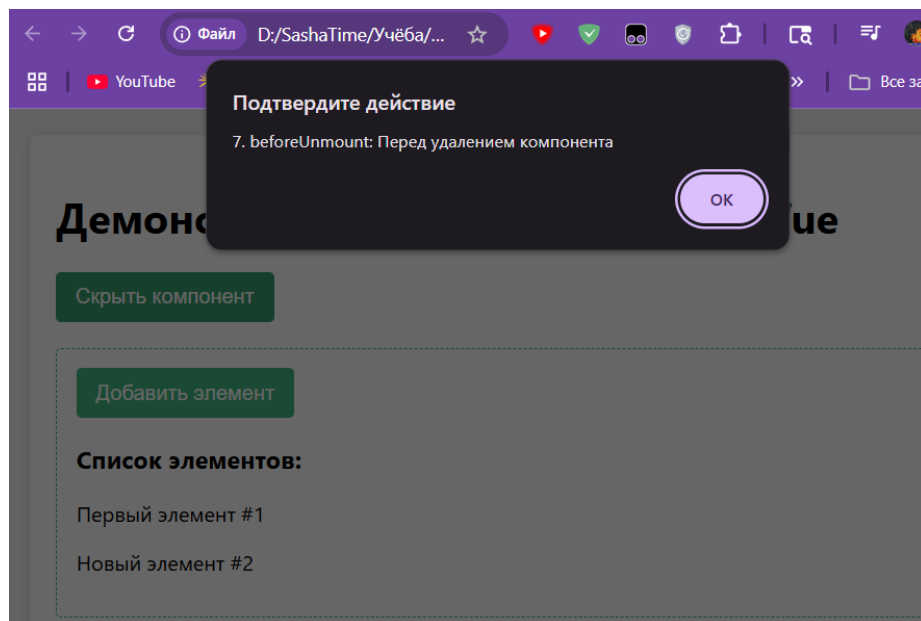


Рисунок 15 – Вызов витка `beforeUnmount` жизненного цикла Vue.js

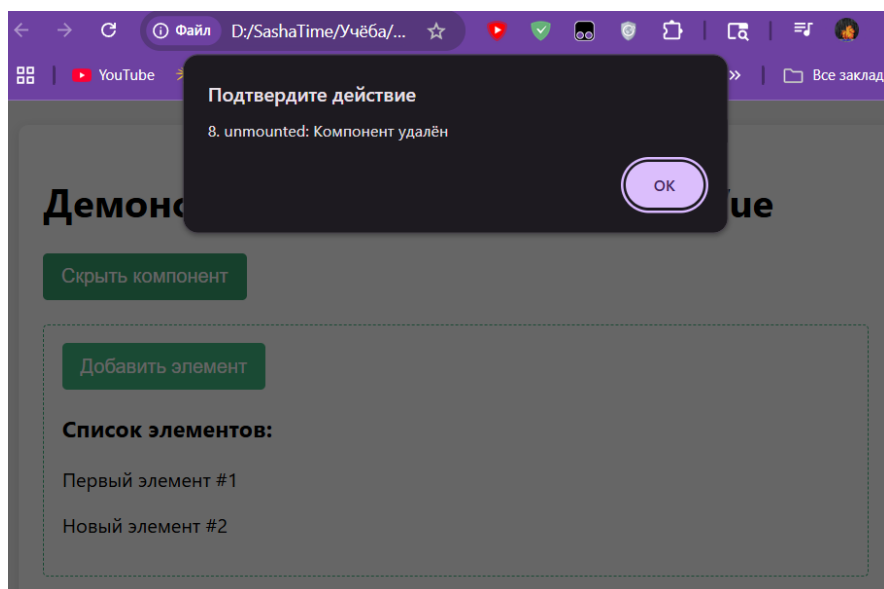


Рисунок 16 – Вызов витка Unmounted жизненного цикла Vue.js

После отработки хуков удаления компонента на странице перестанет отображаться содержимое компонента (см. рисунок 17).

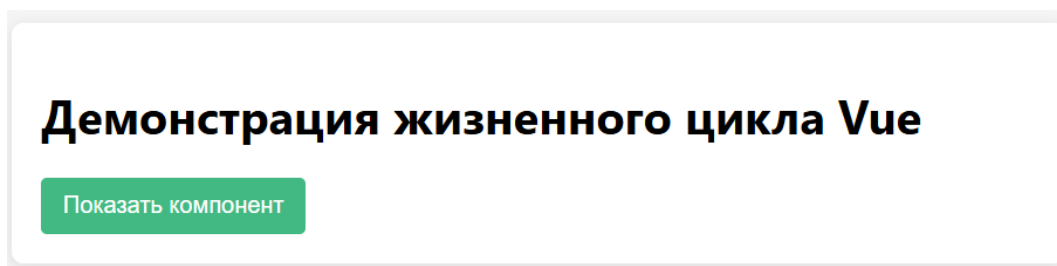


Рисунок 17 – Отображение веб-приложения демонстрации хуков жизненного цикла Vue.js с удалённым компонентом Vue.js

Списки, переборы, v-for

Листинг веб-приложения 2:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Списки, переборы, v-for</title>
  <script src="Vue.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
```

```

    max-width: 1000px;
    margin: 0 auto;
    padding: 20px;
}
.input-container {
    margin-bottom: 20px;
    padding: 15px;
    background: #f5f5f5;
    border-radius: 5px;
}
input {
    padding: 8px;
    width: 60px;
    margin-right: 10px;
}
.spiral-container {
    margin: 30px 0;
    text-align: center;
    overflow-x: auto;
}
.spiral-table {
    border-collapse: collapse;
    margin: 20px auto;
    display: inline-block;
}
.spiral-table td {
    border: 1px solid #ddd;
    padding: 10px;
    text-align: center;
    min-width: 40px;
    height: 40px;
    font-size: 0.9em;
}
.page-controls {
    margin: 15px 0;
    text-align: center;
}
button {
    padding: 8px 15px;
    margin: 0 5px;
    background: #42b983;
    color: white;
    border: none;
    border-radius: 4px;
}

```

```

        cursor: pointer;
    }
    .control-output {
        margin-top: 30px;
        padding: 15px;
        border: 1px solid #eee;
        background-color: #f9f9f9;
        border-radius: 5px;
    }
    .layer-row {
        margin-bottom: 10px;
        padding: 8px;
        background: #fff;
        border-radius: 4px;
    }
}
</style>
</head>
<body>
<div id="app">
    <div class="input-container">
        <label for="dimension">Размерность массива (N×N×N): </label>
        <input
            id="dimension"
            type="number"
            min="1"
            max="5"
            v-model.number="dimension"
            @blur="generateArray"
        >
    </div>

    <div v-if="cubicArray.length > 0" class="spiral-container">
        <h2>Спиральное представление массива</h2>
        <table class="spiral-table">
            <tr v-for="(row, rowIndex) in spiralView" :key="rowIndex">
                <td v-for="(cell, cellIndex) in row" :key="cellIndex">
                    {{ cell || " " }}
                </td>
            </tr>
        </table>
    </div>

    <div v-if="cubicArray.length > 0" class="control-output">
        <h2>Построчное представление по слоям</h2>

```

```

<div class="page-controls">
  <button @click="prevPage" :disabled="currentPage === 0">← Предыдущий
слой</button>
  <span>Слой {{ currentPage + 1 }} из {{ dimension }}</span>
  <button @click="nextPage" :disabled="currentPage === dimension - 1">Следующий
слой →</button>
</div>

```

```

<div v-for="(row, rowIndex) in currentLayer" :key="rowIndex" class="layer-row">
  Строка {{ rowIndex + 1 }}: {{ row.join(' $ ') }}
</div>
</div>
</div>

```

```

<script>
const { createApp, ref, computed } = Vue;

createApp({
  setup() {
    const dimension = ref(2);
    const cubicArray = ref([]);
    const spiralView = ref([]);
    const currentPage = ref(0);

    function generateRandomValue() {
      return Math.floor(Math.random() * 900) + 100; // 100-999
    }

    function calculateTableSize(n) {
      let size = Math.ceil(Math.sqrt(n*n*n));
      return size % 2 === 0 ? size + 1 : size;
    }

    function generateArray() {
      // Генерация 3D массива
      const newArray = [];
      for (let z = 0; z < dimension.value; z++) {
        const layer = [];
        for (let y = 0; y < dimension.value; y++) {
          const row = [];
          for (let x = 0; x < dimension.value; x++) {
            row.push(generateRandomValue());
          }
        }
      }
    }
  }
})

```

```

        layer.push(row);
    }
    newArray.push(layer);
}
cubicArray.value = newArray;
updateSpiralView();
}

function updateSpiralView() {
    const tableSize = calculateTableSize(dimension.value);
    const table = Array(tableSize).fill().map(() => Array(tableSize).fill(null));

    const allElements = [];
    for (let z = 0; z < dimension.value; z++) {
        for (let y = 0; y < dimension.value; y++) {
            for (let x = 0; x < dimension.value; x++) {
                allElements.push(cubicArray.value[z][y][x]);
            }
        }
    }

    let x = Math.floor(tableSize / 2);
    let y = Math.floor(tableSize / 2);
    let dx = 1, dy = 0;
    let segmentLength = 1;
    let segmentPassed = 0;
    let elementsPlaced = 0;

    while (elementsPlaced < allElements.length) {
        if (x >= 0 && x < tableSize && y >= 0 && y < tableSize) {
            table[y][x] = allElements[elementsPlaced];
            elementsPlaced++;
        }

        x += dx;
        y += dy;
        segmentPassed++;

        if (segmentPassed === segmentLength) {
            segmentPassed = 0;
            const temp = dx;
            dx = -dy;
            dy = temp;
        }
    }
}

```

```

        if (dy === 0) {
            segmentLength++;
        }
    }
}

spiralView.value = table;
}

function nextPage() {
    if (currentPage.value < dimension.value - 1) {
        currentPage.value++;
    }
}

function prevPage() {
    if (currentPage.value > 0) {
        currentPage.value--;
    }
}

const currentLayer = computed(() => {
    return cubicArray.value[currentPage.value] || [];
});

// Инициализация
generateArray();

return {
    dimension,
    cubicArray,
    spiralView,
    currentPage,
    currentLayer,
    generateArray,
    nextPage,
    prevPage
};
}
}).mount('#app');
</script>
</body>
</html>

```


Отображение веб-приложения 2-го задания в исходном состоянии в браузере представлено на рисунке 18.

Размерность массива (N×N×N):

Спиральное представление массива

| | | |
|-----|-----|-----|
| 200 | 745 | |
| 891 | 844 | 645 |
| 316 | 906 | 251 |

Построчное представление по слоям

← Предыдущий слой

Слой 1 из 2

Следующий слой →

Строка 1: 844 \$ 645

Строка 2: 251 \$ 906

Рисунок 18 – Отображение веб-приложения 2 задания в исходном состоянии в браузере

Отображение веб-приложения 2-го задания с размерностью 4x4x4 в браузере представлено на рисунке 19.

Размерность массива (N×N×N):

4

Спиральное представление массива

| | | | | | | | | |
|--|-----|-----|-----|-----|-----|-----|-----|-----|
| | | | | | | | | |
| | 155 | 337 | 829 | 377 | 422 | 819 | 680 | 865 |
| | 444 | 376 | 120 | 940 | 351 | 437 | 454 | 554 |
| | 426 | 569 | 885 | 770 | 244 | 593 | 132 | 779 |
| | 869 | 818 | 118 | 895 | 818 | 634 | 519 | 103 |
| | 752 | 598 | 868 | 150 | 642 | 936 | 452 | 526 |
| | 179 | 699 | 893 | 215 | 891 | 939 | 338 | 767 |
| | 949 | 994 | 147 | 671 | 637 | 537 | 159 | 644 |
| | 733 | 104 | 514 | 231 | 859 | 408 | 284 | 516 |

Построчное представление по слоям

← Предыдущий слой

Слой 1 из 4

Следующий слой →

Строка 1: 895 \$ 818 \$ 642 \$ 150

Строка 2: 868 \$ 118 \$ 885 \$ 770

Строка 3: 244 \$ 593 \$ 634 \$ 936

Строка 4: 939 \$ 891 \$ 215 \$ 893

Рисунок 19 – Отображение веб-приложения 2 задания с изменённой размерностью массива в браузере

Визуализация работы спирали на примере с размерностью массива 4x4x4, и демонстрацией построение слоев от 1 до 4 на рисунках 20, 21, 22, 23, 24 соответственно.

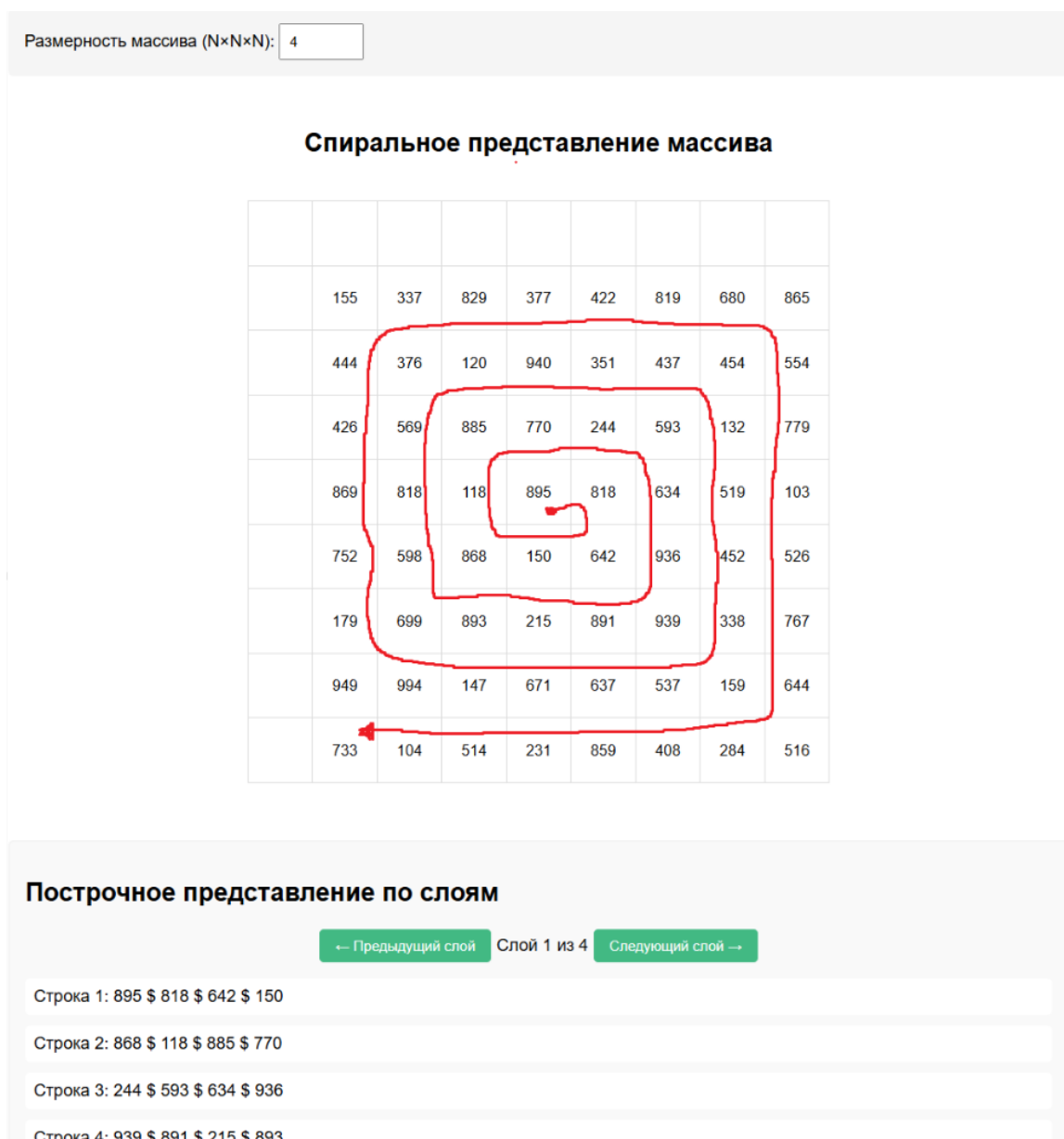


Рисунок 20 – Визуализация работы алгоритма спирали

Построчное представление по слоям

← Предыдущий слой Слой 1 из 4 Следующий слой →

Строка 1: 895 \$ 818 \$ 642 \$ 150

Строка 2: 868 \$ 118 \$ 885 \$ 770

Строка 3: 244 \$ 593 \$ 634 \$ 936

Строка 4: 939 \$ 891 \$ 215 \$ 893

Рисунок 21 – Визуализация работы алгоритма спирали

Построчное представление по слоям

← Предыдущий слой Слой 2 из 4 Следующий слой →

Строка 1: 699 \$ 598 \$ 818 \$ 569

Строка 2: 376 \$ 120 \$ 940 \$ 351

Строка 3: 437 \$ 454 \$ 132 \$ 519

Строка 4: 452 \$ 338 \$ 159 \$ 537

Рисунок 22 – Визуализация работы алгоритма спирали

Построчное представление по слоям

← Предыдущий слой Слой 3 из 4 Следующий слой →

Строка 1: 637 \$ 671 \$ 147 \$ 994

Строка 2: 949 \$ 179 \$ 752 \$ 869

Строка 3: 426 \$ 444 \$ 155 \$ 337

Строка 4: 829 \$ 377 \$ 422 \$ 819

Рисунок 23 – Визуализация работы алгоритма спирали

Построчное представление по слоям

← Предыдущий слой

Слой 4 из 4

Следующий слой →

| |
|------------------------------------|
| Строка 1: 680 \$ 865 \$ 554 \$ 779 |
| Строка 2: 103 \$ 526 \$ 767 \$ 644 |
| Строка 3: 516 \$ 284 \$ 408 \$ 859 |
| Строка 4: 231 \$ 514 \$ 104 \$ 733 |

Рисунок 24 – Визуализация работы алгоритма спирали

По рисункам 20-24, видно, что 3х мерный массив выводится по спирали в верном порядке.

Наблюдатели watch

Листинг веб-приложения 3 задания:

```
<!DOCTYPE html>
<html lang="ru">
<head>
  <meta charset="UTF-8">
  <title>Наблюдатели watch</title>
  <script src="Vue.js"></script>
  <style>
    body {
      font-family: Arial, sans-serif;
      max-width: 800px;
      margin: 0 auto;
      padding: 20px;
    }
    .container {
      display: flex;
      flex-direction: column;
      align-items: center;
    }
    input {
      padding: 10px;
      width: 300px;
      margin-bottom: 20px;
      font-size: 16px;
    }
    .image-container {
      width: 400px;
      height: 300px;
    }
```

```

        border: 2px solid #ddd;
        display: flex;
        justify-content: center;
        align-items: center;
        overflow: hidden;
    }
    .image-container img {
        max-width: 100%;
        max-height: 100%;
        object-fit: contain;
    }
    .available-images {
        margin-top: 30px;
        text-align: center;
    }
    .available-images h3 {
        margin-bottom: 10px;
    }
    .image-list {
        display: flex;
        flex-wrap: wrap;
        justify-content: center;
        gap: 10px;
    }
    .image-list img {
        width: 100px;
        height: 75px;
        object-fit: cover;
        border: 1px solid #ccc;
        cursor: pointer;
    }
    .image-list img:hover {
        border-color: #42b983;
    }
</style>
</head>
<body>
    <div id="app">
        <div class="container">
            <h1>Поиск изображений</h1>

            <input
                type="text"
                v-model="imageName"

```

```

        placeholder="Введите название изображения"
    >

    <div class="image-container">
        
    </div>

</div>

<script>
const { createApp, ref, watch } = Vue;

createApp({
  setup() {
    const imageName = ref("");
    const currentImage = ref("");
    const currentImageAlt = ref("");

    // Доступные изображения (замените на реальные пути)
    const Images = ref([
      'Image1.jpg',
      'Image2.jpg',
      'Image3.jpg',
      'Image4.jpg',
      'Image5.jpg',
      'Image6.jpg'
    ]);

    // Изображение "не найдено"
    const notFoundImage = 'Image7.jpg';

    // Соответствие имен изображениям
    const imageMapping = {
      'image1': 0,
      'image2': 1,
      'image3': 2,
      'image4': 3,
      'image5': 4,
      'image6': 5
    };

    // Наблюдаем за изменением имени изображения
    watch(imageName, (newValue) => {

```

```

const normalizedName = newValue.trim().toLowerCase();

if (imageMapping.hasOwnProperty(normalizedName)) {
  const index = imageMapping[normalizedName];
  currentImage.value = Images.value[index];
  currentImageAlt.value = `Изображение ${index + 1}`;
} else {
  currentImage.value = notFoundImage;
  currentImageAlt.value = 'Изображение отсутствует';
}
});

// Инициализация
currentImage.value = notFoundImage;
currentImageAlt.value = 'Изображение отсутствует';

return {
  imageName,
  currentImage,
  currentImageAlt,
  Images,
};
}
}).mount('#app');
</script>
</body>
</html>

```


Отображение веб-приложения 3-го задания в исходном состоянии в браузере представлено на рисунке 25. По умолчанию стоит image7 где изображена надпись «Изображение отсутствует».

Поиск изображений

Введите название изображения

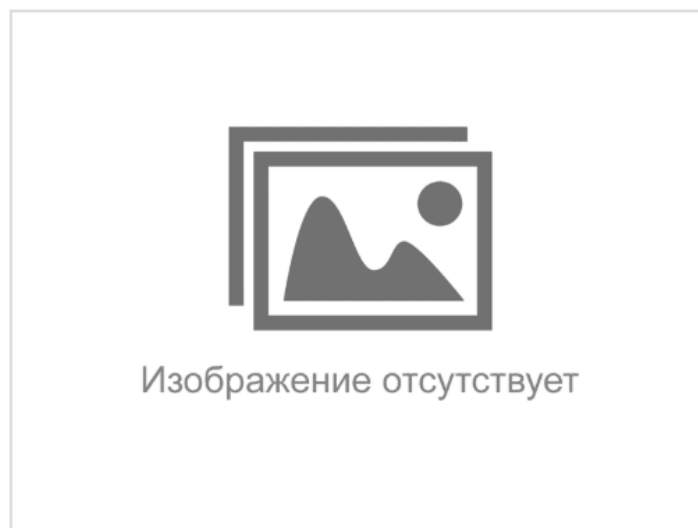


Рисунок 25 –Исходное состоянии в браузере
На рисунках 26-31 демонстрация image 1-6 соответственно.

Поиск изображений

image1|



Рисунок 26 – Демонстрация image1

Поиск изображений

image2



Рисунок 27 – Демонстрация image2

Поиск изображений

image3|



Рисунок 28 – Демонстрация image3

Поиск изображений

image4



Рисунок 29 – Демонстрация image4

Поиск изображений

image5



Рисунок 30 – Демонстрация image5

Поиск изображений

image6|



Рисунок 31 – Демонстрация image6

Вывод

Улучшил навыки работы с фреймворком Vue.js. Получил новый опыт работы с витками жизненного цикла Vue.js, директивой v-for, наблюдателями watch.