

**МИНИСТЕРСТВО ТРАНСПОРТА РОССИЙСКОЙ ФЕДЕРАЦИИ**

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»**

**(РУТ (МИИТ)**

|  |
| --- |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

Кафедра «Управление и защита информации»

**Отчет по лабораторной работе**

« **циклы, наблюдатели, витки**»

**по дисциплине**

«Web программирование»

Вариант 10

**Выполнил:** студенты

группы ТКИ-541

Плуталов Е.А., Мацко А.А.

**Проверил:** доцент кафедры УиЗИ, к.т.н. Сафронов А.И.

**Москва 2024 г.**

Оглавление

[Цель работы 3](#_Toc183505570)

[Формулировки задач 3](#_Toc183505571)

[Технология локального подключения фреймворка Vue.js 4](#_Toc183505572)

[Русифицированные схемы жизненного цикла Vue.js. 5](#_Toc183505573)

[Таблица соответствия переменных и методов, используемых в web-приложениях 6](#_Toc183505574)

[Содержательная часть по виткам жизненного цикла 11](#_Toc183505575)

[Содержательная часть по циклической отрисовке 18](#_Toc183505576)

[Содержательная часть по наблюдателям 22](#_Toc183505577)

[Вывод 25](#_Toc183505578)

# Цель работы

Цель работы заключается в разработке локальных одностраничных web-приложений (LSPWA) под управлением фреймворка Vue.js на языке JavaScript в соответствии с указаниями вариантов индивидуального задания.

# Формулировки задач

– локально (LSPWA), не прибегая к инструментарию Node.js и npm (Node Package Manager);

* **Задание 1:**

Реализовать витки жизненного цикла в файле с именем hooks\_Matsko\_A\_A\_Plutalov\_E\_A.html  
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. Русифицировать схему жизненного цикла Vue.js 3.x (допускается использование готовой реализации из официальной документации).

* **Задание 2:**

Реализовать v-for в файле с именем index-v-for\_ Matsko\_A\_A\_Plutalov\_E\_A.html;

Некоторое изображение нарезано стык в стык на 6 частей таким образом, чтобы оно фрагментарно размещалось в обрамлённой таблице <table> размерности [2 x 3] при использовании директивы v-for. При указании оператором в <input> меньшей и сопоставимой размерности таблицы не производить изменений при выводе фрагментированного изображения. При указании большей размерности – стараться настолько, насколько это возможно, центрировать фрагментированное изображение в таблице.

* **Задание 3:**

Реализовать watches в файле с именем index-watches\_ Matsko\_A\_A\_Plutalov\_E\_A.html;

Организовать <input> со свойствами калькулятора. Вводить в <input> можно что угодно, но реакция на изменение содержимого будет выполняться лишь в случаях соответствия текста одному из шаблонов: «a+b», «a–b», «a\*b» или «a/b», – выводить в этот же <input> результат выполнения арифметической операции. В «a» и «b» могут быть любые вещественные значения. Десятичный разделитель – «,». Предусмотреть настройки, когда количество пробелов в начале и в конце шаблона не имеет абсолютно никакого значения.

# Технология локального подключения фреймворка Vue.js

Vue.js подключается локально через ссылку на загруженный файл `vue.global.js`. Этот файл добавляется в `<script>` в HTML-документе**:**

<script src="vue.global.js"></script>

# Русифицированные схемы жизненного цикла Vue.js.

|  |
| --- |
|  |
| Рисунок 3 – Русифицированная схема жизненного цикла Vue.js. |

# Таблица соответствия переменных и методов, используемых в web-приложениях

**Задание 1:**

|  |  |  |
| --- | --- | --- |
| Переменная/Хук | Тип | Описание |
| Count | Переменная | Используется для хранения значения счетчика, которое отображается на экране. |
| logs | Переменная | Хранит массив строк для отображения логов вызова хуков жизненного цикла. |
| hooks | Переменная | Объект, содержащий статус вызова каждого хука жизненного цикла (true - вызван, false - не вызван). |
| Increment() | Метод | Метод для увеличения значения счетчика. |
| Destroy() | Метод | Метод для размонтирования приложения через app.unmount(). |
| logLifestyle(hook) | Метод | Метод, используемый для регистрации вызова хуков жизненного цикла, обновляющий объект hooks и добавляющий запись в массив logs. |
| beforeCreate() | Хук | Вызывается перед созданием экземпляра Vue. |
| Created() | Хук | Вызывается после создания экземпляра Vue. |
| beforeMount() | Хук | Вызывается перед монтированием экземпляра Vue. |
| mounted() | Хук | Вызывается после монтирования экземпляра Vue. |
| beforeUpdate() | Хук | Вызывается перед обновлением экземпляра Vue. |
| updated() | Хук | Вызывается после обновления экземпляра Vue. |
| beforeUnmount() | Хук | Вызывается перед размонтированием экземпляра Vue. |
| unmounted() | Хук | Вызывается после размонтирования экземпляра Vue. |

**Задание 2:**

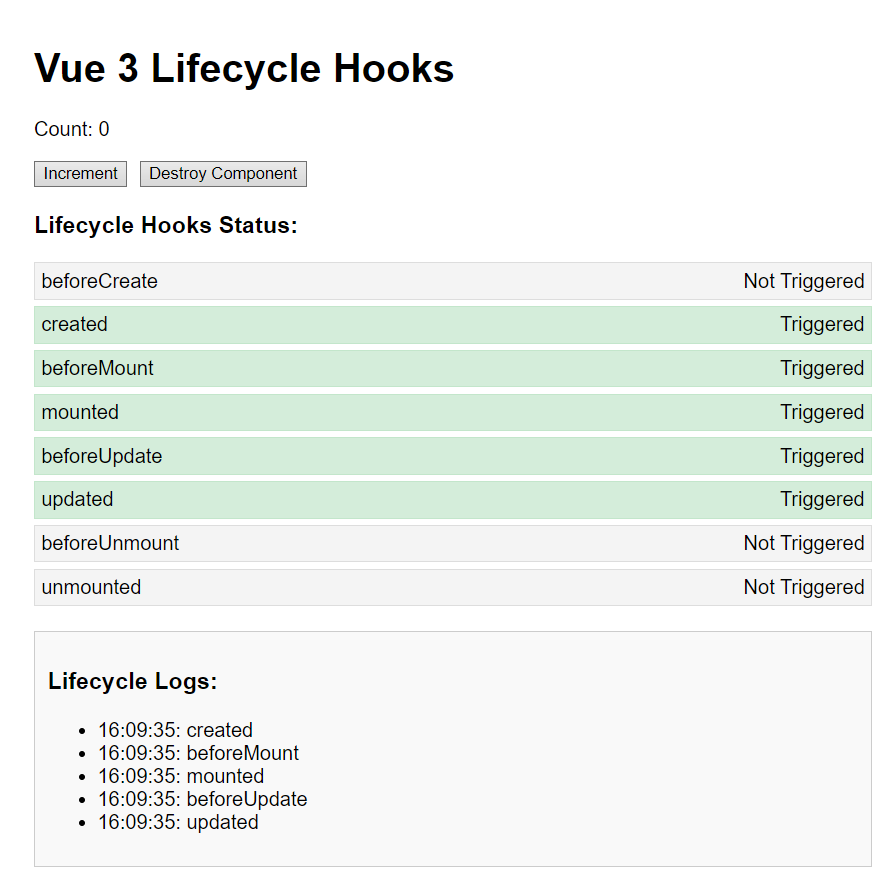
|  |  |  |
| --- | --- | --- |
| Переменная/Метод | Тип | Описание |
| rows | Переменная | Количество строк в таблице. Управляется через поле ввода и используется для построения массива строк (rowsArray). |
| columns | Переменная | Количество столбцов в таблице. Управляется через поле ввода и используется для построения массива столбцов (columnsArray). |
| imageParts | Переменная | Массив, содержащий пути к изображениям для фрагментов таблицы. |
| rowsArray | Вычисляемое | Массив, содержащий индексы строк (генерируется с помощью Array.from на основе значения rows). |
| columnsArray | Вычисляемое | Массив, содержащий индексы столбцов (генерируется с помощью Array.from на основе значения columns). |
| paddedImageParts | Вычисляемое | Массив, в котором изображения центрированы относительно ячеек таблицы. Если таблица больше 2x3, добавляются пустые ячейки, чтобы сбалансировать расположение. |
| rowIndex(row, col) | Метод | Вычисляет индекс текущей ячейки в таблице на основе индексов строки (row) и столбца (col). |
| getCellStyle(index) | Метод | Возвращает объект стилей для ячейки таблицы. Если изображение доступно для данного индекса, применяется фон из imageParts; если нет, ячейка остается пустой. |
| image-table | Таблица | HTML-таблица, создаваемая для отображения фрагментов изображения. Каждый td строится с помощью вычисляемых массивов строк и столбцов. |
| image-cell | Ячейка | Класс для td, применяемый для стилизации ячеек. Если ячейка содержит изображение, используется backgroundImage для отображения фрагмента изображения. |

**Задание 3:**

|  |  |  |
| --- | --- | --- |
| Переменная/Метод | Тип | Описание |
| inputValue | Переменная | Хранит значение из поля ввода (input). Содержит выражение, которое пользователь вводит для вычисления (например, "a+b"). |
| watch: inputValue | Наблюдатель | Следит за изменениями в inputValue и выполняет вычисления, если строка соответствует шаблону арифметического выражения. |
| expression | Локальная переменная | Содержит обрезанное (удалены пробелы) значение из inputValue. |
| operators | Локальная переменная | Список поддерживаемых арифметических операторов: +, -, \*, /. |
| parts | Локальная переменная | Содержит массив строк, полученных при разделении выражения по текущему оператору. |
| left | Локальная переменная | Левый операнд, полученный из выражения. Пробелы убираются, а запятые заменяются точками для корректного преобразования в число. |
| right | Локальная переменная | Правый операнд, полученный из выражения. Аналогично, обрабатывается для преобразования в число. |
| num1 | Локальная переменная | Преобразованный в число левый операнд (left). |
| num2 | Локальная переменная | Преобразованный в число правый операнд (right). |
| result | Локальная переменная | Результат вычисления выражения. Рассчитывается в зависимости от текущего оператора. |

# Содержательная часть по виткам жизненного цикла

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Vue 3 Lifecycle Hooks</title>  <script src="https://unpkg.com/vue@3"></script>  <style>  body {  font-family: Arial, sans-serif;  padding: 20px;  }  .log {  margin-top: 20px;  max-height: 200px;  overflow-y: auto;  border: 1px solid #ccc;  padding: 10px;  background-color: #f9f9f9;  }  button {  margin-right: 10px;  }  .hooks-status {  margin-top: 20px;  }  .hook-state {  margin-bottom: 5px;  padding: 5px;  border: 1px solid #ddd;  background-color: #f4f4f4;  display: flex;  justify-content: space-between;  }  .hook-state.active {  background-color: #d4edda;  border-color: #c3e6cb;  }  </style>  </head>  <body>  <div id="app">  <h1>Vue 3 Lifecycle Hooks</h1>  <p>Count: {{ count }}</p>  <button @click="increment">Increment</button>  <button @click="destroy">Destroy Component</button>  <div class="hooks-status">  <h3>Lifecycle Hooks Status:</h3>  <div v-for="(status, hook) in hooks" :key="hook" :class="['hook-state', { active: status }]">  <span>{{ hook }}</span>  <span>{{ status ? 'Triggered' : 'Not Triggered' }}</span>  </div>  </div>  <div class="log">  <h3>Lifecycle Logs:</h3>  <ul>  <li v-for="(log, index) in logs" :key="index">{{ log }}</li>  </ul>  </div>  </div>  <script>  const app = Vue.createApp({  data() {  return {  count: 0,  logs: [],  hooks: {  beforeCreate: false,  created: false,  beforeMount: false,  mounted: false,  beforeUpdate: false,  updated: false,  beforeUnmount: false,  unmounted: false  }  };  },  methods: {  increment() {  this.count++;  },  destroy() {  app.unmount(); // Размонтирование через экземпляр приложения  },  logLifecycle(hook) {  if (!this.hooks[hook]) {  this.hooks[hook] = true;  this.logs.push(`${new Date().toLocaleTimeString()}: ${hook}`);  }  }  },  beforeCreate() {  console.log('beforeCreate');  // Прямое логирование, так как this.logs недоступно  if (!this.\_logs) this.\_logs = [];  this.\_logs.push(`${new Date().toLocaleTimeString()}: beforeCreate`);  },  created() {  console.log('created');  this.logLifecycle('created');  },  beforeMount() {  console.log('beforeMount');  this.logLifecycle('beforeMount');  },  mounted() {  console.log('mounted');  this.logLifecycle('mounted');  },  beforeUpdate() {  console.log('beforeUpdate');  this.logLifecycle('beforeUpdate');  },  updated() {  console.log('updated');  this.logLifecycle('updated');  },  beforeUnmount() {  console.log('beforeUnmount');  this.logLifecycle('beforeUnmount');  },  unmounted() {  console.log('unmounted');  this.logLifecycle('unmounted');  }  });  const vm = app.mount('#app');  </script>  </body>  </html> |
| Код программы Задание 1 |



|  |
| --- |
|  |
| Рисунок 4 – *Главная страница* |
|  |
| Рисунок 5 – *Главная страница после нажатия «Increment»* |
|  |
| Рисунок 6 – Главная страница после нажатия «Destroy Component» |
|  |
| Рисунок 7 – Сеть Петри к заданию 1 |

# Содержательная часть по циклической отрисовке

|  |
| --- |
| <!DOCTYPE html>  <html lang="ru">  <head>  <meta charset="UTF-8" />  <meta name="viewport" content="width=device-width, initial-scale=1.0" />  <title>Фрагментированное изображение в таблице</title>  <style>  /\* Стили таблицы и ячеек \*/  .image-table {  width: 100%;  max-width: 500px;  margin: auto;  border-collapse: collapse;  border: 1px solid #ddd;  }  .image-cell {  width: 100px;  height: 100px;  border: 1px solid #ddd;  background-size: cover;  background-position: center;  }  </style>  </head>  <body>  <div id="app">  <!-- Ввод количества строк и столбцов -->  <div>  <label>Количество строк:</label>  <input type="number" v-model.number="rows" min="2" @input="rows = Math.max(rows, 2)" />  <label>Количество столбцов:</label>  <input type="number" v-model.number="columns" min="3" @input="columns = Math.max(columns, 3)" />  <!-- Таблица с фрагментами изображения -->  <table class="image-table">  <tr v-for="row in rowsArray" :key="row">  <td  v-for="col in columnsArray"  :key="col"  :style="getCellStyle(rowIndex(row, col))"  class="image-cell"  ></td>  </tr>  </table>  </div>  </div>  <!-- Подключение Vue.js -->  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>  <script>  const { createApp, ref, computed } = Vue;  createApp({  setup() {  const rows = ref(2);  const columns = ref(3);  // Список фрагментов изображения (пути к файлам частей)  const imageParts = [  '../Pict/image\_part\_001.jpg',  '../Pict/image\_part\_002.jpg',  '../Pict/image\_part\_003.jpg',  '../Pict/image\_part\_004.jpg',  '../Pict/image\_part\_005.jpg',  '../Pict/image\_part\_006.jpg',  ];  const rowsArray = computed(() => Array.from({ length: Math.max(rows.value, 2) }, (\_, i) => i));  const columnsArray = computed(() => Array.from({ length: Math.max(columns.value, 3) }, (\_, i) => i));  const paddedImageParts = computed(() => {  const totalCells = Math.max(rows.value, 2) \* Math.max(columns.value, 3);  if (rows.value <= 2 && columns.value <= 3) return imageParts;  const emptyCells = Array(totalCells).fill(null);  const startIndex = Math.floor((totalCells - imageParts.length) / 2);  for (let i = 0; i < imageParts.length; i++) {  emptyCells[startIndex + i] = imageParts[i];  }  return emptyCells;  });  const rowIndex = (row, col) => row \* Math.max(columns.value, 3) + col;  const getCellStyle = (index) => {  const part = paddedImageParts.value[index];  return part  ? {  backgroundImage: `url(${part})`,  backgroundSize: 'cover',  backgroundPosition: 'center',  }  : {};  };  return {  rows,  columns,  rowsArray,  columnsArray,  rowIndex,  getCellStyle,  };  },  }).mount('#app');  </script>  </body>  </html> |
| Код программы Задание 2 |

|  |
| --- |
|  |
| Рисунок 8 – Отображение в браузере задания 2 |
|  |
| Рисунок 9 – Отображение в браузере задания 2 в изменённом состоянии |
|  |
| Рисунок 10 – Сеть Петри к заданию 2 |

# Содержательная часть по наблюдателям

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Calculator Input</title>  <script src="https://unpkg.com/vue@3"></script>  </head>  <body>  <div id="app">  <input v-model="inputValue" placeholder="Enter expression like a+b" />  </div>  <script>  const { createApp, watch } = Vue;  createApp({  data() {  return {  inputValue: '', // Хранит значение из input  };  },  watch: {  inputValue(newValue) {  // Убираем пробелы в начале и конце строки  const expression = newValue.trim();  // Проверка шаблона и выполнение вычисления  const operators = ['+', '-', '\*', '/'];  for (const operator of operators) {  const parts = expression.split(operator);  // Если разделение прошло корректно и есть две части  if (parts.length === 2) {  const left = parts[0].replace(',', '.').trim();  const right = parts[1].replace(',', '.').trim();  // Пробуем привести строки к числовому значению  const num1 = parseFloat(left);  const num2 = parseFloat(right);  // Проверяем, что оба значения являются числовыми  if (!isNaN(num1) && !isNaN(num2)) {  let result;  // Выполняем соответствующую операцию  switch (operator) {  case '+':  result = num1 + num2;  break;  case '-':  result = num1 - num2;  break;  case '\*':  result = num1 \* num2;  break;  case '/':  result = num2 !== 0 ? num1 / num2 : 'Ошибка';  break;  }  // Записываем результат в input, заменяя точку на запятую  this.inputValue = String(result).replace('.', ',');  return; // Прерываем цикл после вычисления  }  }  }  }  }  }).mount('#app');  </script>  </body>  </html> |
| Код программы Задание 3 |

|  |
| --- |
|  |
| Рисунок 11 – Отображение в браузере задания 3 |
| Рисунок 12 – Результат 1231231 \* 6 |
|  |
| Рисунок 13 – Сеть Петри к заданию 3 |

# Вывод

В данной работе были приобретены навыки по созданию локальных веб-приложений на Vue.js, изучены основные понятия, принципы и методы работы с жизненными циклами, наблюдателями, директивами. Было выполнено создание локального приложения без использования Node.js и npm, с акцентом на локальное подключение фреймворка Vue.js. Поставленные задачи выполнены в полном объёме.