

**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 14**

**Выполнил:**

Студенты группы ТКИ-541

Моисеенко М.А.

Земсков Н.А.

**Проверил:**

доцент кафедры УиЗИ, к.т.н.

Сафронов А.И.

**Москва 2024 г.**

Оглавление

[1. Цель работы 3](#_Toc184861141)

[2. Формулировка задачи 3](#_Toc184861142)

[3. Спецификация оборудования, на котором выполнялась работа. 4](#_Toc184861143)

[4. Информация об используемом браузере 5](#_Toc184861144)

[5. Технология локального подключения фреймворка Vue.js 5](#_Toc184861145)

[6. Русифицированная схема жизненного цикла Vue.js 6](#_Toc184861146)

[7. Таблица соответствия переменных и методов 7](#_Toc184861147)

[7.1. Таблица соответствия задание 1 7](#_Toc184861153)

[7.2. Таблица соответствия задание 2 8](#_Toc184861154)

[7.3. Таблица соответствия задание 3 10](#_Toc184861155)

[8. Содержательная часть 11](#_Toc184861156)

[8.1. Задания 1 11](#_Toc184861158)

[8.2. Задание 2 20](#_Toc184861159)

[8.3. Задания 3 28](#_Toc184861160)

[9. Вывод 33](#_Toc184861161)

# Цель работы

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

# Формулировка задачи

* 1. 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) (см. раздел «Задача 1»).
  2. Изобразить в области масштабируемой векторной графики <*svg*> размерности 540 на 960 пикселей по ширине и высоте, соответственно, вертикальную координатную ось со стрелкой и засечками. Ось отстаёт от рамок <*svg*> на 50 пикселей по вертикали и горизонтали, соответственно. Стрелка оси всегда фиксированных размеров, устанавливаемых разработчиков (оператор не управляет настройками вида этого элемента). Оператор меняет шаг засечек при использовании элементов <*input*> в режиме *range* в разумных пределах (аргументировать выбор текстом в отчёте). Между засечками посередине всегда размещаются промежуточные засечки точно посередине. Оператор указывает начало и конец диапазона вывода числовых маркеров под засечками при использовании элементов <*input*>; при использовании <*input*> в режиме *checkbox* оператор меняет направление стрелки оси: либо положительное направление, либо снизу-вверх, либо сверху-вниз. В качестве элементов внутри <*svg*> использовать только <*line*> и <*text*>.
  3. Организовать <*input*> со свойствами калькулятора. Вводить в <*input*> можно что угодно, но как только там, среди текста, появляется конструкции вида: «*Sin(x)*», «*Cos(x)*», «*Tg(x)*», «*Ctg(x)*», «*pi*» – выводить в этот же <*input*> на соответствующих позициях расчётные значения тригонометрических функций и константы.

# Спецификация оборудования, на котором выполнялась работа.

|  |
| --- |
| Рисунок 1 – Сведения об операционной системе |

Таблица 1 – Спецификация параметров операционной системы

|  |  |  |
| --- | --- | --- |
| **№** | **Характеристика** | **Сведения** |
| **1** | Тип устройства | Ноутбук |
| **2** | Модель | Honor NMH-WD9NH |
| **3** | Операционная система | Microsoft Windows 10 Pro |
| **4** | Процессор | AMD Ryzen 5 5500U |
| **5** | Оперативная память | 8 Gb |
| **6** | Объем жесткого диска | 1 Tb |

# Информация об используемом браузере

|  |
| --- |
| Рисунок 2 – Информация о браузере |

# Технология локального подключения фреймворка Vue.js

Для простого приложения достаточно одного HTML-файла в директории проекта. Код размещают внутри файла и сохраняют. Для запуска нужно добавить тег script с внешней библиотекой Vue.js по ссылке: <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>. Затем файл помещают в директорию проекта и перетаскивают в открытое окно браузера.

# Русифицированная схема жизненного цикла Vue.js

|  |
| --- |
| Рисунок 3 – Схема жизненного цикла |

# Таблица соответствия переменных и методов



## Таблица соответствия задание 1

|  |  |  |
| --- | --- | --- |
| Описание | Виток/Хук/Метод | Тип |
| Вызывается перед созданием экземпляра Vue | beforeCreate() | Хук |
| Вызывается после создания экземпляра Vue | created() | Хук |
| Вызывается перед монтированием экземпляра Vue | beforeMount() | Хук |
| Вызывается после монтирования экземпляра Vue | mounted() | Хук |
| Вызывается перед обновлением экземпляра Vue | beforeUpdate() | Хук |
| Вызываеся после обновления экземпляра Vue | updated() | Хук |
| Вызывается перед размонтированием экземпляра Vue | beforeUnmount() | Хук |
| Вызывается после размонтирования экземпляра Vue | unmounted() | Хук |
| Переменная для хранения логов хуков | logs: | Переменная |
| Массив сообщений | messages: | Переменная |
| Флаг для редактирования | isEdeting: | Переменная |
| Индекс редактируемого сообщения | editingIndex: | Переменная |
| Текст нового сообщения | newMessageText: | Переменная |
| Добавление нового сообщения | addMessage() | Метод |
| Удаление сообщения | removeMessage() | Метод |
| Изменение сообщения | changeMessage() | Метод |
| Подтверждение изменения сообщения | confirmChange(index) | Метод |
| Отмена изменения сообщения | canceChangeMessage() | Метод |
| Размонтирование приложения | umountApp() | Метод |

## Таблица соответствия задание 2

|  |  |  |
| --- | --- | --- |
| Описание | Виток/Хук/Метод | Тип |
| Вызывается перед созданием экземпляра Vue | beforeCreate() | Хук |
| Вызывается после создания экземпляра Vue | created() | Хук |
| Вызывается перед монтированием экземпляра Vue | beforeMount() | Хук |
| Вызывается после монтирования экземпляра Vue | mounted() | Хук |
| Вызывается перед обновлением экземпляра Vue | beforeUpdate() | Хук |
| Вызываеся после обновления экземпляра Vue | updated() | Хук |
| Вызывается перед размонтированием экземпляра Vue | beforeUnmount() | Хук |
| Вызывается после размонтирования экземпляра Vue | unmounted() | Хук |
| Переменная для хранения шага засечек | step: | Переменная |
| Переменные для начала и конца диапазона | rangeStart:  rangeEnd: | Переменная |
| Переменная для направления стрелки | invertDirection: | Переменная |
| Переменные для размеров SVG | svgWidth:  svgHeight: | Переменная |
| Вычисляет центр оси X | centerX() | Метод |
| Генерирует координаты крупных засечек | majorTicks() | Метод |
| Генерирует координаты мелких засечек | minorTicks() | Метод |
| Генерирует метки оси | labels() | Метод |
| Координата стрелки (верх) | arrowHeadY1() | Метод |
| Координата стрелки (низ) | arrowHeadY2() | Метод |

## Таблица соответствия задание 3

|  |  |  |
| --- | --- | --- |
| Описание | Виток/Хук/Метод | Тип |
| Вызывается перед созданием экземпляра Vue | beforeCreate() | Хук |
| Вызывается после создания экземпляра Vue | created() | Хук |
| Вызывается перед монтированием экземпляра Vue | beforeMount() | Хук |
| Вызывается после монтирования экземпляра Vue | mounted() | Хук |
| Вызывается перед обновлением экземпляра Vue | beforeUpdate() | Хук |
| Вызываеся после обновления экземпляра Vue | updated() | Хук |
| Вызывается перед размонтированием экземпляра Vue | beforeUnmount() | Хук |
| Вызывается после размонтирования экземпляра Vue | unmounted() | Хук |
| Текстовое поле ввода | inputText: | Переменная |
| Наблюдатель за изменением текста | watch.inputText | Переменная |
| Обработка текста и расчет значений | calculate(text) | Переменная |

# Содержательная часть



## Задания 1

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<title>Демонстрация жизненного цикла Vue.js 3.x с дополнительными методами</title>

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

<style>

body {

font-family: Arial, sans-serif;

margin: 20px;

}

.hook-log {

margin-bottom: 20px;

padding: 10px;

border: 1px solid #ccc;

border-radius: 5px;

}

.messages {

margin-top: 20px;

}

.message {

padding: 10px;

border-bottom: 1px solid #eee;

display: flex;

justify-content: space-between;

align-items: center;

}

.message:last-child {

border-bottom: none;

}

button {

margin-right: 5px;

padding: 5px 10px;

}

.controls {

margin-top: 20px;

}

.edit-input {

margin-top: 10px;

}

</style>

</head>

<body>

<div id="app">

<h1>Демонстрация жизненного цикла Vue.js 3.x с дополнительными методами</h1>

<div class="hook-log">

<h2>События жизненного цикла:</h2>

<ul>

<li>beforeCreate: {{ logs.beforeCreate }}</li>

<li>created: {{ logs.created }}</li>

<li>beforeMount: {{ logs.beforeMount }}</li>

<li>mounted: {{ logs.mounted }}</li>

<li>beforeUpdate: {{ logs.beforeUpdate }}</li>

<li>updated: {{ logs.updated }}</li>

</ul>

</div>

<div class="messages">

<h2>Сообщения:</h2>

<div class="message">

<span>{{ messages[0] }}</span>

<button @click="changeMessage(0)">Изменить</button>

</div>

<div class="message">

<span>{{ messages[1] }}</span>

<button @click="changeMessage(1)">Изменить</button>

</div>

<div class="message">

<span>{{ messages[2] }}</span>

<button @click="changeMessage(2)">Изменить</button>

</div>

</div>

<div class="controls">

<button @click="addMessage">Добавить сообщение</button>

<button @click="removeMessage">Удалить сообщение</button>

<button @click="unmountApp">Размонтировать приложение</button>

</div>

<div v-if="isEditing" class="edit-input">

<h3>Изменить сообщение "{{ editingMessage }}"</h3>

<input v-model="newMessageText" placeholder="Новое сообщение" />

<button @click="confirmChangeMessage">Подтвердить</button>

<button @click="cancelChangeMessage">Отмена</button>

</div>

</div>

<script>

const { createApp } = Vue;

const app = createApp({

data() {

return {

logs: {

beforeCreate: 'Не выполнено',

created: 'Не выполнено',

beforeMount: 'Не выполнено',

mounted: 'Не выполнено',

beforeUpdate: 'Не выполнено',

updated: 'Не выполнено'

},

messages: ['Сообщение 1', 'Сообщение 2', 'Сообщение 3'],

isEditing: false,

editingIndex: null,

newMessageText: ''

}

},

methods: {

addMessage() {

const newMessage = `Сообщение ${this.messages.length + 1}`;

if (this.messages.length < 3) {

this.messages.push(newMessage);

console.log(`Добавлено: ${newMessage}`);

} else {

alert('Максимум три сообщения.');

}

},

removeMessage() {

if (this.messages.length > 0) {

const removed = this.messages.pop();

console.log(`Удалено: ${removed}`);

} else {

console.log('Нет сообщений для удаления.');

}

},

changeMessage(index) {

this.editingIndex = index;

this.newMessageText = this.messages[index];

this.isEditing = true;

console.log(`Редактирование сообщения ${index + 1}`);

},

confirmChangeMessage() {

if (this.newMessageText.trim() !== '') {

const oldMessage = this.messages[this.editingIndex];

this.messages[this.editingIndex] = this.newMessageText.trim();

console.log(`Сообщение ${this.editingIndex + 1} изменено с "${oldMessage}" на "${this.newMessageText.trim()}"`);

this.isEditing = false;

this.editingIndex = null;

this.newMessageText = '';

} else {

alert('Текст сообщения не может быть пустым.');

}

},

cancelChangeMessage() {

this.isEditing = false;

this.editingIndex = null;

this.newMessageText = '';

console.log('Редактирование сообщения отменено.');

},

unmountApp() {

app.unmount();

console.log('Приложение размонтировано.');

}

}

});

app.mount('#app');

</script>

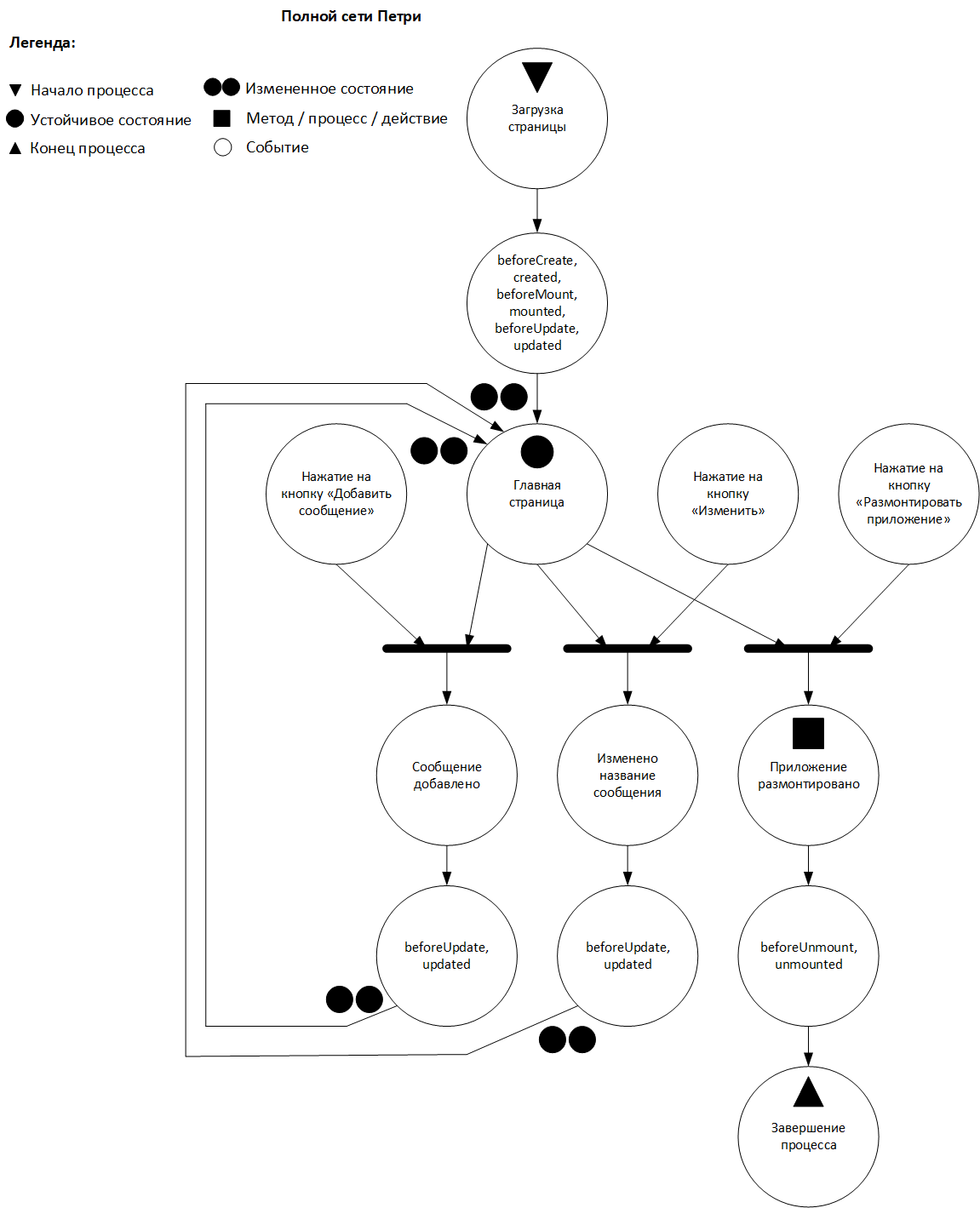
</body>

</html>

На рисунке 4-7 представлено отображение в браузере нашего web-приложения. Согласно заданию, необходимо показать работу всех витков/хуков/методов жизненного цикла.

|  |
| --- |
| Рисунок 4 – Отображение в браузере |
| Рисунок 5 – Удалили сообщение |
| Рисунок 6 – Сообщение 2 изменено, Сообщение 3 добавлено |
| Рисунок 7 – Отображение в консоли (кнопка «Размонтировать приложение» нажата) |

Ниже представлена сеть Петри процесса с момента запуска файла страницы до демонстрации работы витков/хуков/методов жизненного цикла.



## Задание 2

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>SVG Координатная Ось</title>

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

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

margin-top: 20px;

}

svg {

border: 1px solid black;

display: block;

margin: 20px auto;

}

.controls {

margin-bottom: 20px;

}

</style>

</head>

<body>

<div id="app">

<div class="controls">

<label>

Шаг засечек:

<input type="range" v-model="step" min="20" max="100">

{{ step }} px

</label>

<br>

<label>

Начало диапазона:

<input type="number" v-model.number="rangeStart">

</label>

<label>

Конец диапазона:

<input type="number" v-model.number="rangeEnd">

</label>

<br>

<label>

Направление стрелки:

<input type="checkbox" v-model="invertDirection"> Снизу вверх

</label>

</div>

<svg :width="svgWidth" :height="svgHeight">

<!-- Ось -->

<line :x1="centerX" :y1="margin" :x2="centerX" :y2="svgHeight - margin" stroke="black" stroke-width="2"></line>

<!-- Стрелка -->

<line

:x1="centerX"

:y1="arrowHeadY1"

:x2="centerX - 10"

:y2="arrowHeadY2"

stroke="black" stroke-width="2"></line>

<line

:x1="centerX"

:y1="arrowHeadY1"

:x2="centerX + 10"

:y2="arrowHeadY2"

stroke="black" stroke-width="2"></line>

<!-- Засечки -->

<line v-for="(y, index) in majorTicks" :key="index" :x1="centerX - 10" :y1="y" :x2="centerX + 10" :y2="y" stroke="black" stroke-width="1"></line>

<!-- Промежуточные засечки -->

<line v-for="(y, index) in minorTicks" :key="'minor-' + index" :x1="centerX - 5" :y1="y" :x2="centerX + 5" :y2="y" stroke="black" stroke-width="1"></line>

<!-- Числовые метки -->

<text v-for="(label, index) in labels" :key="'label-' + index" :x="centerX + 15" :y="label.y" font-size="12" fill="black">

{{ label.value }}

</text>

</svg>

</div>

<script>

const app = Vue.createApp({

data() {

return {

svgWidth: 540,

svgHeight: 960,

margin: 50,

step: 50, // Шаг между засечками

rangeStart: -10,

rangeEnd: 10,

invertDirection: false, // Направление стрелки

};

},

computed: {

centerX() {

return this.svgWidth / 2;

},

majorTicks() {

const ticks = [];

const startY = this.margin;

const endY = this.svgHeight - this.margin;

const totalHeight = endY - startY;

// Общее количество шагов

const numSteps = Math.floor(totalHeight / this.step);

for (let i = 0; i <= numSteps; i++) {

const y = this.invertDirection

? endY - i \* this.step

: startY + i \* this.step;

ticks.push(y);

}

return ticks;

},

minorTicks() {

const ticks = [];

for (let i = 0; i < this.majorTicks.length - 1; i++) {

const midY = (this.majorTicks[i] + this.majorTicks[i + 1]) / 2;

ticks.push(midY);

}

return ticks;

},

labels() {

const labels = [];

const totalLabels = this.majorTicks.length;

const rangeStep = (this.rangeEnd - this.rangeStart) / (totalLabels - 1 || 1);

for (let i = 0; i < totalLabels; i++) {

const currentValue = this.invertDirection

? this.rangeEnd - i \* rangeStep

: this.rangeStart + i \* rangeStep;

labels.push({

y: this.majorTicks[i] + 5,

value: currentValue.toFixed(1),

});

}

return labels;

},

arrowHeadY1() {

return this.invertDirection ? this.margin : this.svgHeight - this.margin;

},

arrowHeadY2() {

return this.invertDirection ? this.margin + 10 : this.svgHeight - this.margin - 10;

},

},

});

app.mount("#app");

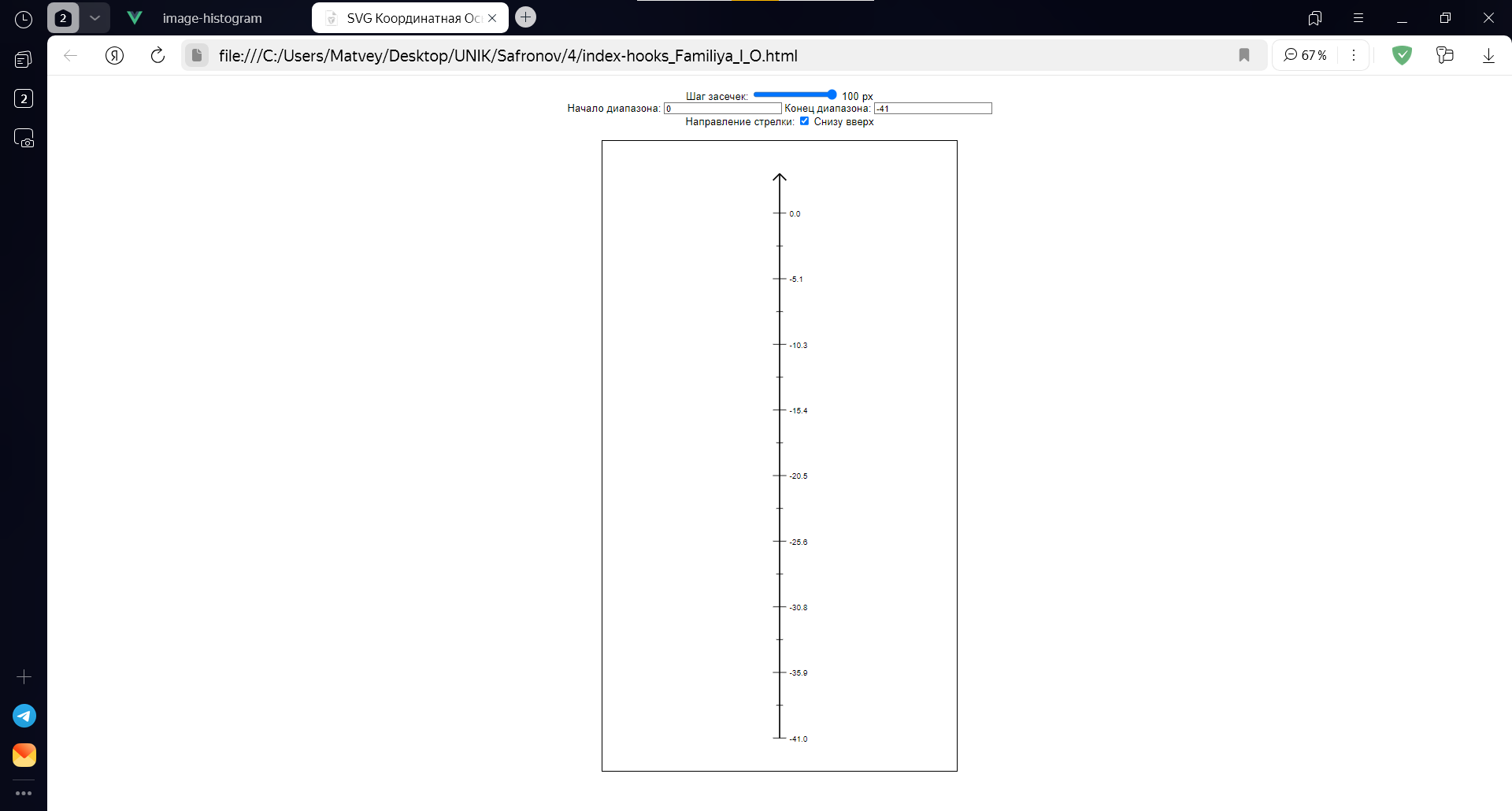
</script>

</body>

</html>

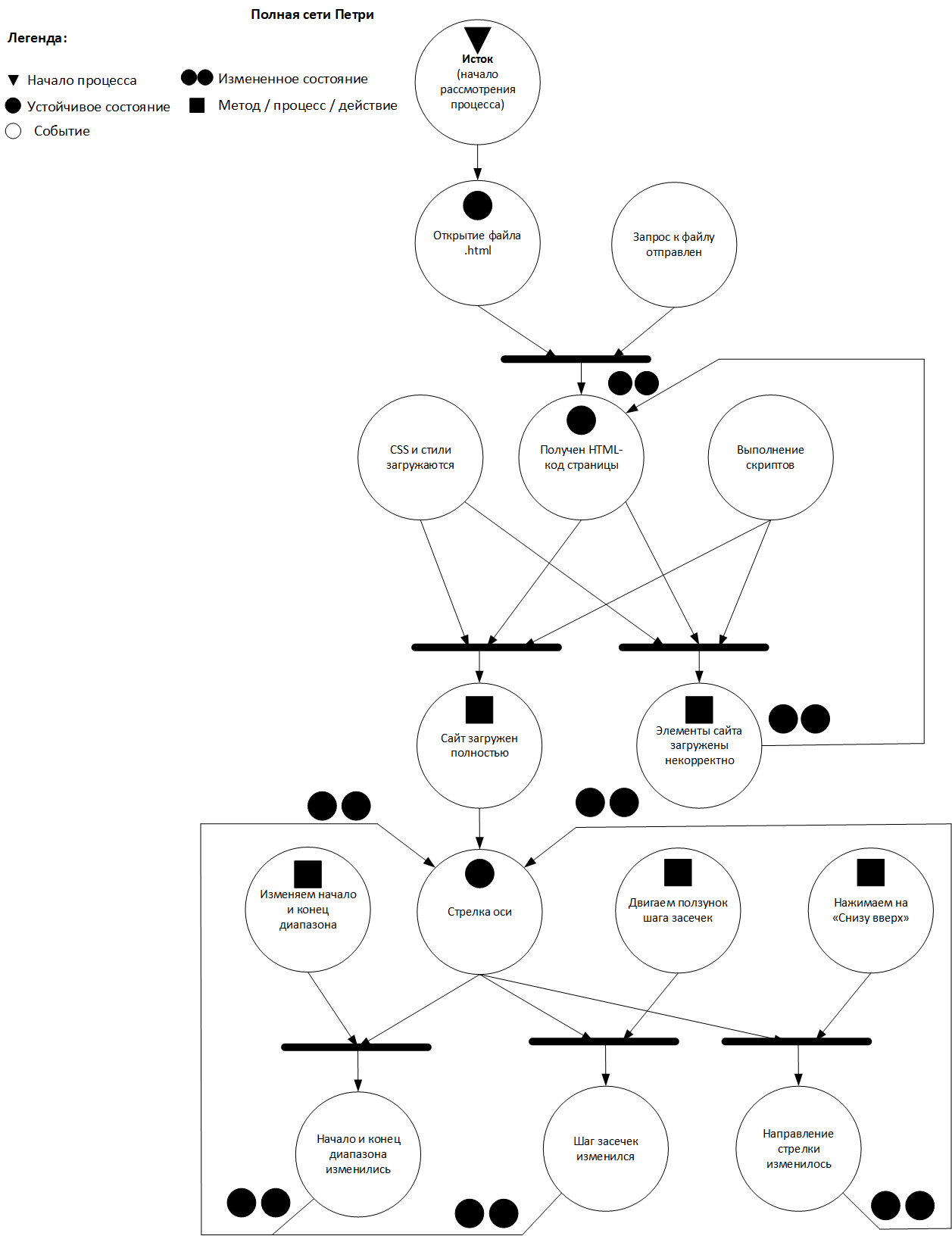
На рисунке 8-9 представлено отображение в браузере нашего web‑приложения.

|  |
| --- |
| Рисунок 8 – Отображение в браузере |



*Рисунок 9 – Изменение значений*

Ниже представлена сеть Петри процесса с момента запуска файла страницы до демонстрации работы витков/хуков/методов жизненного цикла.



## Задания 3

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Калькулятор с наблюдателями</title>

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

</head>

<body>

<div id="app">

<h1>Калькулятор</h1>

<p>Введите текст, содержащий конструкции вида: <code>Sin(x)</code>, <code>Cos(x)</code>, <code>Tg(x)</code>, <code>Ctg(x)</code>, <code>pi</code>.</p>

<input type="text" v-model="inputText" placeholder="Введите текст">

</div>

<script>

const app = Vue.createApp({

data() {

return {

inputText: "", // Текст в <input>

};

},

watch: {

inputText(value) {

// Наблюдатель за изменением текста

this.inputText = this.calculate(value);

},

},

methods: {

calculate(text) {

// Замены для константы и функций

try {

// Обработка π

text = text.replace(/pi/gi, Math.PI.toFixed(5));

// Обработка Sin(x)

text = text.replace(/Sin\((-?\d+\.?\d\*)\)/gi, (match, x) => {

return Math.sin(parseFloat(x)).toFixed(5);

});

// Обработка Cos(x)

text = text.replace(/Cos\((-?\d+\.?\d\*)\)/gi, (match, x) => {

return Math.cos(parseFloat(x)).toFixed(5);

});

// Обработка Tg(x)

text = text.replace(/Tg\((-?\d+\.?\d\*)\)/gi, (match, x) => {

return Math.tan(parseFloat(x)).toFixed(5);

});

// Обработка Ctg(x)

text = text.replace(/Ctg\((-?\d+\.?\d\*)\)/gi, (match, x) => {

const value = 1 / Math.tan(parseFloat(x));

return value.toFixed(5);

});

} catch (error) {

console.error("Ошибка при обработке выражения:", error);

}

return text;

},

},

});

app.mount("#app");

</script>

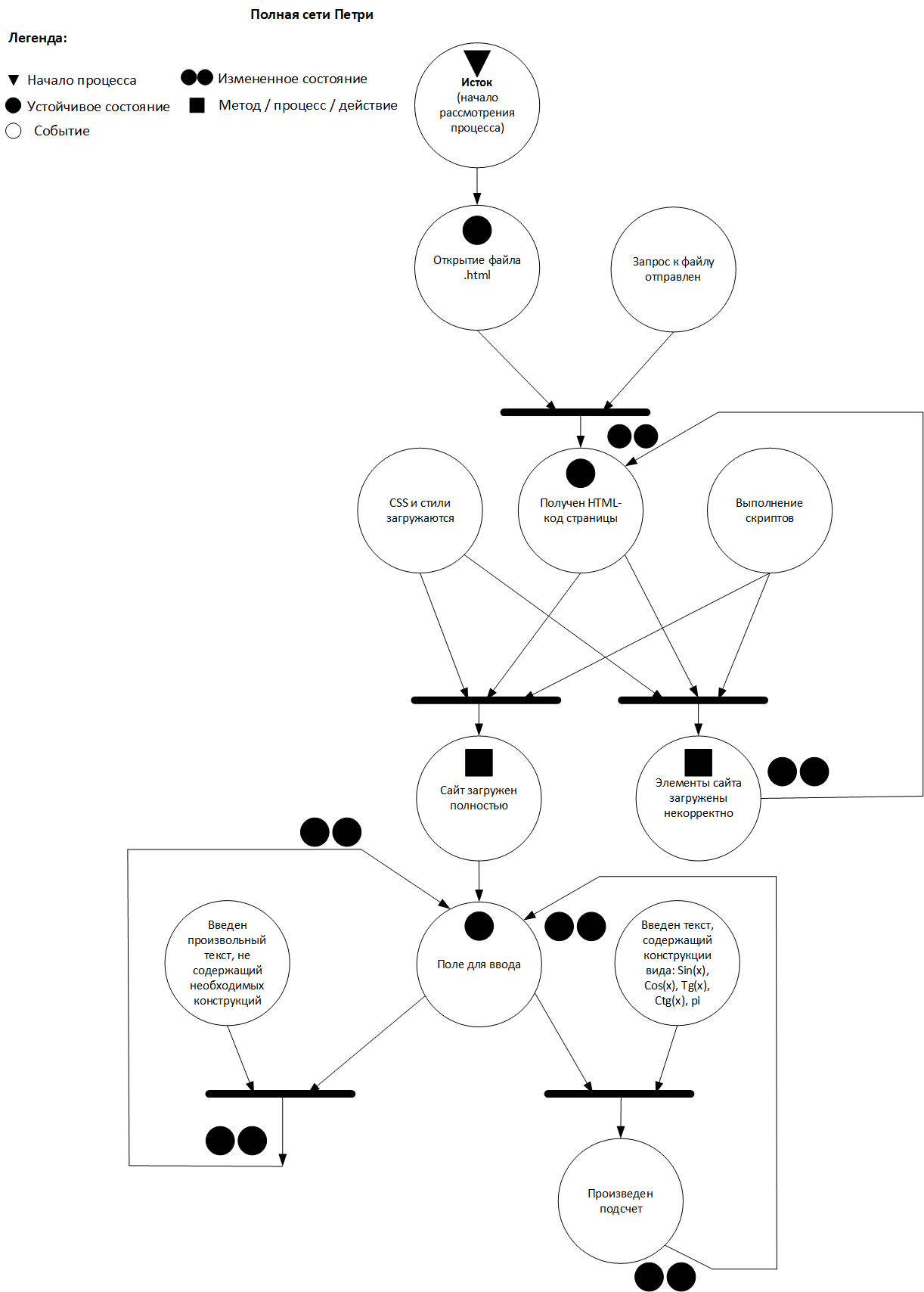
</body>

</html>

На рисунке 10-13 представлено отображение в браузере нашего web-приложения.

|  |
| --- |
| Рисунок 10 – Отображение в браузере |
| Рисунок 11 – Введен произвольный текст |
| Рисунок 12 – Введено «sin(30)» |
| Рисунок 13 – Введено «PI» |

Ниже представлена сеть Петри процесса с момента запуска файла страницы до демонстрации работы витков/хуков/методов жизненного цикла.



# 9. 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.