|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Задание №4**  **по теме «Технология локального подключения фреймворков»**  **по дисциплине «Web-программирование»** | |
|  | Выполнил:  Студент группы ТКИ-542  Волочинский И.О.  Ефремов Д.Ю.  Проверил:  Доцент кафедры УиЗИ, к.т.н., с.н.с  Сафронов А.И. |
| Москва 2024 | |

Оглавление

[1. Цель работы 3](#_gjdgxs)

[2. ФОРМУЛИРОВАКА ЗАДАЧИ 3](#_1fob9te)

[3. Пункт 1 (Web-страница 1) 4](#_3znysh7)

[3.1. Формулировка задачи 4](#_2et92p0)

[3.2. Код web-приложения 4](#_tyjcwt)

[3.3. Результат отображения в браузере 10](#_3dy6vkm)

[3.4. Схема Сетей Петри 11](#_1t3h5sf)

[4. Пункт 1 (Web-страница 1) 12](#_4d34og8)

[4.1. Формулировка задачи 12](#_2s8eyo1)

[4.2. Код web-приложения 12](#_17dp8vu)

[4.3. Результат отображения в браузере 14](#_3rdcrjn)

[4.4. Схема Сетей Петри 15](#_26in1rg)

[5. пункт 3 (Web-старница 3) 16](#_lnxbz9)

[5.1. Формулировка задачи 16](#_35nkun2)

[5.2. Код web-приложения 16](#_1ksv4uv)

[5.3. Результат отображения в браузере 17](#_44sinio)

[5.4. Схема Сетей Петри 18](#_2jxsxqh)

[6. вывод по работе 19](#_3j2qqm3)

1. **Цель работы**

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

1. **ФОРМУЛИРОВАКА ЗАДАЧИ**

**Реализовать:**

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

– *v-for* в файле с именем *index-v-for\_Familiya\_I\_O.html;*

– *watches* в файле с именем *index-watches\_Familiya\_I\_O.html;*

– витки жизненного цикла в файле с именем *index-hooks\_Familiya\_I\_O.html.*

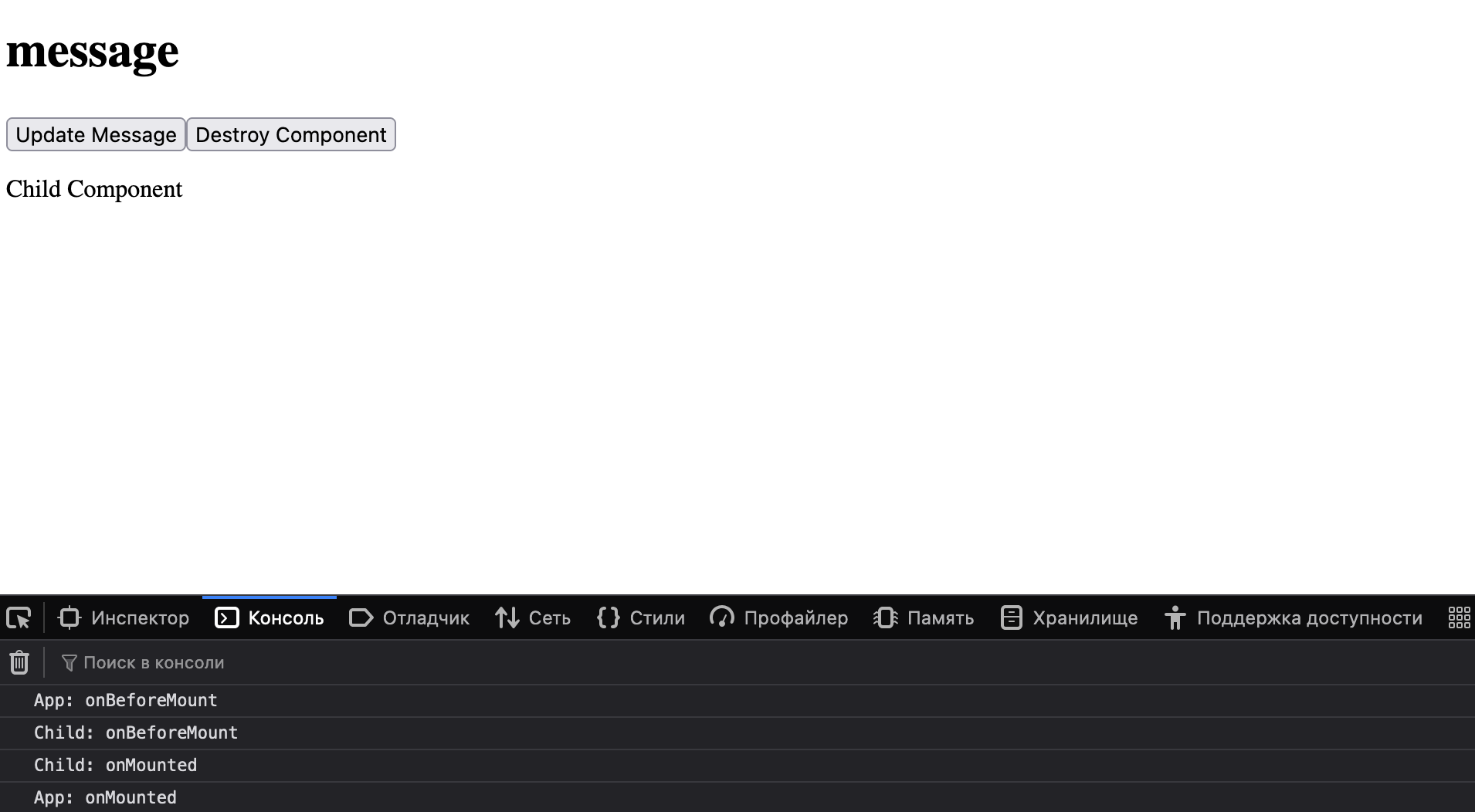
1. **Пункт 1 (Web-страница 1)**
   1. **Формулировка задачи**

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 (Приложение 1). Русифицировать схему жизненного цикла Vue.js 3.x (допускается использование готовой реализации из официальной документации).

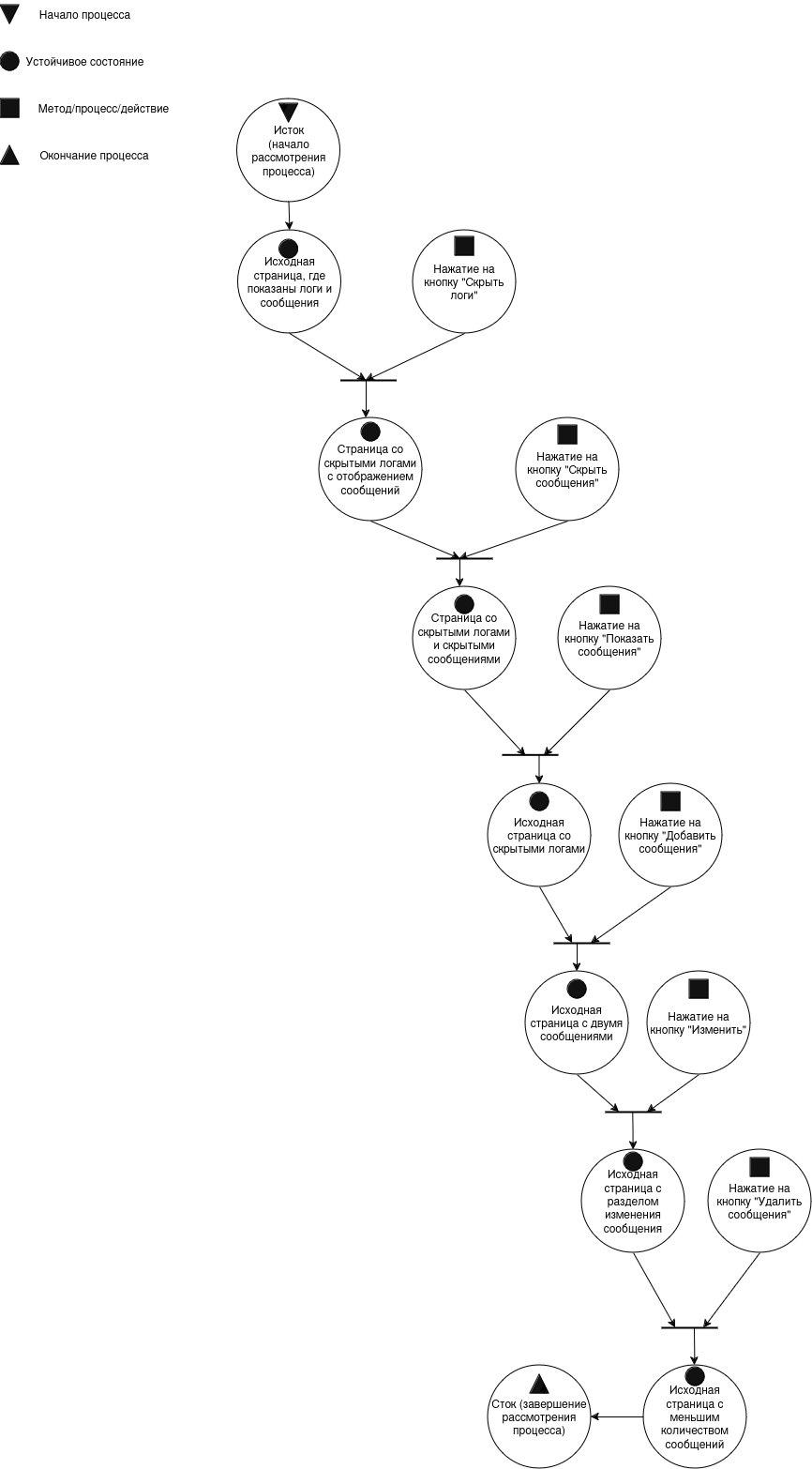
* 1. **Код web-приложения**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Hooks Example</title>  <script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>  </head>  <body>  <div id="app">  <h1>{{ message }}</h1>  <button @click="updateMessage">Update Message</button>  <button @click="destroyComponent">Destroy Component</button>  <div v-if="showComponent">  <child-component></child-component>  </div>  </div>  <script>  const {  createApp,  ref,  onBeforeMount,  onMounted,  onBeforeUpdate,  onUpdated,  onBeforeUnmount,  onUnmounted  } = Vue  const ChildComponent = {  template: `<p>Child Component</p>` ,  setup() {  onBeforeMount(() => console.log('Child: onBeforeMount'));  onMounted(() => console.log('Child: onMounted'));  onBeforeUpdate(() => console.log('Child: onBeforeUpdate'));  onUpdated(() => console.log('Child: onUpdated'));  onBeforeUnmount(() => console.log('Child: onBeforeUnmount'));  onUnmounted(() => console.log('Child: onUnmounted'));  },  };  createApp({  components: { ChildComponent },  setup() {  const message = ref('message');  const showComponent = ref(true);  const updateMessage = () => message.value = 'Message updated!';  const destroyComponent = () => showComponent.value = false;  onBeforeMount(() => console.log('App: onBeforeMount'));  onMounted(() => console.log('App: onMounted'));  onBeforeUpdate(() => console.log('App: onBeforeUpdate'));  onUpdated(() => console.log('App: onUpdated'));  onBeforeUnmount(() => console.log('App: onBeforeUnmount'));  onUnmounted(() => console.log('App: onUnmounted'));  return { message, showComponent, updateMessage, destroyComponent };  }  }).mount("#app")  </script>  </body>  </html> |

* 1. **Результат отображения в браузере**



1. Результат выполнения пункта 1
   1. **Схема Сетей Петри**



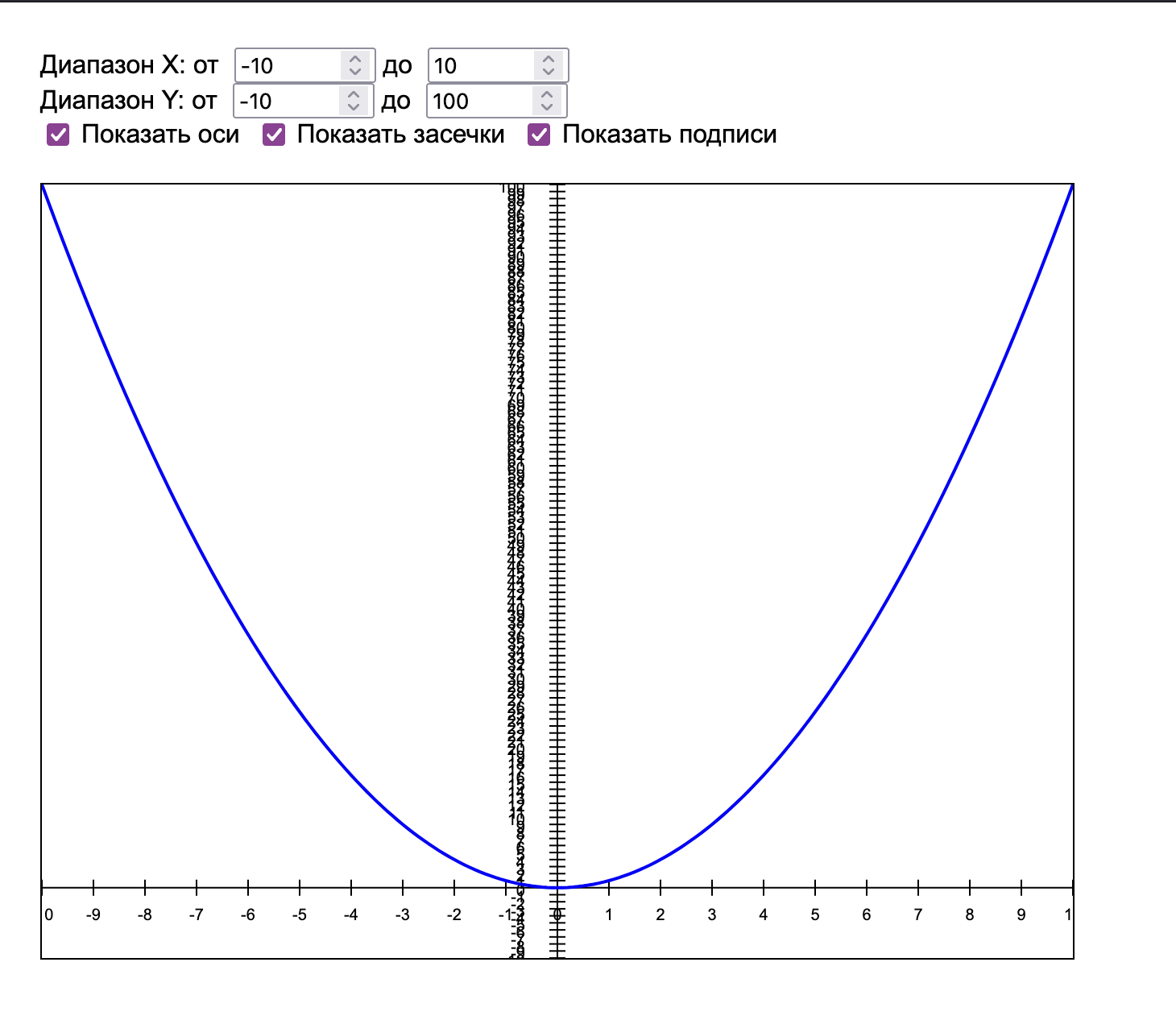
1. Схема Петри пункта 1
2. **Пункт 1 (Web-страница 1)**
   1. **Формулировка задачи**

**Вариант 17.** В области масштабируемой векторной графики <svg> фиксированных размеров 640 на 480 пикселей по ширине и высоте, соответственно, строить отрезками с единичным шагом функцию y(x) = x \* x. Предоставить оператору возможность вводить параметры диапазонов рассмотрения графика функции как по абсциссе, так и по ординате, намечать засечки на осях с численными значениями под ними. Предусмотреть возможность отключения осей по отдельности, отключения подписей с числовыми значениями, отключения засечек. Для каждого состояния видимости / невидимости ввести элемент <input> в режиме checkbox.

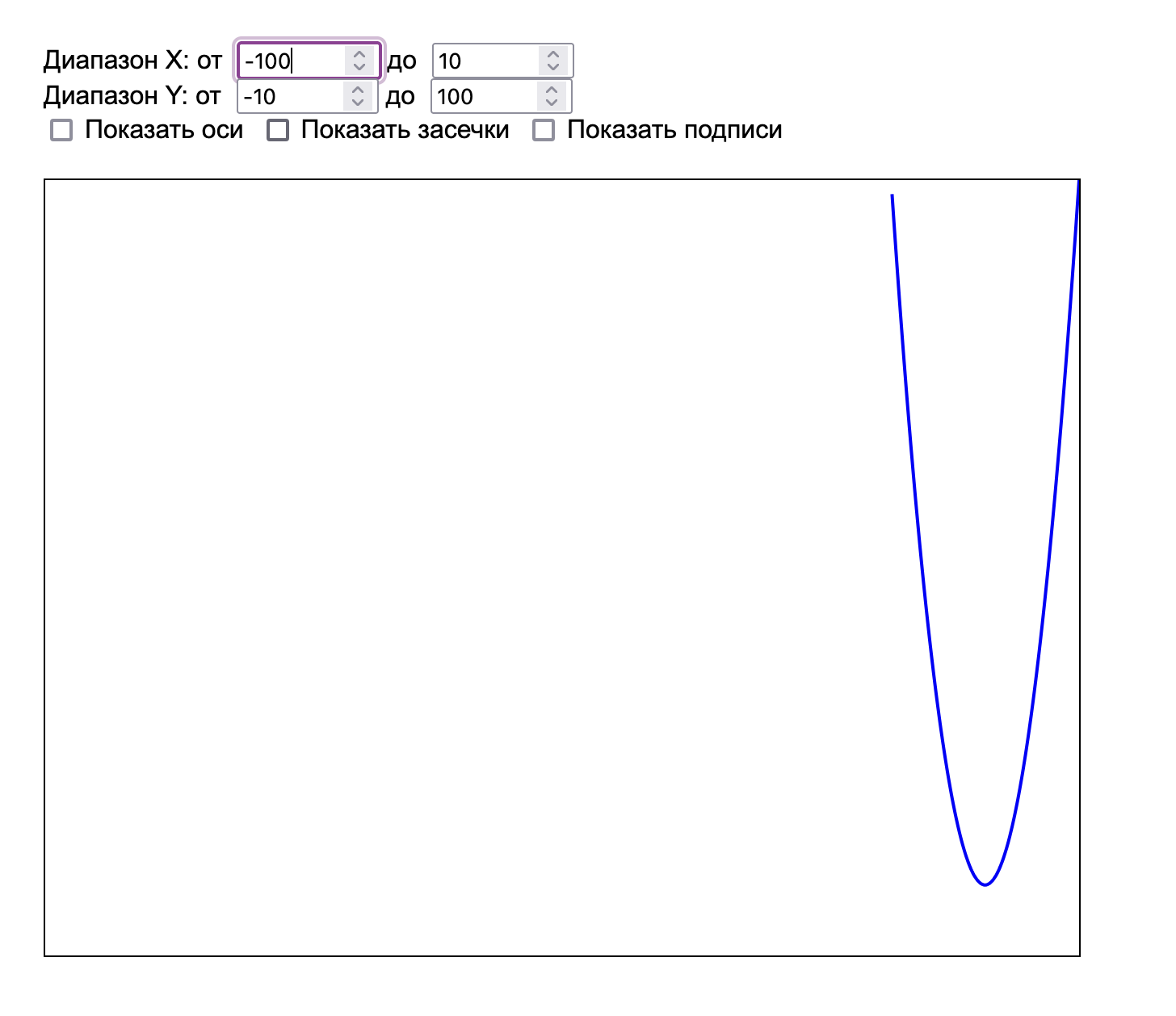
* 1. **Код web-приложения**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Vue.js SVG Graph</title>  <script src="https://cdn.jsdelivr.net/npm/vue@3/dist/vue.global.js"></script>  <style>  body {  font-family: Arial, sans-serif;  padding: 20px;  }  .controls {  margin-bottom: 20px;  }  .checkbox-group {  display: flex;  gap: 10px;  margin-bottom: 10px;  }  input[type="number"] {  width: 80px;  margin-left: 5px;  }  </style>  </head>  <body>  <div id="app">  <div class="controls">  <!-- Диапазоны -->  <div>  <label>  Диапазон X:  от <input type="number" v-model.number="rangeX.min" />  до <input type="number" v-model.number="rangeX.max" />  </label>  </div>  <div>  <label>  Диапазон Y:  от <input type="number" v-model.number="rangeY.min" />  до <input type="number" v-model.number="rangeY.max" />  </label>  </div>  <!-- Флажки для управления видимостью -->  <div class="checkbox-group">  <label>  <input type="checkbox" v-model="showAxes" /> Показать оси  </label>  <label>  <input type="checkbox" v-model="showTicks" /> Показать засечки  </label>  <label>  <input type="checkbox" v-model="showLabels" /> Показать подписи  </label>  </div>  </div>  <!-- SVG график -->  <svg width="640" height="480" viewBox="0 0 640 480" xmlns="http://www.w3.org/2000/svg" style="border: 1px solid black;">  <!-- Оси -->  <line  v-if="showAxes"  x1="0" :x2="width"  :y1="yToSvg(0)" :y2="yToSvg(0)"  stroke="black"  />  <line  v-if="showAxes"  :x1="xToSvg(0)" :x2="xToSvg(0)"  y1="0" :y2="height"  stroke="black"  />  <!-- Засечки и подписи по X -->  <g v-if="showTicks && showAxes">  <g v-for="x in xTicks" :key="'x' + x">  <line  :x1="xToSvg(x)"  :x2="xToSvg(x)"  :y1="yToSvg(0) - 5"  :y2="yToSvg(0) + 5"  stroke="black"  />  <text  v-if="showLabels"  :x="xToSvg(x)"  :y="yToSvg(0) + 20"  font-size="10"  text-anchor="middle">  {{ x }}  </text>  </g>  </g>  <!-- Засечки и подписи по Y -->  <g v-if="showTicks && showAxes">  <g v-for="y in yTicks" :key="'y' + y">  <line  :x1="xToSvg(0) - 5"  :x2="xToSvg(0) + 5"  :y1="yToSvg(y)"  :y2="yToSvg(y)"  stroke="black"  />  <text  v-if="showLabels"  :x="xToSvg(0) - 20"  :y="yToSvg(y) + 5"  font-size="10"  text-anchor="end">  {{ y }}  </text>  </g>  </g>  <!-- График функции -->  <polyline  :points="graphPoints"  fill="none"  stroke="blue"  stroke-width="2"  />  </svg>  </div>  <script>  const App = {  data() {  return {  width: 640, // Ширина SVG  height: 480, // Высота SVG  rangeX: { min: -10, max: 10 }, // Диапазон X  rangeY: { min: -10, max: 100 }, // Диапазон Y  showAxes: true, // Видимость осей  showTicks: true, // Видимость засечек  showLabels: true, // Видимость подписей  };  },  computed: {  // Засечки по X  xTicks() {  return this.generateTicks(this.rangeX.min, this.rangeX.max);  },  // Засечки по Y  yTicks() {  return this.generateTicks(this.rangeY.min, this.rangeY.max);  },  // Точки для графика  graphPoints() {  const points = [];  for (let x = this.rangeX.min; x <= this.rangeX.max; x += 0.1) {  const y = x \* x;  if (y >= this.rangeY.min && y <= this.rangeY.max) {  points.push(`${this.xToSvg(x)},${this.yToSvg(y)}`);  }  }  return points.join(" ");  },  },  methods: {  // Генерация засечек  generateTicks(min, max) {  const ticks = [];  for (let i = Math.ceil(min); i <= Math.floor(max); i++) {  ticks.push(i);  }  return ticks;  },  // Преобразование X в координаты SVG  xToSvg(x) {  const scaleX = this.width / (this.rangeX.max - this.rangeX.min);  return (x - this.rangeX.min) \* scaleX;  },  // Преобразование Y в координаты SVG  yToSvg(y) {  const scaleY = this.height / (this.rangeY.max - this.rangeY.min);  return this.height - (y - this.rangeY.min) \* scaleY;  },  },  };  Vue.createApp(App).mount("#app");  </script>  </body>  </html> |

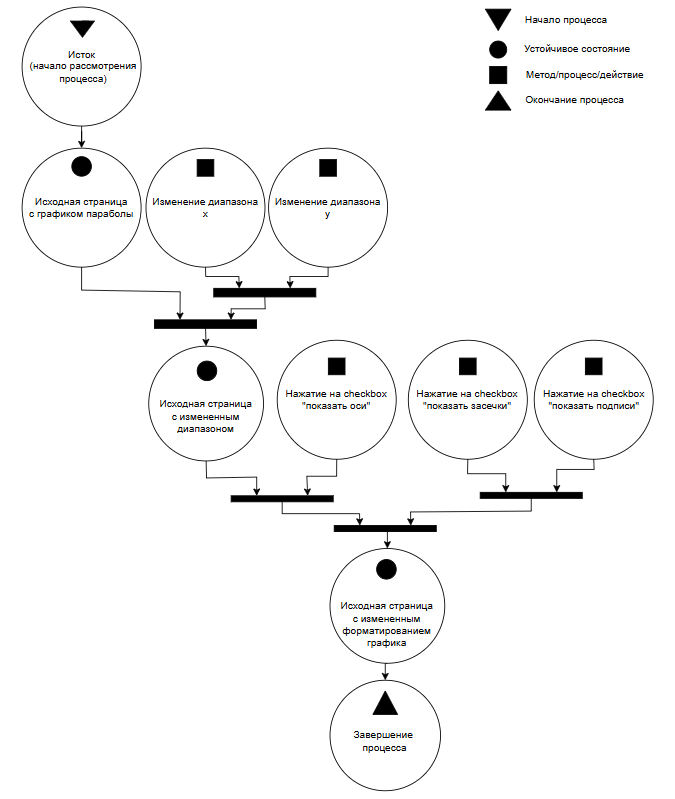
* 1. **Результат отображения в браузере**



1. Результат выполнения пункта 2



1. Результат выполнения пункта 2
   1. **Схема Сетей Петри**



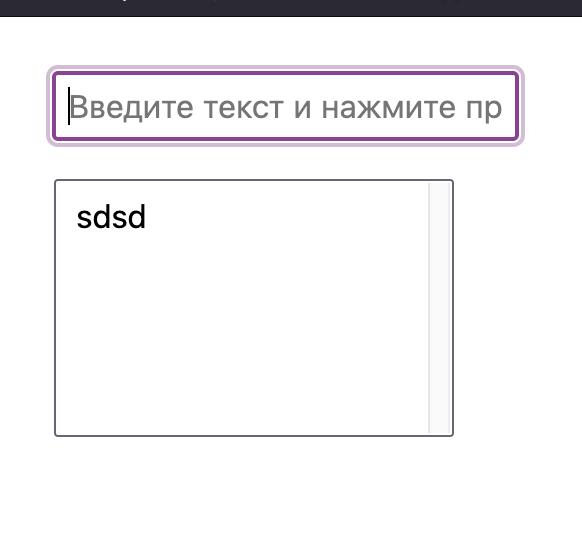
1. Схема Петри пункта 2
2. **пункт 3 (Web-старница 3)**
   1. **Формулировка задачи**

**Вариант 5.**Список <select> наполняется последовательно пунктами, наименование которых вводятся в <input> и подтверждается нажатием на «пробел» («Spacebar»). Выставляется ограничение на ввод – запрещено вводить более 15 символов. Как только очередной вводимый элемент составляет более 15 символов – содержимое списка очищается. Предусмотреть возможность возврата значений списка по «Ctrl» + «z» при фокусе на <input>.

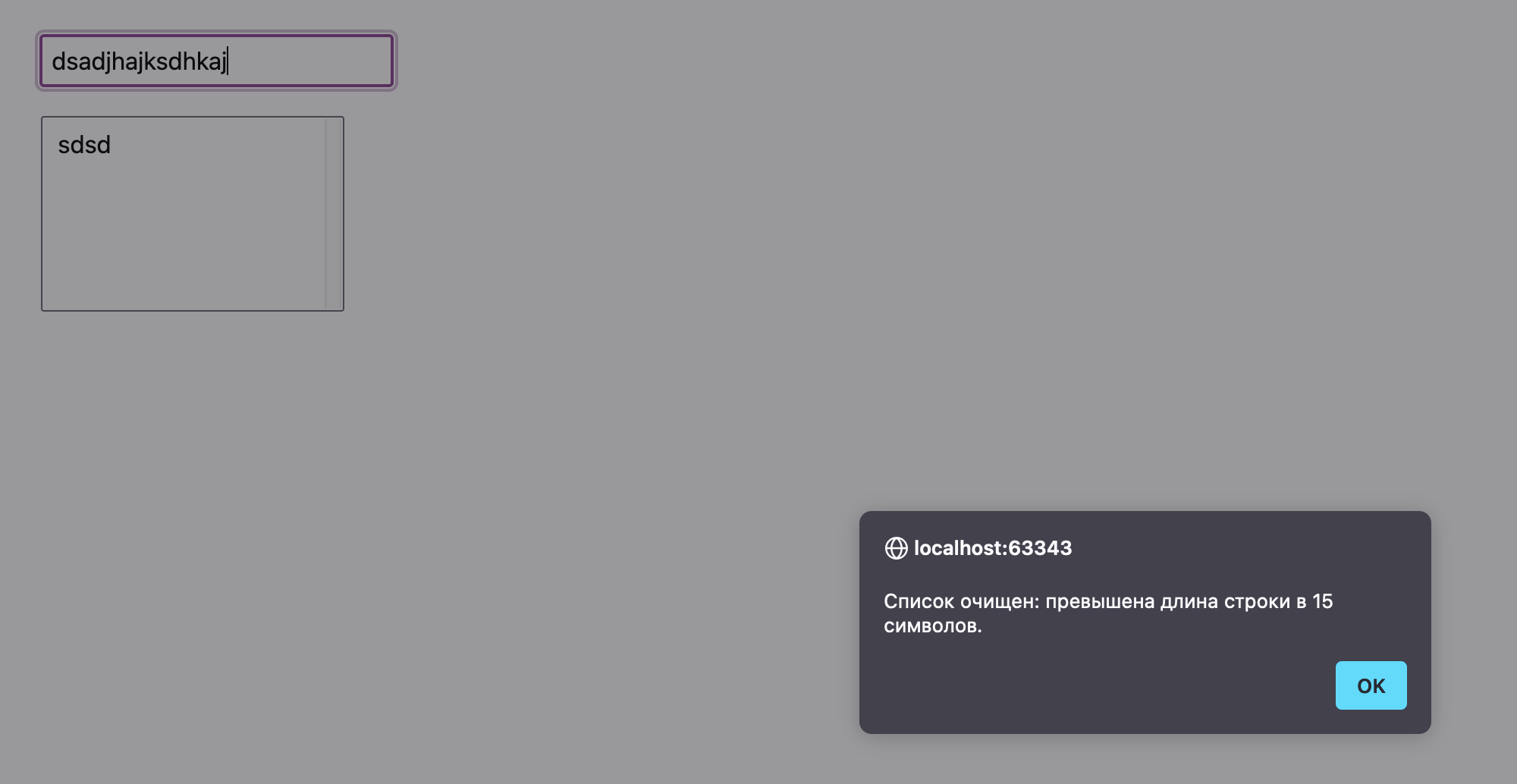
* 1. **Код web-приложения**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <title>Vue.js Select Input with Watch</title>  <script src="https://cdn.jsdelivr.net/npm/vue@3/dist/vue.global.js"></script>  <style>  body {  font-family: Arial, sans-serif;  padding: 20px;  }  input {  padding: 5px;  font-size: 16px;  margin-bottom: 20px;  display: block;  }  select {  padding: 5px;  font-size: 16px;  width: 200px;  }  </style>  </head>  <body>  <div id="app">  <input  type="text"  v-model="currentInput"  placeholder="Введите текст и нажмите пробел"  @keydown.space.prevent="addItem"  @keydown.ctrl.z="undo"  />  <select size="5">  <option v-for="(item, index) in items" :key="index">{{ item }}</option>  </select>  </div>  <script>  const App = {  data() {  return {  currentInput: "", // Текущий ввод текста  items: [], // Список элементов  history: [], // История изменений для отмены  };  },  watch: {  // Наблюдатель за длиной текущего ввода  currentInput(newVal) {  if (newVal.length > 15) {  // Сохранение текущего состояния списка в историю  this.history.push([...this.items]);  // Очищение списка  this.items = [];  // Очистка текущего ввода  this.currentInput = "";  alert("Список очищен: превышена длина строки в 15 символов.");  }  },  },  methods: {  // Добавление нового пункта в список  addItem() {  if (this.currentInput.trim() === "") return;  // Сохранение текущего состояния списка в историю  this.history.push([...this.items]);  // Добавление элемента в список  this.items.push(this.currentInput.trim());  // Очистка поля ввода  this.currentInput = "";  },  // Отмена последнего действия  undo() {  if (this.history.length > 0) {  // Возврат последнего состояния из истории  this.items = this.history.pop();  }  },  },  };  Vue.createApp(App).mount("#app");  </script>  </body>  </html> |

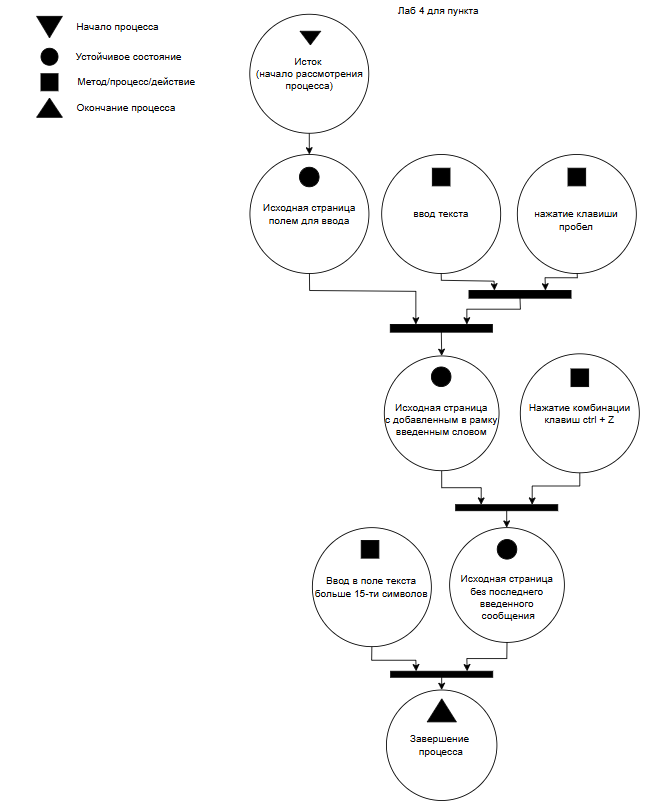
* 1. **Результат отображения в браузере**



1. Результат выполнения пункта 3



1. Результат выполнения пункта 3
   1. **Схема Сетей Петри**



1. Схема Петри пункта 3
2. **вывод по работе**

Мы получили навыки по созданию локальных одностраничных web-приложение под управлением фреймворка Vue.js на языке JavaScript.