

**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**

«Линейный и разветвляющийся вычислительные процессы»

**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](#_Toc180340420)

[2. Формулировка задачи 3](#_Toc180340421)

[3. Технология локального подключения фреймворка Vue.js 5](#_Toc180340422)

[4. Таблица соответствия переменных и методов 6](#_Toc180340423)

[5. Задача 1 7](#_Toc180340424)

[5.1. Код web-приложения 7](#_Toc180340425)

[5.2. Отображение в браузере 7](#_Toc180340426)

[6. Задача 2 10](#_Toc180340427)

[6.1 Код web-приложения 10](#_Toc180340428)

[6.2. Отображение в браузере 12](#_Toc180340429)

[7. Задача 3 14](#_Toc180340430)

[7.1 Код web-приложения 14](#_Toc180340431)

[7.2. Отображение в браузере 19](#_Toc180340432)

[7.3. Сеть Петри 20](#_Toc180340433)

[8. Задача 4 22](#_Toc180340434)

[8.1 Код web-приложения 22](#_Toc180340435)

[8.2. Отображение в браузере 29](#_Toc180340436)

[8.3. Сеть Петри 32](#_Toc180340437)

[9. Вывод 34](#_Toc180340438)

# 1. Цель работы

Разработать четыре простых локальных одностраничных web-приложение (LSPWA) под управлением фреймворка Vue.js на языке JavaScript в соответствии с указаниями вариантов индивидуального задания (покрываются три раздела последовательного вычислительного процесса и один раздел разветвляющегося вычислительного процесса).

# 2. Формулировка задачи

1. Настроить при помощи параметрической подстановки тег масштабируемой векторной графики <*svg*></*svg*>. Ширину и высоту параметрически выставить к значениям 640 и 480 пикселей, соответственно. Параметрически линиями <*line*></*line*> (или полигоном <polygon></polygon>) внутри <*svg*> изобразить равносторонний треугольник, отстоящий от границ <*svg*> на 100 пикселей. Толщина линий составляет 10 пикселей. Цвет линий – зелёный.
2. Для тега настройки шрифта <font></font> менять семейство, размер и цвет шрифта по нажатии по трём соответствующим экранным кнопкам <button></button>:

1. Циклически «Courier New» / «Book Antiqua» / «Tahoma»;

2. Циклически 1 – 6;

3. Циклически «красный» / «зелёный» / «синий» (из числа перечислимых цветовых констант).

1. Продумать опросник на произвольно выбранную тематику, состоящий из десяти вопросов. Для каждого вопроса предлагаются два, три или четыре варианта ответа, из которых всегда один и только один является верным. В качестве активных элементов управления использовать флажки (*input* в режиме *checkbox*). **Важно:** не может быть двух и более одновременно выставленных флажка. Вопросы задаются все сразу. Завершение тестирования реализуется по нажатии на экранную кнопку <*button*><*/button*> «Завершить». Нажатие на кнопку завершение сопровождается блокировкой всех выше расположенных элементов, в том числе и самой кнопки. Без сокрытия. По итогам тестирования выдавать результат успеха прохождения опроса в процентах. Подсчёт: 1 верный ответ – 1 балл.
2. Смоделировать работу автомобильного светофора. На протяжении некоторого времени горит «зелёный» сигнал, затем фиксированное время горит «жёлтый» и, на протяжении третьего времени, горит «красный». Рядом с «красным» и «зелёными» сигналами имеются окна обратного отсчёта до переключения. В разворачиваемых по галочке настройках иметь возможность настраивать времена горения «зелёного» и «красного» сигналов, а также мерцания «жёлтого» сигнала при приостановке светофора. Предусмотреть режимы управления «включён» / «приостановлен» / «выключен», активируемых тремя радиокнопками. Режим «приостановлен» подразумевает мигание жёлтого сигнала светофора.

# 3. 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.

# 4. Таблица соответствия переменных и методов

|  |  |  |
| --- | --- | --- |
| Имя файла задания | Переменные | Методы |
| 1.htm | width, height, margin, lineWidth, lineColor | trianglePoints |
| 2.htm | fontFamily, fontSize, fontColor, fontFamilies, fontSizes, fontColors, currentFontFamilyIndex, currentFontSizeIndex, currentFontColorIndex | changeFontFamily(), changeFontSize(), changeFontColor() |
| 3.htm | questions, selectedAnswers, isSubmitted, score | toggleCheckbox(), submitTest(), calculateScore() |
| 4.htm | mode, greenTime, redTime, yellowBlinkTime, currentLight, greenCounter, redCounter, isYellowBlinking | startTrafficLight(), switchLights(), startYellowBlinking(), stopTrafficLight(), stopYellowBlinking() |

# 5. Задача 1

## 5.1. Код web-приложения

## <div id="app">

## <svg

## v-bind:width="width"

## v-bind:height="height"

## v-bind:viewBox="'0 0 ' + width + ' ' + height">

## 

## <polygon

## v-bind:points="trianglePoints"

## v-bind:stroke="lineColor"

## v-bind:stroke-width="lineWidth"

## fill="none"/>

## </svg>

## </div>

## 

## <script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

## <script>

## new Vue({

## el: '#app',

## data: {

## width: 640, // ширина SVG

## height: 480, // высота SVG

## margin: 100, // отступ от краев для треугольника

## lineWidth: 10, // толщина линий треугольника

## lineColor: 'green', // цвет линий треугольника

## },

## computed: {

## trianglePoints() {

## const w = this.width - 2 \* this.margin; // ширина треугольника

## const h = this.height - 2 \* this.margin; // высота отступа сверху и снизу

## 

## // Координаты вершин треугольника

## const x1 = this.width / 2;

## const y1 = this.margin;

## 

## const x2 = this.margin;

## const y2 = this.height - this.margin;

## 

## const x3 = this.width - this.margin;

## const y3 = this.height - this.margin;

## 

## // Возвращаем координаты в формате "x1,y1 x2,y2 x3,y3"

## return `${x1},${y1} ${x2},${y2} ${x3},${y3}`;

## }

## }

## });

## </script>

## 5.2. Отображение в браузере

На рисунке 1 представлено отображение в браузере нашего web-приложения согласно заданию.

|  |
| --- |
| Рисунок 1 – Отображение в браузере |

# 6. Задача 2

## 6.1 Код web-приложения

<div id="app">

<!-- Кнопки, которые вызывают методы через события нажатия (v-on:click) -->

<div>

<button v-on:click="changeFontFamily">Изменить семейство шрифта</button>

<button v-on:click="changeFontSize">Изменить размер шрифта</button>

<button v-on:click="changeFontColor">Изменить цвет шрифта</button>

</div>

<!-- Тег <font> с параметрически меняемыми атрибутами через методы -->

<font

v-bind:face="fontFamily"

v-bind:size="fontSize"

v-bind:color="fontColor">

Пример текста с параметрическими изменениями шрифта.

</font>

</div>

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

<script>

new Vue({

el: '#app',

data: {

// Инициализация параметров шрифта

fontFamily: 'Courier New', // Начальное семейство шрифта

fontSize: 3, // Начальный размер шрифта

fontColor: 'red', // Начальный цвет шрифта

// Варианты значений для циклической смены

fontFamilies: ['Courier New', 'Book Antiqua', 'Tahoma'], // Семейства шрифтов

fontSizes: [1, 2, 3, 4, 5, 6], // Размеры шрифта

fontColors: ['red', 'green', 'blue'], // Цвета шрифта

// Индексы для отслеживания текущего состояния циклических параметров

currentFontFamilyIndex: 0, // Текущий индекс семейства шрифта

currentFontSizeIndex: 2, // Текущий индекс размера шрифта (по умолчанию 3)

currentFontColorIndex: 0, // Текущий индекс цвета шрифта

},

methods: {

// Метод изменения семейства шрифта

changeFontFamily() {

// Циклическое переключение индекса семейства шрифта

this.currentFontFamilyIndex = (this.currentFontFamilyIndex + 1) % this.fontFamilies.length;

this.fontFamily = this.fontFamilies[this.currentFontFamilyIndex]; // Обновляем семейство шрифта

},

// Метод изменения размера шрифта

changeFontSize() {

// Циклическое переключение индекса размера шрифта

this.currentFontSizeIndex = (this.currentFontSizeIndex + 1) % this.fontSizes.length;

this.fontSize = this.fontSizes[this.currentFontSizeIndex]; // Обновляем размер шрифта

},

// Метод изменения цвета шрифта

changeFontColor() {

// Циклическое переключение индекса цвета шрифта

this.currentFontColorIndex = (this.currentFontColorIndex + 1) % this.fontColors.length;

this.fontColor = this.fontColors[this.currentFontColorIndex]; // Обновляем цвет шрифта

}

}

});

</script>

## 6.2. Отображение в браузере

На рисунках 2-4 представлено отображение в браузере нашего web-приложения согласно заданию.

|  |
| --- |
| Рисунок 2 – Отображение в браузере (1 вариант) |
| Рисунок 3 – Отображение в браузере (2 вариант) |

|  |
| --- |
| Рисунок 4 – Отображение в браузере (3 вариант) |

# 7. Задача 3

## 7.1 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 Test</title>

<!-- Подключение Vue.js -->

<script src="https://unpkg.com/vue@3.2.47/dist/vue.global.js"></script>

</head>

<body>

<div id="app">

<div v-for="(question, index) in questions" :key="index">

<p>{{ index + 1 }}. {{ question.text }}</p>

<div v-for="(option, optIndex) in question.options" :key="optIndex">

<input type="checkbox"

:name="'question-' + index"

:disabled="isSubmitted"

:checked="selectedAnswers[index] === optIndex"

@change="toggleCheckbox(index, optIndex)">

<label>{{ option }}</label>

</div>

</div>

<button @click="submitTest" :disabled="isSubmitted">Завершить</button>

<p v-if="isSubmitted">Ваш результат: {{ score }}%</p>

</div>

<script>

const app = Vue.createApp({

data() {

return {

questions: [

{

text: 'Какого цвета небо?',

options: ['Синее', 'Зелёное', 'Красное'],

correctAnswer: 0

},

{

text: 'Сколько дней в неделе?',

options: ['5', '6', '7'],

correctAnswer: 2

},

{

text: 'Как называется столица Франции?',

options: ['Берлин', 'Париж', 'Рим'],

correctAnswer: 1

},

{

text: 'Какое животное является символом России?',

options: ['Лев', 'Медведь', 'Тигр'],

correctAnswer: 1

},

{

text: 'Какой самый большой океан на Земле?',

options: ['Атлантический', 'Индийский', 'Тихий'],

correctAnswer: 2

},

{

text: 'Сколько планет в Солнечной системе?',

options: ['7', '8', '9'],

correctAnswer: 1

},

{

text: 'Какой химический элемент обозначается символом O?',

options: ['Кислород', 'Водород', 'Углерод'],

correctAnswer: 0

},

{

text: 'Как называется самая высокая гора в мире?',

options: ['Эверест', 'Килиманджаро', 'Монблан'],

correctAnswer: 0

},

{

text: 'Какой самый маленький континент?',

options: ['Европа', 'Австралия', 'Африка'],

correctAnswer: 1

},

{

text: 'Сколько градусов в прямом угле?',

options: ['90', '180', '360'],

correctAnswer: 0

}

],

selectedAnswers: Array(10).fill(null),

isSubmitted: false,

score: 0

};

},

methods: {

toggleCheckbox(questionIndex, optionIndex) {

// Снимаем выбор, если уже был выбран этот ответ

if (this.selectedAnswers[questionIndex] === optionIndex) {

this.selectedAnswers[questionIndex] = null;

} else {

// Обеспечиваем выбор только одного флажка

this.selectedAnswers[questionIndex] = optionIndex;

}

},

submitTest() {

if (this.selectedAnswers.includes(null)) {

alert("Пожалуйста, ответьте на все вопросы.");

return;

}

this.isSubmitted = true;

this.calculateScore();

},

calculateScore() {

let correctAnswersCount = 0;

this.questions.forEach((question, index) => {

if (this.selectedAnswers[index] === question.correctAnswer) {

correctAnswersCount++;

}

});

this.score = Math.round((correctAnswersCount / this.questions.length) \* 100);

}

}

});

app.mount('#app');

</script>

</body>

</html>

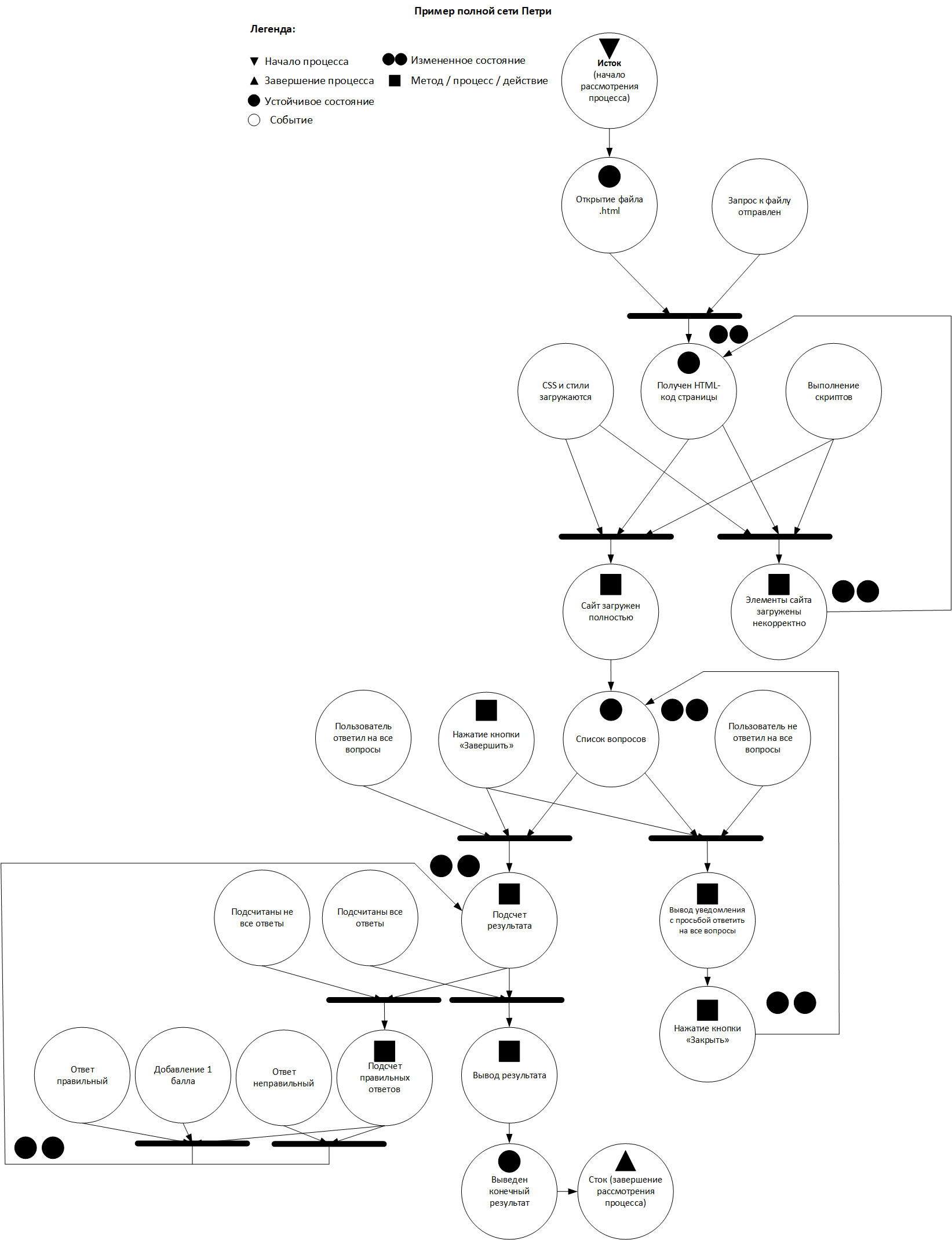
## 7.2. Отображение в браузере

На рисунках 5-7 представлено отображение в браузере нашего web-приложения согласно заданию.

|  |
| --- |
| Рисунок 5 – Отображение в браузере (начальное состояние) |
| Рисунок 6 – Отображение в браузере (попытка завершить тест без ответа на все вопросы) |
| Рисунок 7 – Отображение в браузере (завершенный тест) |

## 7.3. Сеть Петри

Ниже представлена сеть Петри процесса с момента запуска файла страницы до завершения теста.



# 8. Задача 4

## 8.1 Код web-приложения

<div id="app">

<!-- Радиокнопки для управления светофором -->

<div>

<label>

<input type="radio" v-model="mode" value="on"> Включён

</label>

<label>

<input type="radio" v-model="mode" value="paused"> Приостановлен

</label>

<label>

<input type="radio" v-model="mode" value="off"> Выключен

</label>

</div>

<!-- Настройки времени для светофора -->

<details>

<summary>Настройки времени</summary>

<div>

<label>Время зелёного сигнала (сек):</label>

<input type="number" v-model="greenTime" min="5">

</div>

<div>

<label>Время красного сигнала (сек):</label>

<input type="number" v-model="redTime" min="5">

</div>

<div>

<label>Время мерцания жёлтого сигнала (сек):</label>

<input type="number" v-model="yellowBlinkTime" min="1">

</div>

</details>

<!-- Светофор -->

<div class="traffic-light">

<div v-if="mode === 'on'">

<!-- Зелёный сигнал -->

<div class="light green" v-if="currentLight === 'green'">

Зелёный ({{ greenCounter }}) сек

</div>

<!-- Жёлтый сигнал -->

<div class="light yellow" v-if="currentLight === 'yellow'">

Жёлтый

</div>

<!-- Красный сигнал -->

<div class="light red" v-if="currentLight === 'red'">

Красный ({{ redCounter }}) сек

</div>

</div>

<!-- Приостановлен: мигание жёлтого сигнала -->

<div v-if="mode === 'paused'">

<div class="light yellow" :class="{ blinking: isYellowBlinking }">

Жёлтый (мигает)

</div>

</div>

<!-- Выключен: все сигналы отключены -->

<div v-if="mode === 'off'">

<p>Светофор выключен</p>

</div>

</div>

</div>

<script src="https://cdn.jsdelivr.net/npm/vue@2"></script>

<script>

new Vue({

el: '#app',

data: {

// Состояние светофора

mode: 'off', // 'on', 'paused', 'off'

// Настройки времени

greenTime: 10, // Время зелёного сигнала

redTime: 10, // Время красного сигнала

yellowBlinkTime: 2, // Время мерцания жёлтого сигнала

// Внутренние переменные для отслеживания текущего состояния светофора

currentLight: 'green', // Текущий сигнал ('green', 'yellow', 'red')

greenCounter: 10, // Таймер обратного отсчёта для зелёного сигнала

redCounter: 10, // Таймер обратного отсчёта для красного сигнала

isYellowBlinking: false, // Флаг для мигания жёлтого сигнала

},

mounted() {

this.startTrafficLight();

},

watch: {

// Перезапуск светофора при изменении режима

mode(newVal) {

if (newVal === 'on') {

this.startTrafficLight();

} else if (newVal === 'paused') {

this.startYellowBlinking();

} else {

this.stopTrafficLight();

}

},

// Перезапуск светофора при изменении времени

greenTime(newVal) {

this.greenCounter = newVal;

},

redTime(newVal) {

this.redCounter = newVal;

}

},

methods: {

// Запуск работы светофора

startTrafficLight() {

this.stopYellowBlinking();

this.currentLight = 'green';

this.greenCounter = this.greenTime;

this.switchLights();

},

// Переключение сигналов светофора

switchLights() {

if (this.mode !== 'on') return;

if (this.currentLight === 'green') {

this.greenCounter--;

if (this.greenCounter <= 0) {

this.currentLight = 'yellow';

setTimeout(() => {

this.currentLight = 'red';

this.redCounter = this.redTime;

this.switchLights();

}, 2000); // Жёлтый горит 2 секунды

} else {

setTimeout(this.switchLights, 1000);

}

} else if (this.currentLight === 'red') {

this.redCounter--;

if (this.redCounter <= 0) {

this.currentLight = 'green';

this.greenCounter = this.greenTime;

this.switchLights();

} else {

setTimeout(this.switchLights, 1000);

}

}

},

// Мигание жёлтого сигнала в режиме "Приостановлен"

startYellowBlinking() {

this.isYellowBlinking = true;

setInterval(() => {

this.isYellowBlinking = !this.isYellowBlinking;

}, this.yellowBlinkTime \* 1000);

},

// Остановка работы светофора

stopTrafficLight() {

this.stopYellowBlinking();

},

// Остановка мигания жёлтого сигнала

stopYellowBlinking() {

clearInterval(this.yellowBlinkInterval);

this.isYellowBlinking = false;

}

}

});

</script>

<style>

.traffic-light {

width: 100px;

text-align: center;

}

.light {

width: 100%;

height: 100px;

margin: 10px 0;

line-height: 100px;

font-size: 1.5em;

font-weight: bold;

color: white;

border-radius: 50%;

}

.green {

background-color: green;

}

.yellow {

background-color: yellow;

color: black;

}

.red {

background-color: red;

}

.blinking {

animation: blink 1s infinite;

}

@keyframes blink {

50% {

opacity: 0;

}

}

</style>

## 8.2. Отображение в браузере

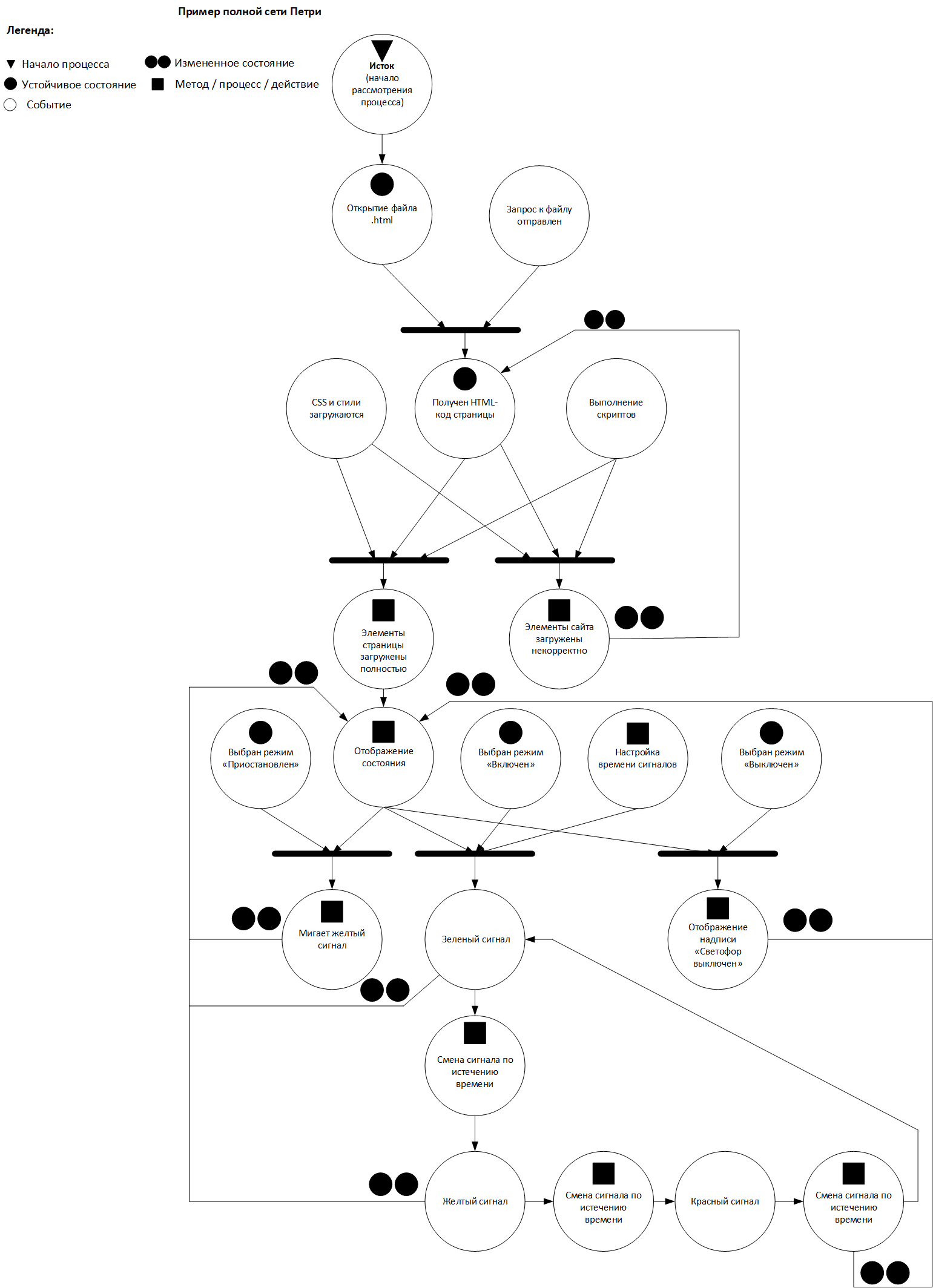
На рисунках 8-14 представлено отображение в браузере нашего web-приложения. Согласно заданию, введенный в текстовое поле текст, должен перемещаться внутри таблицы с помощью стрелок. Как только череда возможных выравниваний по горизонтали (по левому краю, по центру, по правому краю) или по вертикали (сверху, посередине, снизу) завершается – надпись переходит в соседнюю ячейку таблицы. Необходимо исключить возможность текста выходить за пределы таблицы.

|  |
| --- |
| Рисунок 8– Отображение в браузере (начальное состояние) |
| Рисунок 9 – Отображение в браузере (светофор включен, горит зеленый) |

|  |
| --- |
| Рисунок 10 – Отображение в браузере (светофор работает, горит желтый) |
| *Рисунок 11 – Отображение в браузере (светофор работает, горит красный)* |
| Рисунок 12 – Отображение в браузере (настройки времени) |
| *Рисунок 13 – Отображение в браузере (светофор приостановлен, мигает желтый)* |

## 8.3. Сеть Петри

Ниже представлена сеть Петри процесса с момента запуска файла страницы.



# 9. Вывод

В данной работе были получены навыки работы с фреймворком Vue.js на языке JavaScript. Поставленные задачи выполнены в полном объёме.