ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»  
(РУТ (МИИТ))

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

ОТЧЁТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

По дисциплине «Web-программирование»

ВАРИАНТ 4

Выполнили: ст. гр. ТКИ-541

Поткин Дмитрий

Галыба Леонид

Проверил: к.т.н., доц. Сафронов А.И.

Москва 2024

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. Формулировка 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;*

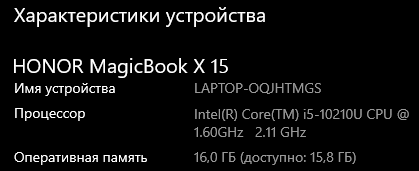
– *watches* в файле с именем *index-watches\_Familiya\_I\_O.html;*

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

**Задания по варианту:**

* 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*
* 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 «$».
* Подобрать шесть идентичных по ширине и высоте иллюстраций, выводимых в <*img*> 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».

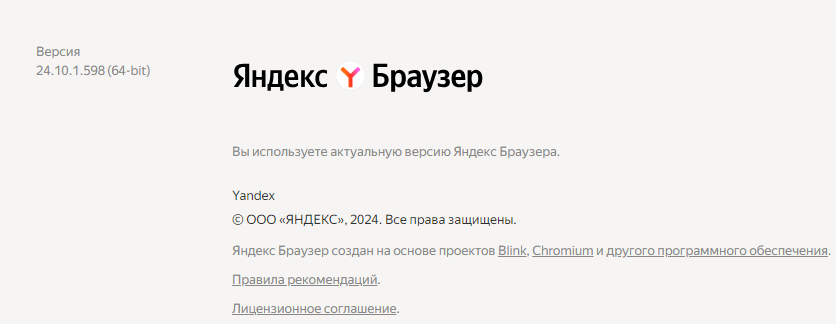
1. Спецификация оборудования, на котором выполнялась работа



Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Информация об используемом браузере

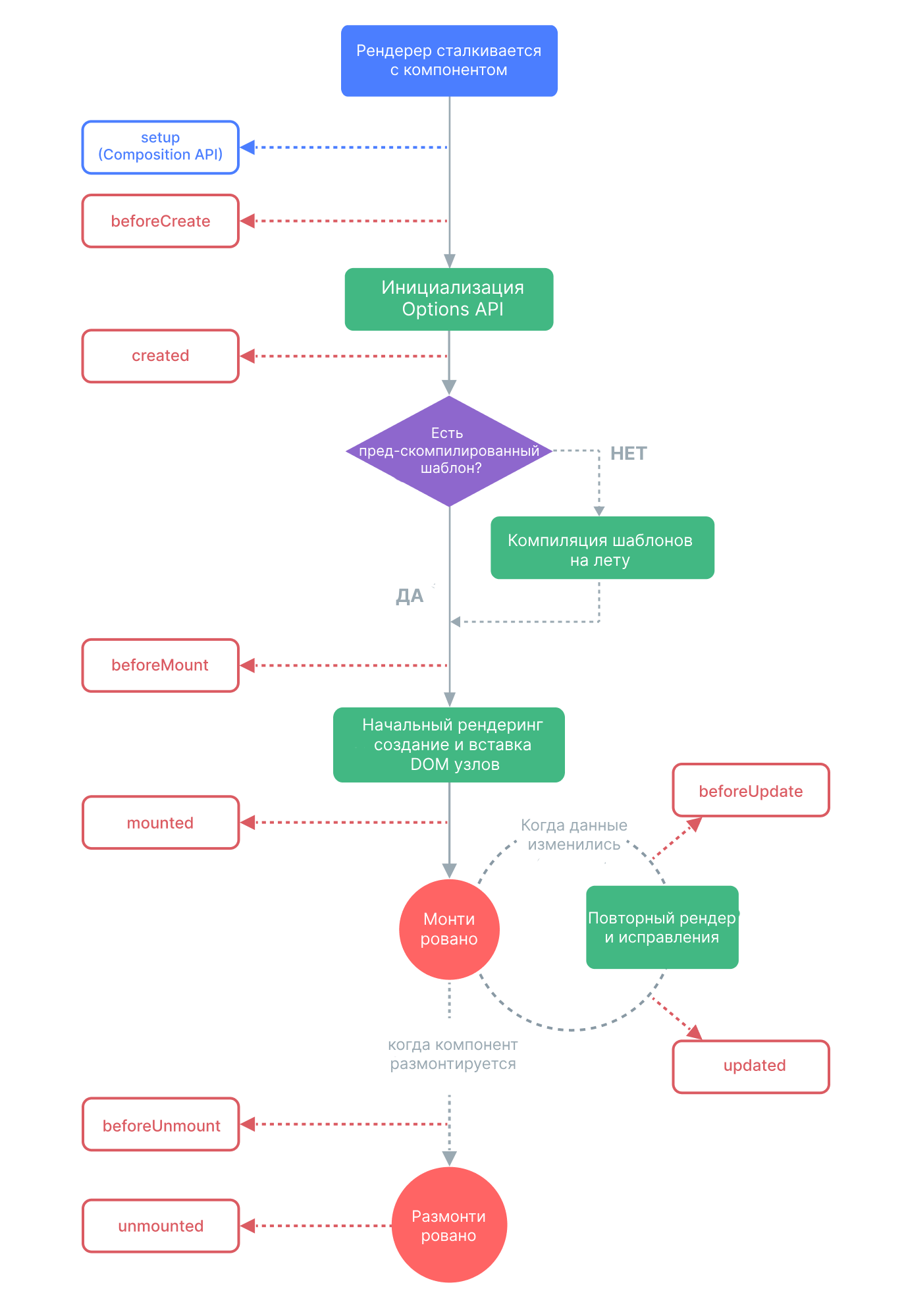


1. Технология локального подключения Vue.js

В работе была использована vue.js 3 версии.

К каждому файлу задания была подключена сборка vue.js, установленная с источника <https://unpkg.com/vue@3/dist/vue.global.js>.

1. Русифицированная схема жизненного цикла *Vue.js*.



1. Таблица соотвествия переменных и методов, используемых в web-приложении

|  |  |  |
| --- | --- | --- |
| Задание | Переменная/метод | Описание |
| 1 | logLifecycle() | Регистрация событий жизненного цикла |
| 1 | beforeCreate() | Хук жизненного цикла, вызывается до инициализации данных компонента |
| 1 | created() | Хук жизненного цикла, вызывается после инициализации данных компонента |
| 1 | beforeMount() | Хук жизненного цикла, вызывается перед монтированием компонента в DOM |
| 1 | mounted() | Хук жизненного цикла, вызывается после того, как компонент смонтирован в DOM |
| 1 | beforeUpdate() | Хук жизненного цикла, вызывается перед обновлением компонента |
| 1 | updated() | Хук жизненного цикла, вызывается после обновления компонента в DOM |
| 1 | beforeUnmount() | Хук жизненного цикла, вызывается перед размонтированием компонента |
| 1 | unmounted() | Хук жизненного цикла, вызывается после удаления компонента из DOM |
| 2 | dimension | Хранит размерность кубического массива, вводимую пользователем. |
| 2 | cubeArray | Хранит кубический массив размером dimension x dimension x dimension |
| 2 | spiralArray | Хранит массив размером dimension x dimension, который представляет данные кубического массива в спиральном порядке |
| 2 | flattenedArray | Плоский (одномерный) массив, созданный из всех значений кубического массива |
| 2 | getRandomInt() | Генерирует случайное целое число от -999 до 999 |
| 2 | generateArray() | Создаёт кубический массив размером dimension x dimension x dimension, заполненный случайными значениями |
| 2 | createSpiral() | Преобразует кубический массив в двумерный массив размером dimension x dimension, представляя его в спиральном порядке. |
| 3 | imageName | Имя файла |
| 3 | imageSrc | Путь к файлу |

1. Содержательная часть по виткам жизненного цикла
   1. Код web-приложения

<!DOCTYPE html>

<html>

<head>

<script src="vue.global.js"></script> <!-- Подключение Vue.js -->

</head>

<body>

<div id="app"></div>

<script>

const logLifecycle = (name, hook) => console.log(`${name} component: ${hook}`);

const Message = {

template: `<div class="component"><p>{{ message }}</p></div>`,

data: () => ({ message: 'Lifecycle Vue.js' }),

beforeCreate() { logLifecycle('Message', 'beforeCreate'); },

created() { logLifecycle('Message', 'created'); },

beforeMount() { logLifecycle('Message', 'beforeMount'); },

mounted() { logLifecycle('Message', 'mounted'); },

beforeUpdate() { logLifecycle('Message', 'beforeUpdate'); },

updated() { logLifecycle('Message', 'updated'); },

beforeUnmount() { logLifecycle('Message', 'beforeUnmount'); },

unmounted() { logLifecycle('Message', 'unmounted'); }

};

const Counter = {

template: `

<div class="component">

<p>Значение: {{ count }}</p>

<button @click="count++">Увеличить</button>

</div>

`,

data: () => ({ count: 0 }),

beforeCreate() { logLifecycle('Counter', 'beforeCreate'); },

created() { logLifecycle('Counter', 'created'); },

beforeMount() { logLifecycle('Counter', 'beforeMount'); },

mounted() { logLifecycle('Counter', 'mounted'); },

beforeUpdate() { logLifecycle('Counter', 'beforeUpdate'); },

updated() { logLifecycle('Counter', 'updated'); },

beforeUnmount() { logLifecycle('Counter', 'beforeUnmount'); },

unmounted() { logLifecycle('Counter', 'unmounted'); }

};

const App = {

template: `

<div>

<Counter />

<Message />

</div>

`,

components: { Counter, Message },

beforeCreate() { logLifecycle('App', 'beforeCreate'); },

created() { logLifecycle('App', 'created'); },

beforeMount() { logLifecycle('App', 'beforeMount'); },

mounted() { logLifecycle('App', 'mounted'); },

beforeUpdate() { logLifecycle('App', 'beforeUpdate'); },

updated() { logLifecycle('App', 'updated'); },

beforeUnmount() { logLifecycle('App', 'beforeUnmount'); },

unmounted() { logLifecycle('App', 'unmounted'); }

};

Vue.createApp(App).mount('#app');

</script>

</body>

</html>

* 1. Отображение в браузере

|  |
| --- |
|  |
| 1. – Отображение в браузере |

* 1. Сеть Петри

|  |
| --- |
|  |
| 1. – Полная сеть Петри |

1. Содержательная часть по циклической отрисовке
   1. Код web-приложения

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

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

<title>Кубический массив на Vue.js</title>

<script src="vue.global.js"></script> <!-- Подключение Vue.js -->

<style>

body {

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

align-items: center;

margin: 0;

padding: 20px;

}

h2 {

margin-bottom: 20px;

}

input {

margin-bottom: 20px;

padding: 5px;

font-size: 16px;

}

table {

border-collapse: collapse;

margin: 10px 0;

}

.cell {

border: 1px solid black;

padding: 5px;

width: 50px;

text-align: center;

}

p {

font-weight: bold;

margin-top: 20px;

max-width: 100vw;

}

</style>

</head>

<body>

<div id="app">

<h2>Кубический массив и спиральный вывод</h2>

<input

type="number"

min="1"

v-model="dimension"

@blur="generateArray"

placeholder="Введите размерность массива"

/>

<table v-if="spiralArray.length">

<tr v-for="(row, rowIndex) in spiralArray" :key="'row-' + rowIndex">

<td v-for="(cell, cellIndex) in row" :key="'cell-' + rowIndex + '-' + cellIndex" class="cell">

{{ cell }}

</td>

</tr>

</table>

<p v-if="flattenedArray.length">{{ flattenedArray.join('$') }}</p>

</div>

<script>

const { createApp, ref, computed } = Vue;

createApp({

setup() {

const dimension = ref(0);

const cubeArray = ref([]);

const spiralArray = ref([]);

// Генерация случайного целого числа от -999 до 999

const getRandomInt = () => Math.floor(Math.random() \* 1999) - 999;

const generateArray = () => {

const size = dimension.value;

if (size < 1) return;

// Заполняем массив случайными значениями

cubeArray.value = Array.from({ length: size }, () =>

Array.from({ length: size }, () =>

Array.from({ length: size }, () => getRandomInt())

)

);

// Преобразуем массив в спиральное представление

spiralArray.value = createSpiral(cubeArray.value, size);

};

// Функция создания спирали

const createSpiral = (array, size) => {

const result = Array.from({ length: size }, () => Array(size).fill(""));

let top = 0, bottom = size - 1, left = 0, right = size - 1;

let idx = 0;

while (top <= bottom && left <= right) {

for (let i = left; i <= right; i++) result[top][i] = array.flat(2)[idx++];

top++;

for (let i = top; i <= bottom; i++) result[i][right] = array.flat(2)[idx++];

right--;

if (top <= bottom) for (let i = right; i >= left; i--) result[bottom][i] = array.flat(2)[idx++];

bottom--;

if (left <= right) for (let i = bottom; i >= top; i--) result[i][left] = array.flat(2)[idx++];

left++;

}

return result;

};

// Плоский массив для вывода по индексу

const flattenedArray = computed(() => {

return cubeArray.value

.flatMap(layer => layer.flatMap(row => row))

.map((num) => num.toString());

});

return {

dimension,

spiralArray,

flattenedArray,

generateArray,

};

},

}).mount("#app");

</script>

</body>

</html>

* 1. Отображение в браузере

|  |
| --- |
|  |
| 1. – Отображение в браузере |

* 1. Сеть Петри

|  |
| --- |
|  |
| 1. – Полная сеть Петри |

1. Содержательная часть по наблюдателям
   1. Код web-приложения

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

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

<title>Выбор изображения на Vue.js</title>

<script src="vue.global.js"></script> <!-- Подключение Vue.js локально -->

<style>

body {

font-family: Arial, sans-serif;

display: flex;

flex-direction: column;

align-items: center;

margin: 0;

padding: 20px;

}

h2 {

margin-bottom: 20px;

text-align: center;

}

input {

margin-bottom: 20px;

padding: 5px;

font-size: 16px;

display: block;

width: 98%;

}

img {

border: 1px solid #ddd;

margin-top: 10px;

}

</style>

</head>

<body>

<div id="app">

<h2>Выбор изображения</h2>

<img :src="imageSrc" alt="Изображение" />

<input type="text" v-model="imageName" placeholder="Введите название изображения" />

</div>

<script>

const { createApp, ref, watch } = Vue;

createApp({

setup() {

const imageName = ref('');

const imageSrc = ref('images/7.png'); // Путь к изображению по умолчанию

// Список доступных изображений

const availableImages = [

'images/1.png',

'images/2.png',

'images/3.png',

'images/4.png',

'images/5.png',

'images/6.png'

];

// Наблюдатель за изменением imageName

watch(imageName, (newName) => {

const imagePath = `images/${newName}.png`;

if (availableImages.includes(imagePath)) {

imageSrc.value = imagePath;

} else {

imageSrc.value = 'images/7.png';

}

});

return {

imageName,

imageSrc

};

}

}).mount('#app');

</script>

</body>

</html>

* 1. Отображение в браузере

|  |
| --- |
|  |
| 1. – Состояние при просмотре четвертой картинки |

|  |
| --- |
|  |
| 1. – Состояние при просмотре несуществующей картинки |

* 1. Сеть Петри

|  |
| --- |
|  |
| 1. – Полная сеть Петри |

1. Вывод

В результате проведенной работы были разработаны локальные одностраничные web-приложения (LSPWA) под управлением фреймворка Vue.js на языке JavaScript в соответствии с указаниями вариантов индивидуального задания.