|  |  |
| --- | --- |
| **Российский университет транспорта (МИИТ)**  **Институт транспортной техники и систем управления**  **Кафедра «Управление и защита информации»** | |
| **Отчет**  **по практическому заданию**  **по теме «Разработка с использованием Vue.js. Задание 7»**  **по дисциплине**  **«Web-программирование»** | |
|  | Выполнил:  Cтудент группы ТКИ-541,  Николаев А.М.  Проверил:  Доцент кафедры УиЗИ к.т.н.,  Сафронов А. И. |
| г. Москва - 2023 | |

ОГЛАВЛЕНИЕ

[Цель работы 3](#_Toc148529429)

[Формулировка задачи 3](#_Toc148529430)

[Спецификация оборудования, на котором выполнялась работа 4](#_Toc148529431)

[Информация об используемом браузере 4](#_Toc148529432)

[Технология подключения фреймворка *Vue.js.* 5](#_Toc148529433)

[Руссифицированные схемы жизненного цикла *Vue.js* 6](#_Toc148529434)

[Таблица соответствия переменных и методов 7](#_Toc148529435)

[ХОД РАБОТЫ 10](#_Toc148529436)

[ВЫВОД 16](#_Toc148529437)

Цель работы

Разработать и продемонстрировать схему тестирования всех жизненных циклов (хуков) фреймворка Vue.js 3.x в формате одностраничного веб-приложения, обеспечив при этом его русификацию и удовлетворяя заданным техническим ограничениям.

Формулировка задачи

Продумать схему тестирования и демонстрации работы всех витков / хуков / методов жизненного цикла фреймворка *Vue.js* в формате одностраничного веб-приложения. Реализовать схему тестирования строго под *Vue.js 3.x* (Приложение 1). Русифицировать схему жизненного цикла *Vue.js 3.x.*

Близкая реализация приложения для тестирования под *Vue.js 2.x*.

Привести информацию об используемом браузере по аналогии с иллюстрациями.

Составить таблицу соответствия переменных и методов, используемых в *web*-приложении.

Заполнить спецификацию оборудования, используемого для выполнения работы.

**Не использовать:** *jQuerry*, «чистый» *JavaScript,* каскадные таблицы стилей (*CSS*), директиву *v-for*, директивы *v-if*, *v-else*, *v-else-if*, *v-model*, теги <*template*>.

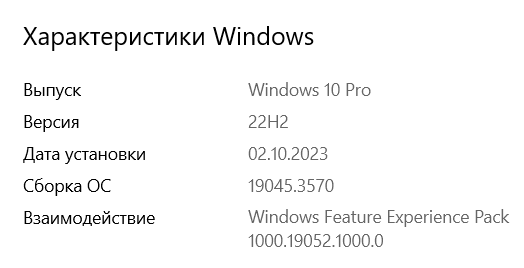
**Реализовать:**

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

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

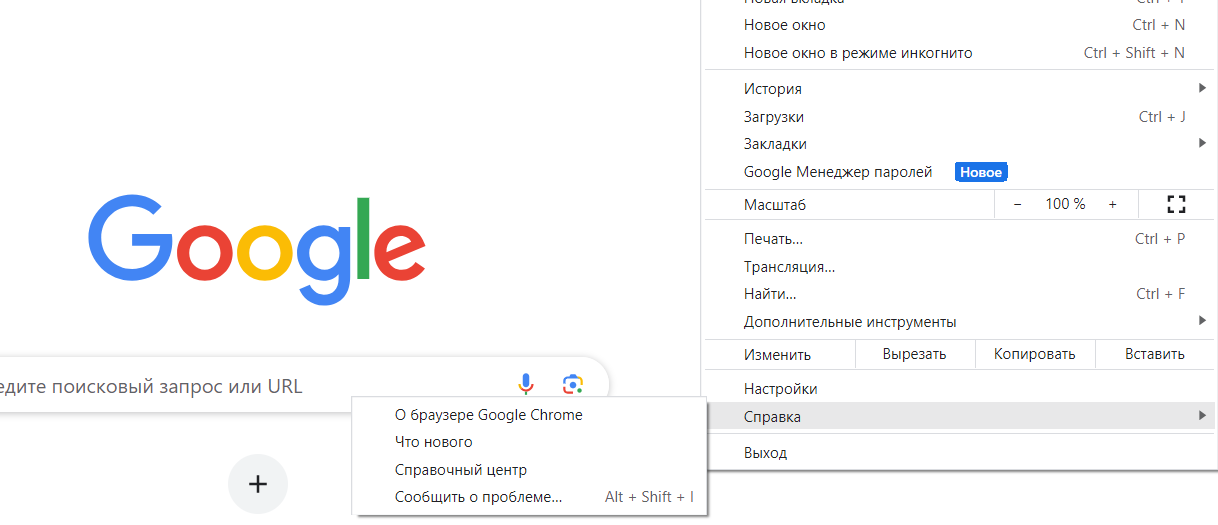
Спецификация оборудования, на котором выполнялась работа

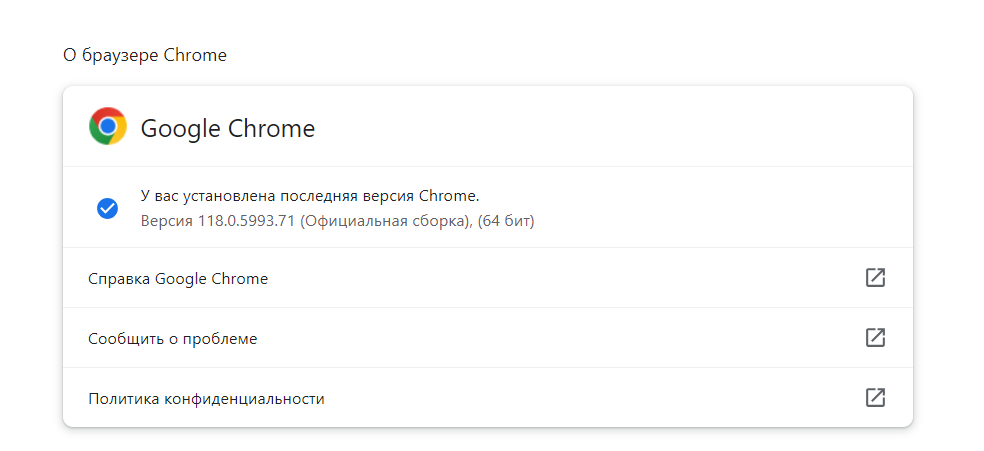




|  |  |  |
| --- | --- | --- |
| **№ п/п** | **Характеристика** | **Сведения** |
| **1** | Тип устройства | Ноутбук |
| **2** | Операционная система | *Microsoft Windows 10* |
| **3** | Процессор | *Intel Core i5* |
| **4** | Оперативная память | 16 Гб |
| **5** | Объём жёсткого диска | 1 Тб |

Информация об используемом браузере





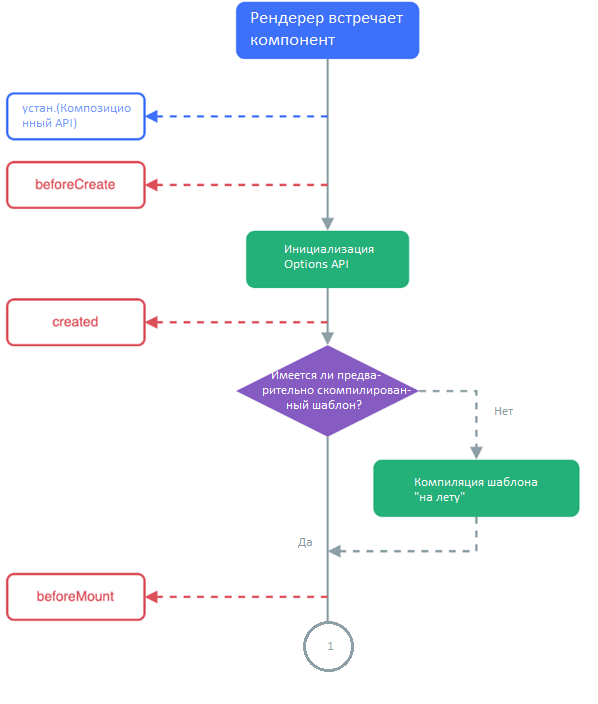
Технология подключения фреймворка *Vue.js.*

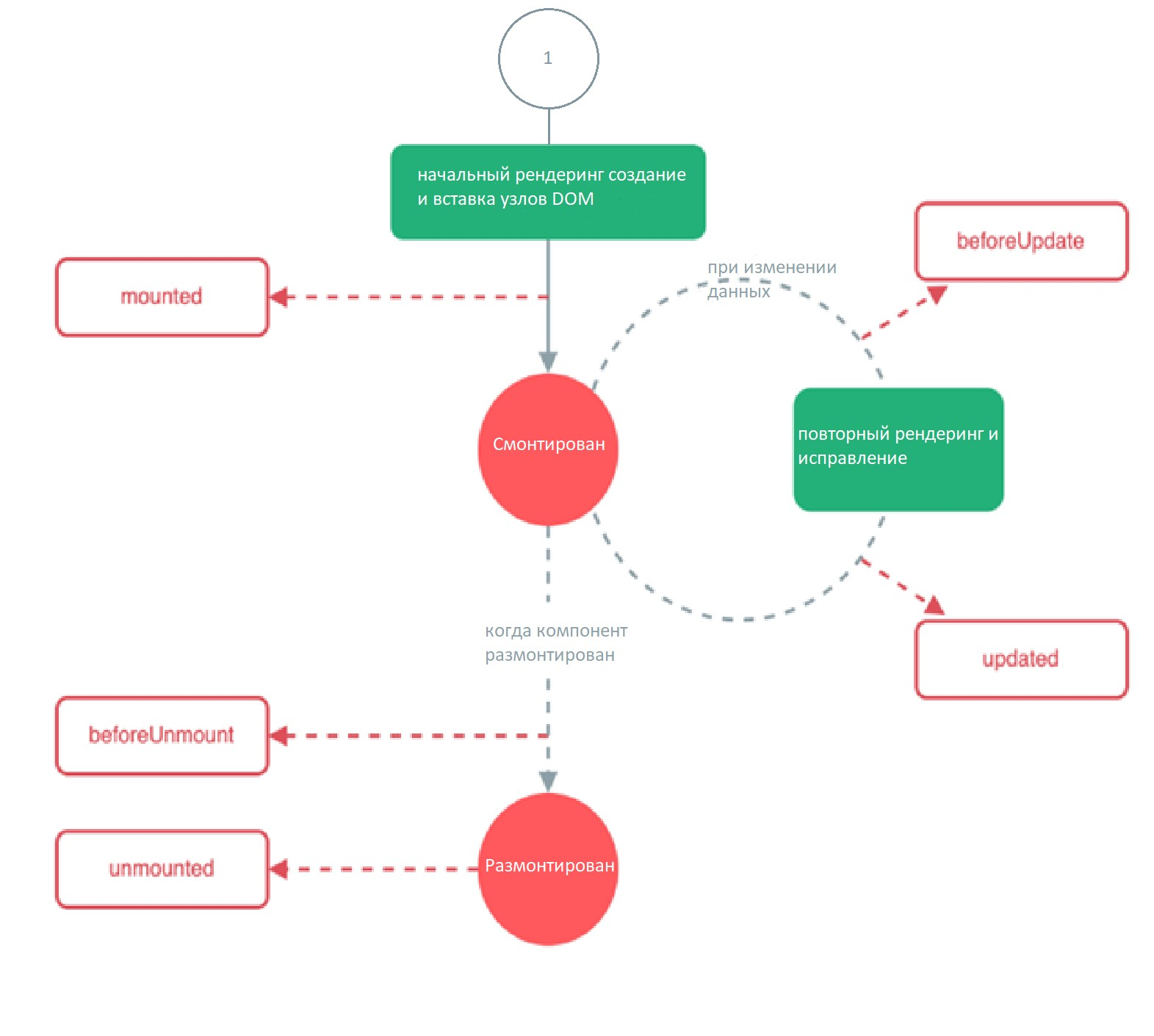
Vue.js можно подключить локально, добавив в ваш HTML следующий тег script:

<script src="https://cdn.jsdelivr.net/npm/vue@3"></script>

Этот тег подключает Vue.js напрямую из CDN, и нет необходимости устанавливать что-либо через Node.js или npm.

Руссифицированные схемы жизненного цикла *Vue.js*





# Таблица соответствия переменных и методов

|  |  |  |
| --- | --- | --- |
| Элемент | Тип | Описание |
| Переменные |  |  |
| message | Переменная | Содержит текстовое сообщение в компоненте LifecycleComponent. Изначально установлено значение "Привет!". |
| events | Переменная | Массив, содержащий записи журнала событий. |
| showComponent | Переменная | Логическая переменная, определяющая, следует ли отображать компонент LifecycleComponent. |
| Методы |  |  |
| changeMessage | Метод | Изменяет значение переменной message на "Привет, Vue!". |
| createComponent | Метод | Устанавливает значение showComponent в true, что приводит к созданию и монтированию компонента LifecycleComponent. |
| destroyComponent | Метод | Устанавливает значение showComponent в false, что приводит к размонтированию компонента LifecycleComponent. |
| addEvent | Метод | Добавляет запись о событии в массив events. |
| Хуки жизненного цикла |  |  |
| beforeMount | Хук | Вызывается перед монтированием компонента. |
| mounted | Хук | Вызывается после монтирования компонента. |
| beforeUpdate | Хук | Вызывается перед обновлением компонента. |
| updated | Хук | Вызывается после обновления компонента. |
| beforeUnmount | Хук | Вызывается перед размонтированием компонента. |
| unmounted | Хук | Вызывается после размонтирования компонента. |

# ХОД РАБОТЫ

Код index\_Nikolaev\_A\_M.html:

<!DOCTYPE html>

<html>

<head>

    <meta charset='utf-8'>

    <title>index\_Nikolaev\_A\_M</title>

    <link rel="stylesheet" href="styles.css">

</head>

<body>

    <div id="appVue">

        <button @click="createComponent">Создать компонент</button>

        <button @click="destroyComponent" v-if="showComponent">Удалить компонент</button>

        <h3>Журнал событий:</h3>

        <div class="event-log" v-for="event in events">{{ event }}</div>

        <lifecycle-component v-if="showComponent" />

    </div>

    <script src="https://cdn.jsdelivr.net/npm/vue@3"></script>

    <script src="script.js"></script>

</body>

</html>

Код script.js:

const LifecycleComponent = {

    data() {

        return {

            message: "Привет!"

        };

    },

    template: '<div @click="changeMessage">{{ message }}</div>',

    beforeMount() {

        this.$parent.addEvent("Компонент готовится к монтированию");

    },

    mounted() {

        this.$parent.addEvent("Компонент был смонтирован");

    },

    beforeUpdate() {

        this.$parent.addEvent("Компонент готовится к обновлению");

    },

    updated() {

        this.$parent.addEvent("Компонент обновлен");

    },

    beforeUnmount() {

        this.$parent.addEvent("Компонент готовится к удалению");

    },

    unmounted() {

        this.$parent.addEvent("Компонент удален");

    },

    methods: {

        changeMessage() {

            this.message = "Привет, Vue!";

        }

    }

};

const App = {

    components: {

        LifecycleComponent

    },

    data() {

        return {

            events: [],

            showComponent: false

        };

    },

    methods: {

        createComponent() {

            this.showComponent = true;

        },

        destroyComponent() {

            this.showComponent = false;

        },

        addEvent(message) {

            const now = new Date();

            this.events.push(`${message} в ${now.toLocaleTimeString()} и ${now.getMilliseconds()}мс`);

        }

    }

};

const app = Vue.createApp(App);

app.mount('#appVue');

Код styles.css:

body {

    font-family: 'Arial', sans-serif;

    background-color: #f3f4f6;

    color: #333;

    padding: 20px;

}

button {

    background-color: #007BFF;

    color: #fff;

    border: none;

    padding: 10px 15px;

    margin: 5px;

    border-radius: 5px;

    cursor: pointer;

    transition: background-color 0.3s;

}

button:hover {

    background-color: #0056b3;

}

.event-log {

    border: 1px solid #ddd;

    padding: 10px;

    margin: 5px 0;

    background-color: #ffffff;

    box-shadow: 0 2px 5px rgba(0,0,0,0.1);

}

|  |
| --- |
|  |
| Рисунок 1 – Вид страницы |

|  |
| --- |
|  |
| Рисунок 2 – Вид страницы |

|  |
| --- |
|  |
| Рисунок 3 – Вид страницы |

|  |
| --- |
|  |
| Рисунок 4 – Вид страницы |

|  |
| --- |
|  |
| Рисунок 5 – Сеть Петри |

ВЫВОД

В ходе выполнения работы было разработано одностраничное веб-приложение на фреймворке Vue.js 3.x, целью которого было демонстрировать работу всех основных жизненных циклов (хуков) этого фреймворка. Приложение успешно продемонстрировало активацию каждого из хуков в различных сценариях использования, начиная от момента создания компонента и заканчивая его удалением.