

Федеральное государственное образовательное бюджетное учреждение
высшего образования
**«Финансовый университет при Правительстве
Российской Федерации»**
(Финансовый университет)

Колледж информатики и программирования

ПМ.08 Разработка кода информационных систем

Группа: ЗИСИП-622

УТВЕРЖДАЮ

Председатель предметно-цикловой комиссии информационных систем и
программирования

_____ Т.Г. Аксёнова

«_____» _____ 2025 г.

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

Преподаватель

_____ Р. Р. Абзалимов

Исполнитель

_____ Л. Д. Слепцов

Оценка: _____

«_____» _____ 2025 г.

Москва

2025

Цель работы

Изучить основные методы работы с Document Object Model (DOM) в JavaScript, научиться манипулировать элементами веб-страницы и обрабатывать события.

Ход работы

Практические задания

1. Создайте кнопку, которая при нажатии меняет цвет фона элемента вывода на случайный цвет. Используйте `Math.random()` для генерации цвета.

Листинг 1 – Смена цвета фона на случайный

```
case 23: // Смена цвета фона на случайный
    const randomColor =
`#${Math.floor(Math.random() * 16777215).toString(16)}`;
    outputDiv.style.backgroundColor = randomColor;
    break;
```

Результаты листинга 1 представлены на рисунке 1.



Рисунок 1 – смена цвета фона на случайный

2. Добавьте кнопку, которая запрашивает у пользователя текст через `prompt` и добавляет его на страницу в виде нового параграфа с рамкой.

Листинг 2 – Запрос текста и добавление параграфа

```
case 24: // Запрос текста и добавление параграфа
    const textInput = prompt("Введите текст для
добавления:");
    if (textInput) {
        const newParagraph =
document.createElement('p');
        newParagraph.innerText = textInput;
        newParagraph.style.border = "1px solid
black";
        outputDiv.appendChild(newParagraph);
    }
```

```
break;
```

Результаты листинга 2 представлены на рисунках 2.1 и 2.2.

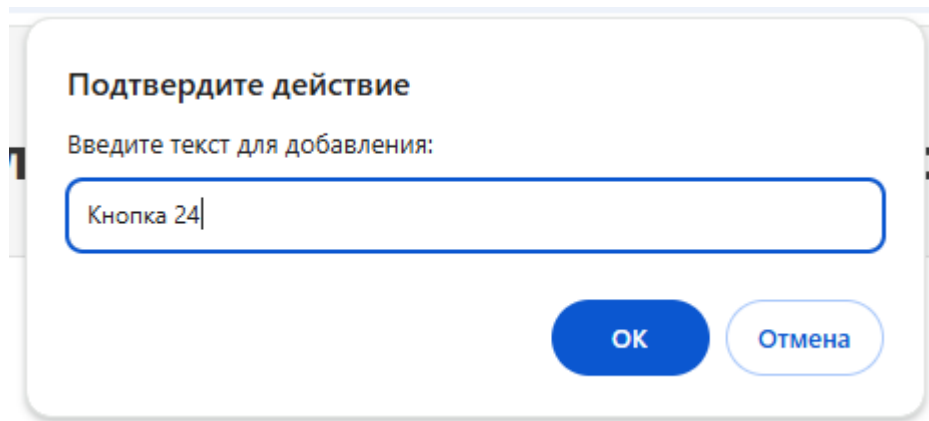


Рисунок 2.1 – запрос текста и добавление параграфа



Рисунок 2.2 – запрос текста и добавление параграфа

3. Реализуйте кнопку, которая подсчитывает и выводит количество всех элементов на странице, используя различные методы выборки элементов.

Листинг 3 – Подсчет всех элементов на странице с помощью `getElementsByTagName`

```
case 25: // Подсчет всех элементов на странице с помощью
getElementsByTagName
        const totalElementsByTagName =
document.getElementsByTagName('*').length;
        alert(`Количество всех элементов на странице с
помощью getElementsByTagName: ${totalElementsByTagName}`);
        break;
```

Результаты листинга 3 представлены на рисунке 3.

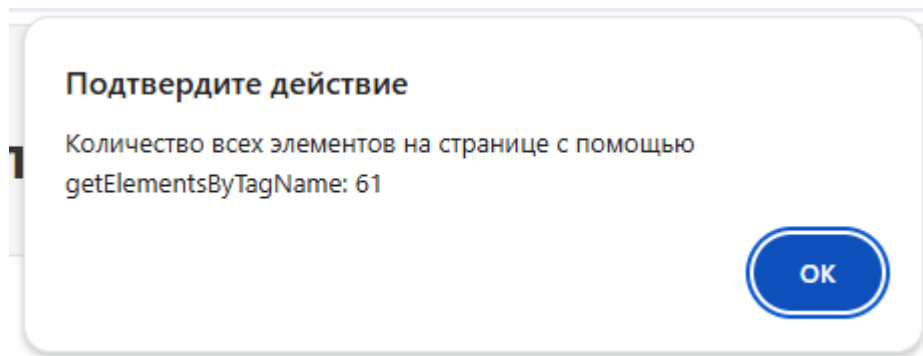


Рисунок 3 – подсчет всех элементов на странице с помощью
getElementsByTagName

Листинг 4 – Подсчет всех элементов на странице с помощью
querySelectorAll

```
case 26: // Подсчет всех элементов на странице с помощью  
querySelectorAll  
        const totalElementsByQuerySelectorAll =  
document.querySelectorAll('*').length;  
        alert(`Количество всех элементов на странице с  
помощью querySelectorAll: ${totalElementsByQuerySelectorAll}`);  
        break;
```

Результаты листинга 4 представлены на рисунке 4.

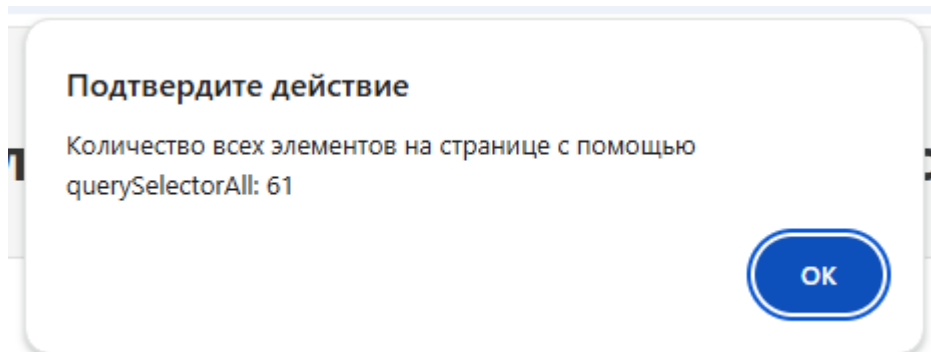


Рисунок 4 – подсчет всех элементов на странице с помощью querySelectorAll

Индивидуальное задание

Листинг 5 – Показать текущее время

```
case 27: // Показать текущее время  
        alert(`Текущее время: ${new  
Date().toLocaleTimeString()}`);  
        break;
```

Результаты листинга 5 представлены на рисунке 5.

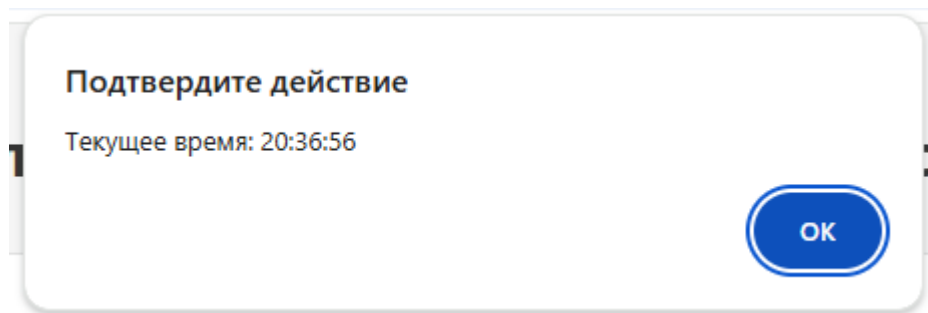


Рисунок 5 – показать текущее время

Листинг 6 – Изменить цвет текста на случайный

```
case 28: // Изменить цвет текста на случайный
    const randomTextColor = `#${Math.floor(Math.random() *
16777215).toString(16)}`;
    outputDiv.style.color = randomTextColor;
    break;
```

Результаты листинга 6 представлены на рисунке 6.

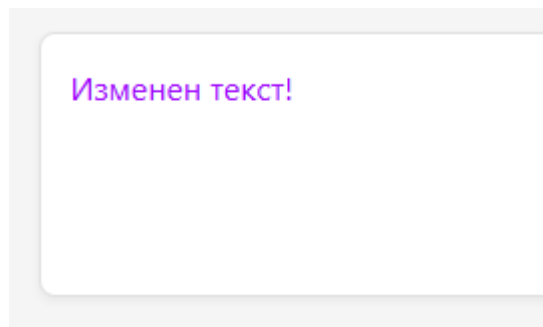


Рисунок 6 – изменить цвет текста на случайный

Листинг 7 – Добавление изображения

```
case 29: // Добавление изображения
    const imgSrc = prompt("Введите URL изображения:");
    if (imgSrc) {
        const img = document.createElement('img');
        img.src = imgSrc;
        img.style.maxWidth = '100%'; // Максимальная ширина
для адаптивности
        outputDiv.appendChild(img);
    }
    break;
```

Результаты листинга 7 представлены на рисунках 7.1 и 7.2.

Подтвердите действие

Введите URL изображения:

3484d8e701e5b0047ca27897de90539d4-5236271-images-thumbs&n=13

OK Отмена

Рисунок 7.1 – добавление изображения

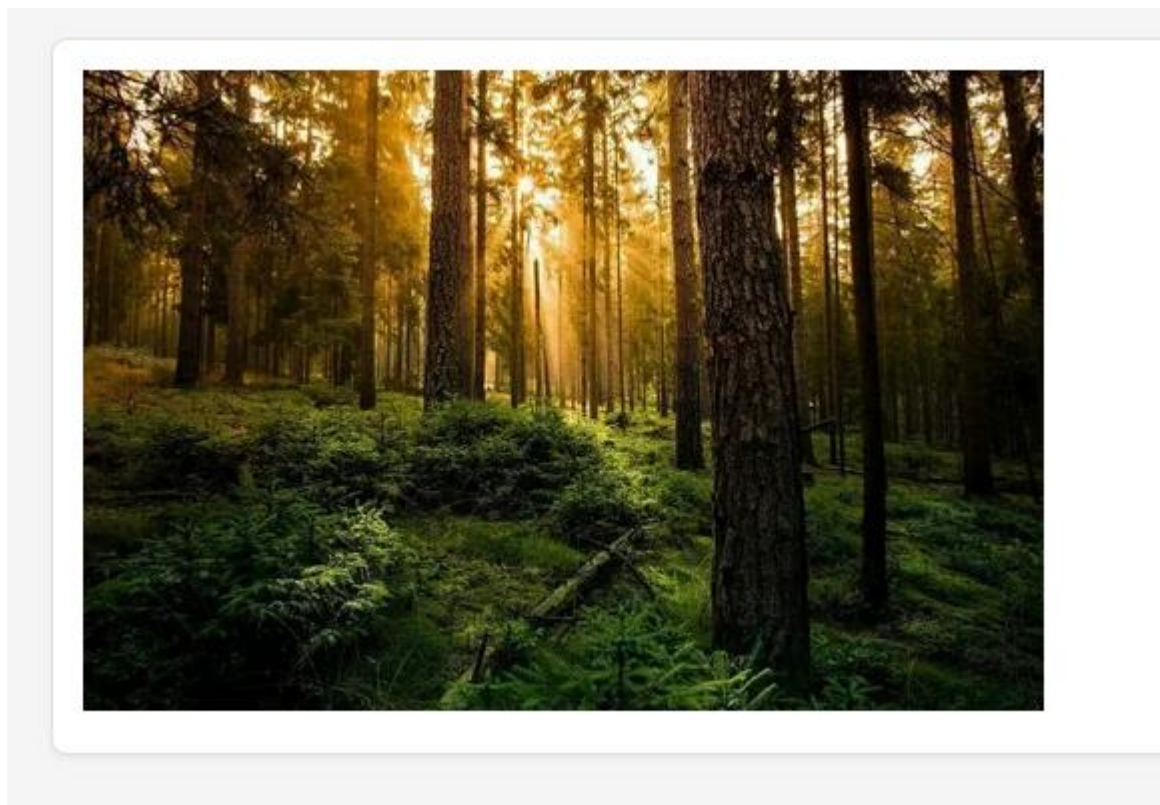


Рисунок 7.2 – добавление изображения

Листинг 8 – Добавить кнопку

```
case 30: // Добавить кнопку
    const button = document.createElement('button');
    button.innerText = "Новая кнопка";
    outputDiv.appendChild(button);
    break;
```

Результаты листинга 8 представлены на рисунке 8.

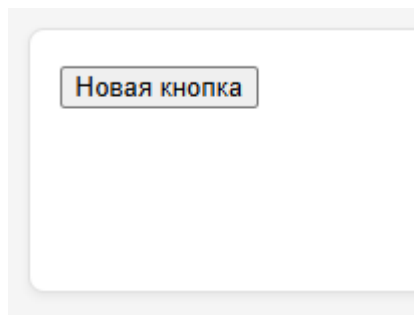


Рисунок 8 – добавить кнопку

Листинг 9 – Отобразить список доступных функций

```
case 31: // Отобразить список доступных функций
    alert("Доступные функции:\n" +
        "27: Показать текущее время\n" +
        "28: Изменить цвет текста на случайный\n" +
        "29: Добавление изображения\n" +
        "30: Добавить кнопку\n" +
        "31: Отобразить список доступных функций\n" +
        "32: Изменение размера текста\n" +
        "33: Добавить список\n" +
        "34: Добавление ссылки\n" +
        "35: Смена шрифта\n" +
        "36: Увеличение отступов\n" +
        "37: Смена стиля границы\n" +
        "38: Смена карусели изображений\n" +
        "39: Смена прозрачности\n" +
        "40: Переключение видимости\n" +
        "41: Увеличить масштаб содержимого\n" +
        "42: Открытие нового окна\n" +
        "43: Показать текущую дату\n" +
        "44: Копирование текста в буфер обмена\n" +
        "45: Показать случайный элемент из массива\n" +
        "46: Изменение прозрачности фона\n" +
        "47: Сохранение текста в локальное хранилище\n" +
        "48: Заполнение outputDiv сохраненным текстом\n" +
        "49: Создание таблицы для переключения между
        контентом\n" +
        "50: Смена цвета фона на выбранный\n");
    break;
```

Результаты листинга 9 представлены на рисунке 9.

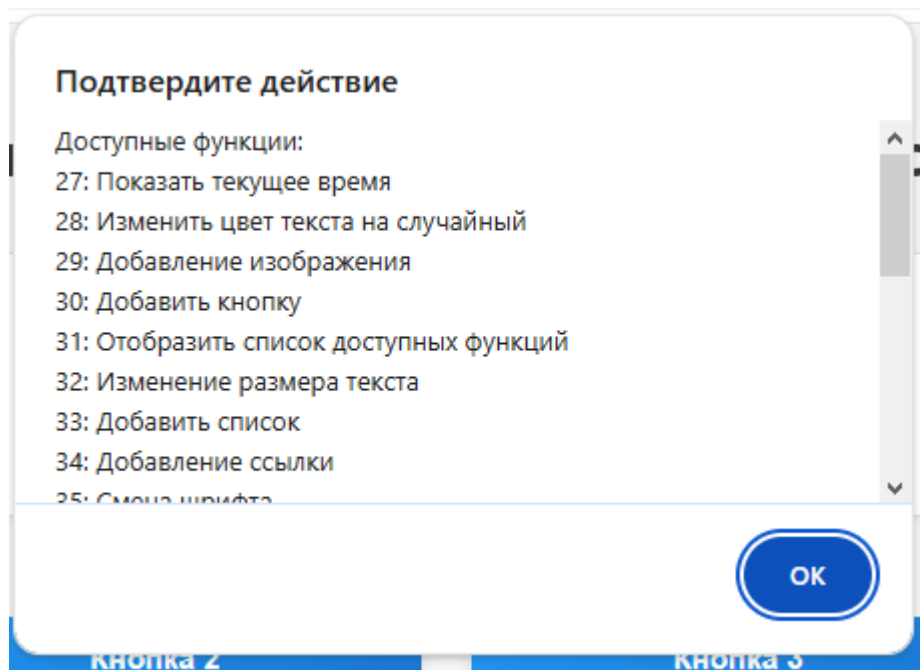


Рисунок 9 – отобразить список доступных функций

Листинг 10 – Изменение размера текста

```
case 32: // Изменение размера текста
    const newSize = prompt("Введите новый размер текста
(например, 20px):");
    outputDiv.style.fontSize = newSize;
    break;
```

Результаты листинга 10 представлены на рисунках 10.1 и 10.2.

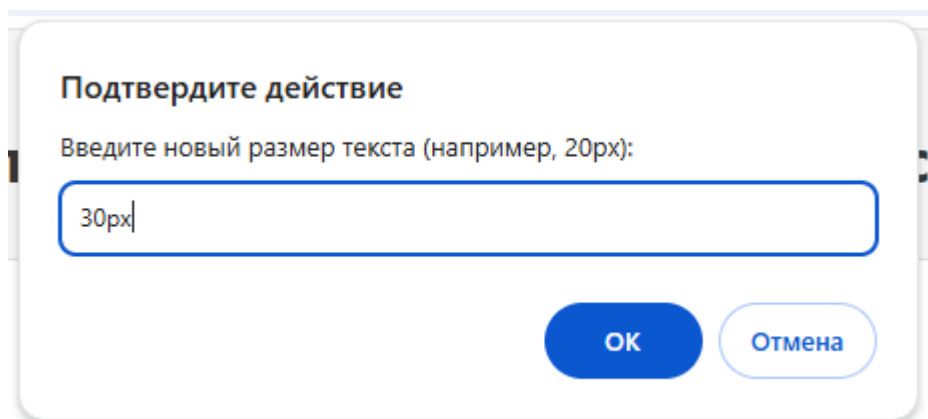


Рисунок 10.1 – изменение размера текста

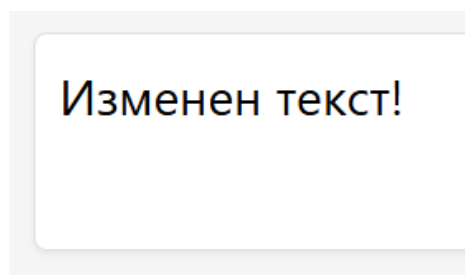


Рисунок 10.2 – изменение размера текста

Листинг 11 – Добавить список

```
case 33: // Добавить список
    const listItems = prompt("Введите элементы списка,
разделенные запятой:");
    if (listItems) {
        const ul = document.createElement('ul');
        listItems.split(',').forEach(item => {
            const li = document.createElement('li');
            li.innerText = item.trim();
            ul.appendChild(li);
        });
        outputDiv.appendChild(ul);
    }
    break;
```

Результаты листинга 11 представлены на рисунках 11.1 и 11.2.

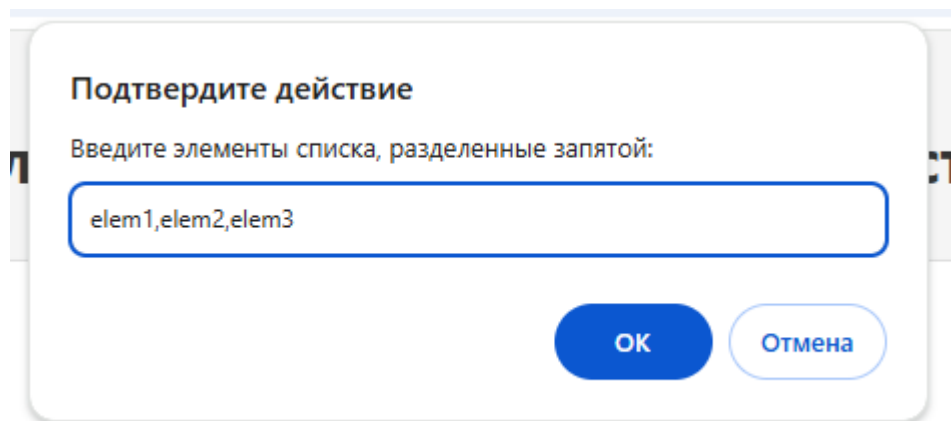


Рисунок 11.1 – добавить список

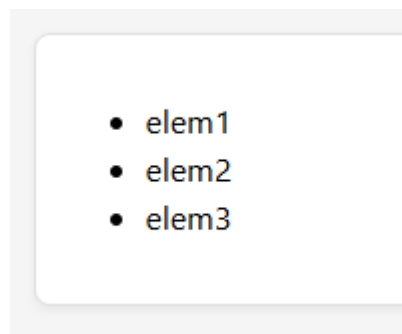


Рисунок 11.2 – добавить список

Листинг 12 – Добавление ссылки

```
case 34: // Добавление ссылки
    const linkText = prompt("Введите текст ссылки:");
    const linkUrl = prompt("Введите URL:");
    if (linkText && linkUrl) {
        const a = document.createElement('a');
        a.href = linkUrl;
        a.innerText = linkText;
```

```
outputDiv.appendChild(a);  
}  
break;
```

Результаты листинга 12 представлены на рисунках 12.1, 12.2 и 12.3.

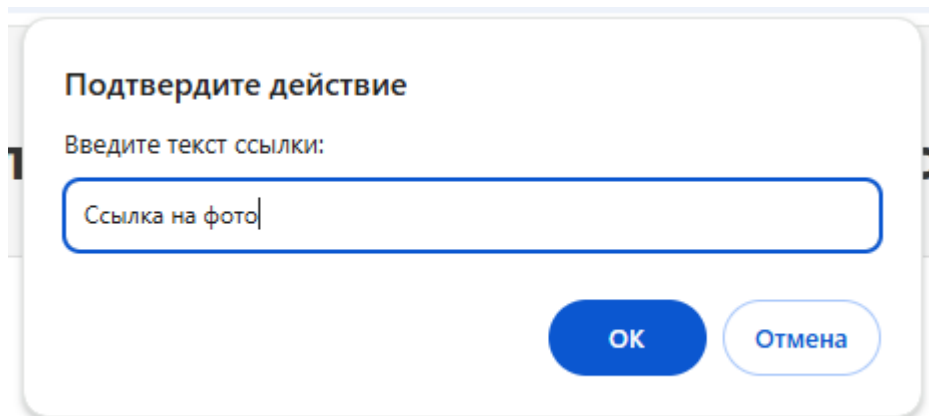


Рисунок 12.1 – добавление ссылки

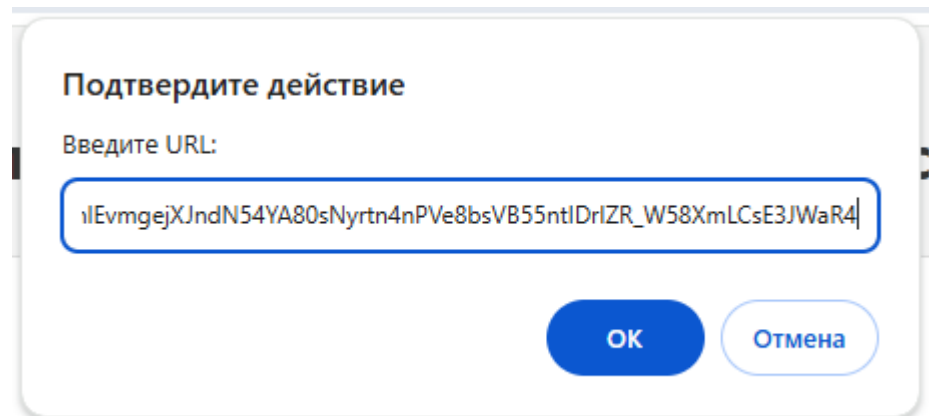


Рисунок 12.2 – добавление ссылки

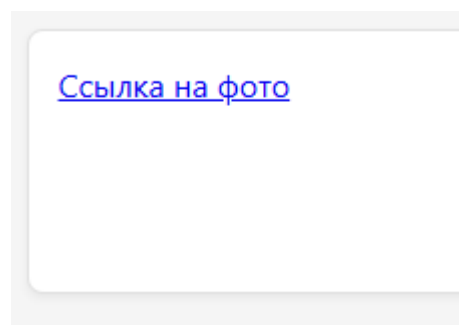


Рисунок 12.3 – добавление ссылки

Листинг 13 – Смена шрифта

```
case 35: // Смена шрифта  
    const newFont = prompt("Введите название шрифта:");  
    outputDiv.style.fontFamily = newFont;  
    break;
```

Результаты листинга 13 представлены на рисунках 13.1 и 13.2.

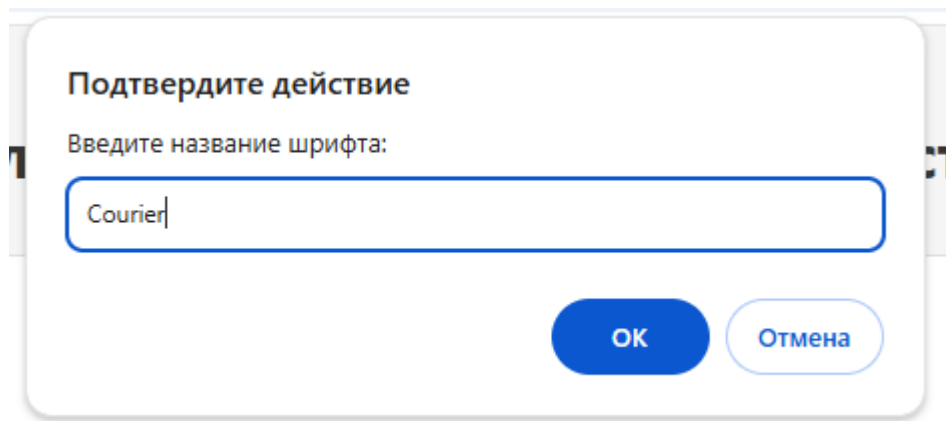


Рисунок 13.1 – смена шрифта

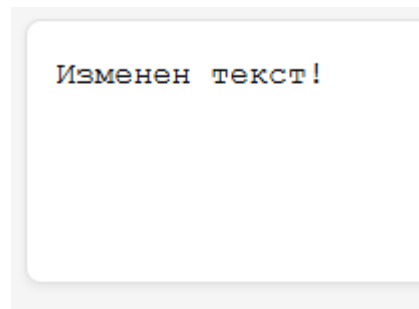


Рисунок 13.2 – смена шрифта

Листинг 14 – Увеличение отступов

```
case 36: // Увеличение отступов
    const padding = prompt("Введите размер отступов
(например, 20px):");
    outputDiv.style.padding = padding;
    break;
```

Результаты листинга 14 представлены на рисунках 14.1 и 14.2.

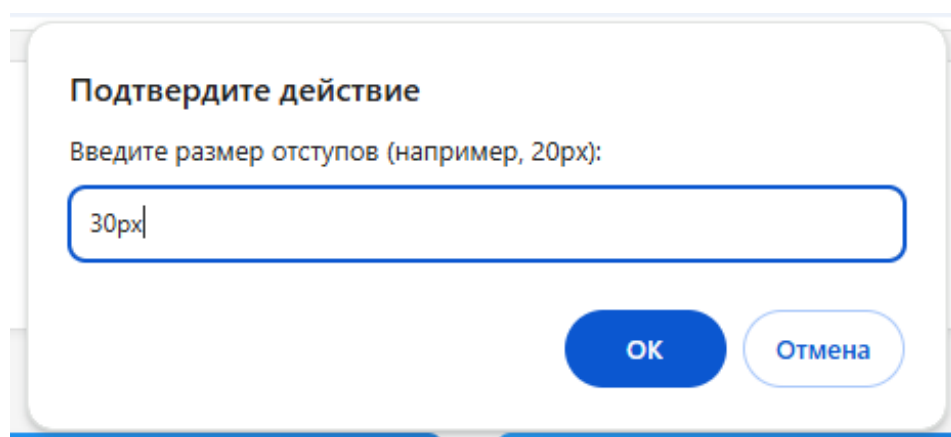


Рисунок 14.1 – увеличение отступов

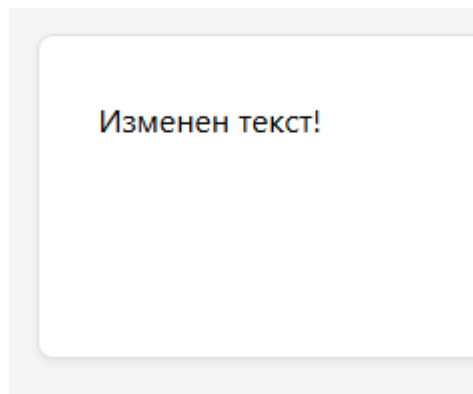


Рисунок 14.2 – увеличение отступов

Листинг 15 – Смена стиля границы

```
case 37: // Смена стиля границы
    const borderStyle = prompt("Введите стиль границы
(например, solid):");
    const borderColor = prompt("Введите цвет границы:");
    outputDiv.style.border = `2px ${borderStyle}
${borderColor}`;
    break;
```

Результаты листинга 15 представлены на рисунках 15.1, 15.2 и 15.3.

Рисунок 15.1 – смена стиля границы

Рисунок 15.2 – смена стиля границы

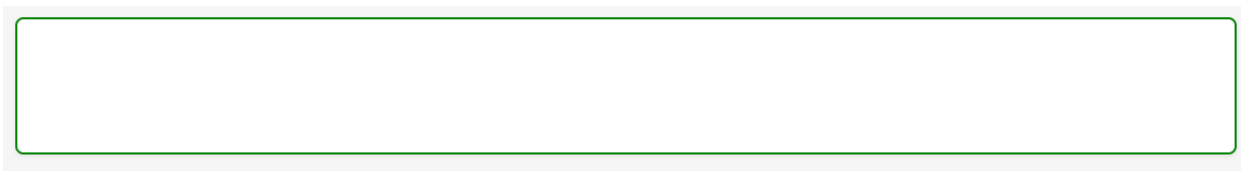


Рисунок 15.3 – смена стиля границы

Листинг 16 – Смена карусели изображений

```
case 38: // Смена карусели изображений
    const images = prompt("Введите URL изображений,
разделенные запятой:");
    if (images) {
        const imgArray = images.split(',');
        let i = 0;
        const carousel = setInterval(() => {
            outputDiv.innerHTML = `})
```

Результаты листинга 16 представлены на рисунках 16.1, 16.2, 16.3 и 16.4.

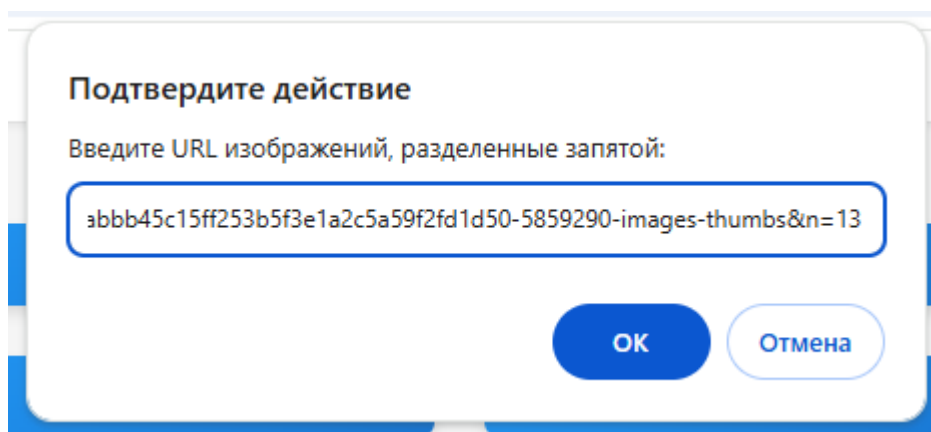


Рисунок 16.1 – смена карусели изображений



Рисунок 16.2 – смена карусели изображений



Рисунок 16.3 – смена карусели изображений



Рисунок 16.4 – смена карусели изображений

Листинг 17 – Смена прозрачности

```
case 39: // Смена прозрачности
    const opacity = prompt("Введите уровень прозрачности (от
0 до 1):");
    outputDiv.style.opacity = opacity;
    break;
```

Результаты листинга 17 представлены на рисунках 17.1 и 17.2.

A dialog box titled "Подтвердите действие" (Confirm action) with a prompt "Введите уровень прозрачности (от 0 до 1):" (Enter the transparency level (from 0 to 1):). The input field contains "0.4". There are "OK" and "Отмена" (Cancel) buttons.

Рисунок 17.1 – смена прозрачности

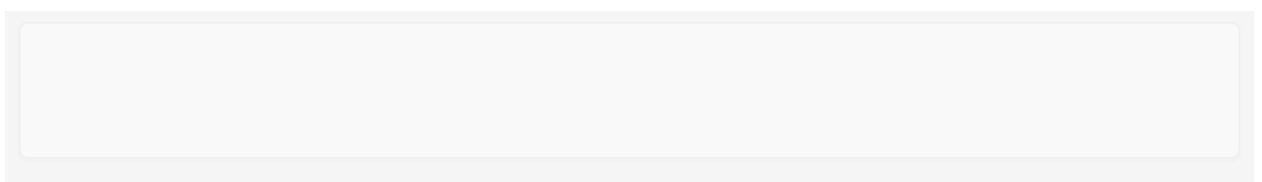


Рисунок 17.2 – смена прозрачности

Листинг 18 – Переключение видимости

```
case 40: // Переключение видимости
        outputDiv.style.display = outputDiv.style.display ===
'none' ? 'block' : 'none';
        break;
```

Результаты листинга 18 представлены на рисунке 18.

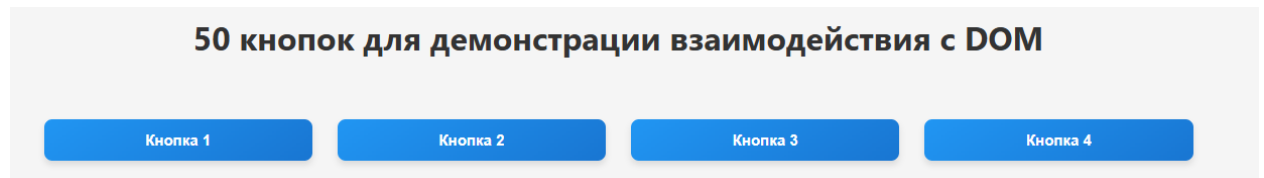


Рисунок 18 – переключение видимости

Листинг 19 – Увеличить масштаб содержимого

```
case 41: // Увеличить масштаб содержимого
        const zoomLevel = prompt("Введите уровень зума (например,
1.5):");
        outputDiv.style.transform = `scale(${zoomLevel})`;
        break;
```

Результаты листинга 19 представлены на рисунках 19.1 и 19.2.

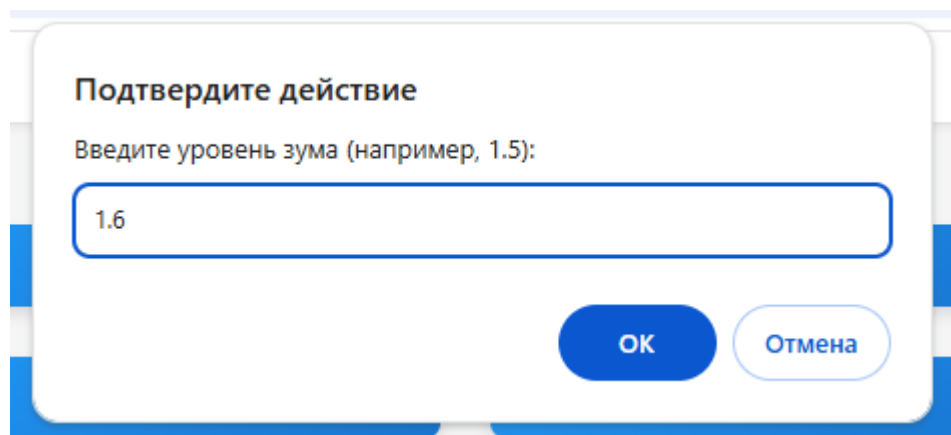


Рисунок 19.1 – увеличить масштаб содержимого

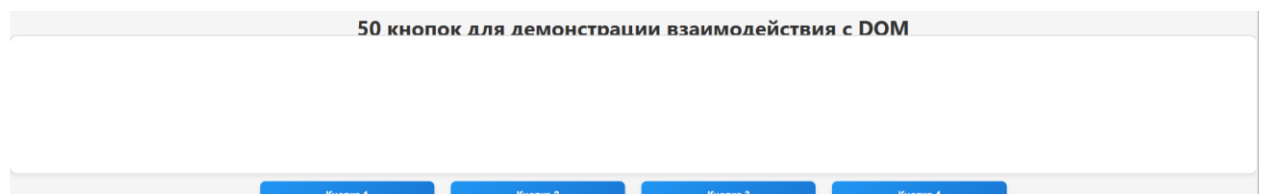


Рисунок 19.2 – увеличить масштаб содержимого

Листинг 20 – Открытие нового окна

```
case 42: // Открытие нового окна
        window.open('https://example.com');
        break;
```

Результаты листинга 20 представлены на рисунке 20.

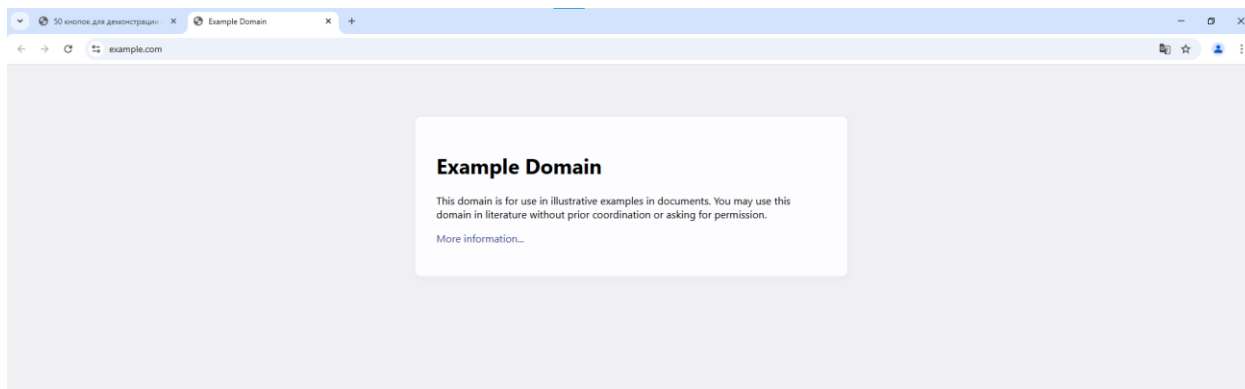


Рисунок 20 – открытие нового окна

Листинг 21 – Показать текущую дату

```
case 43: // Показать текущую дату
    alert(`Текущая дата: ${new
Date().toLocaleDateString()}`);
    break;
```

Результаты листинга 21 представлены на рисунке 21.

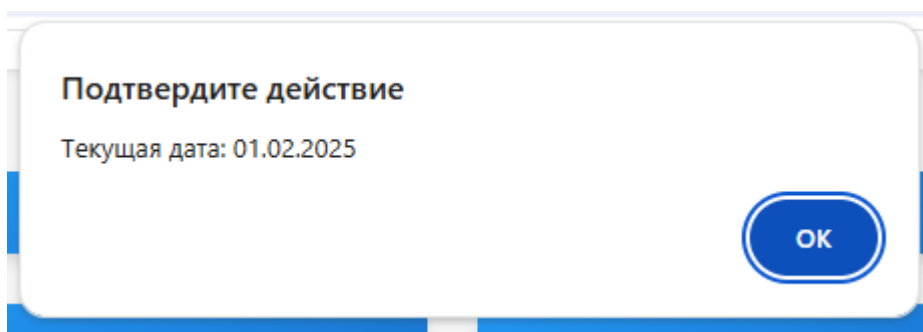


Рисунок 21 – показать текущую дату

Листинг 22 – Копирование текста в буфер обмена

```
case 44: // Копирование текста в буфер обмена
    const copyText = prompt("Введите текст для
копирования:");
    navigator.clipboard.writeText(copyText).then(() => {
        alert('Текст скопирован в буфер обмена!');
    }).catch(err => {
        console.error('Ошибка при копировании текста:', err);
    });
    break;
```

Результаты листинга 22 представлены на рисунках 22.1 и 22.2.

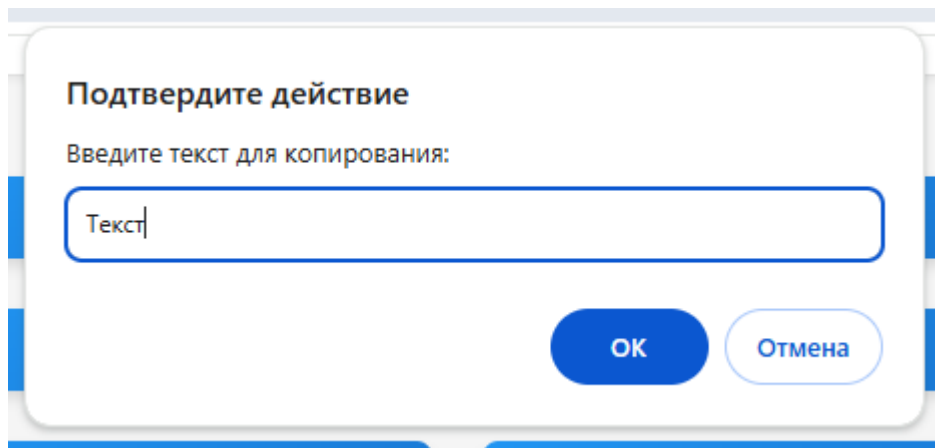


Рисунок 22.1 – копирование текста в буфер обмена

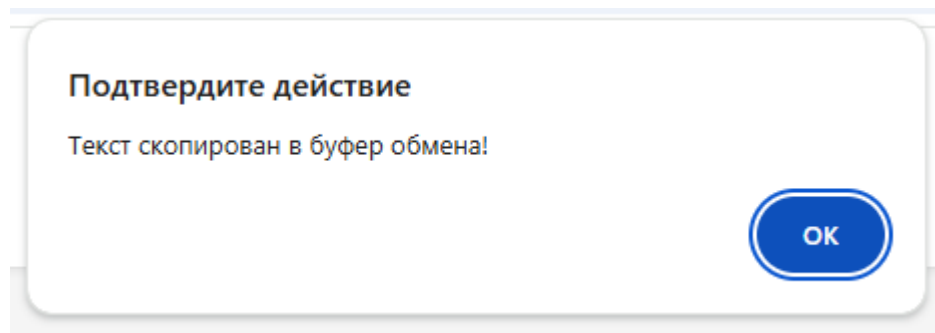


Рисунок 22.2 – копирование текста в буфер обмена

Листинг 23 – Показать случайный элемент из массива

```
case 45: // Показать случайный элемент из массива
    const randomItems = ['Элемент 1', 'Элемент 2', 'Элемент 3', 'Элемент 4'];
    const randomItem = randomItems[Math.floor(Math.random() * randomItems.length)];
    alert(`Случайный элемент: ${randomItem}`);
    break;
```

Результаты листинга 23 представлены на рисунке 23.

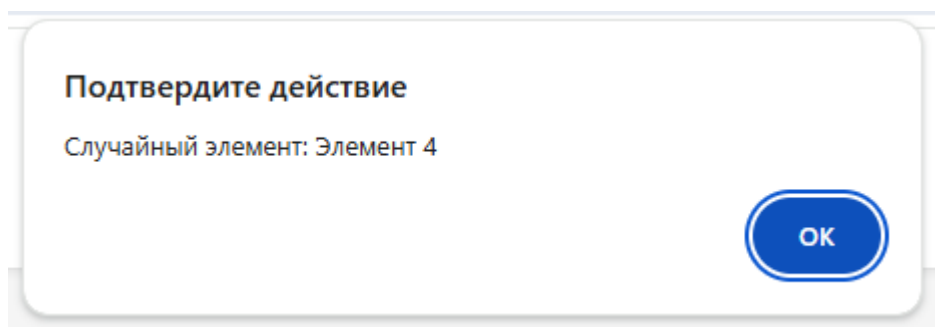


Рисунок 23 – показать случайный элемент из массива

Листинг 24 – Изменение прозрачности фона

```
case 46: // Изменение прозрачности фона
    const backgroundOpacity = prompt("Введите уровень прозрачности фона (от 0 до 1):");
```

```
outputDiv.style.backgroundColor = `rgba(255, 255, 255,
${backgroundOpacity})`;
break;
```

Результаты листинга 24 представлены на рисунках 24.1 и 24.2.

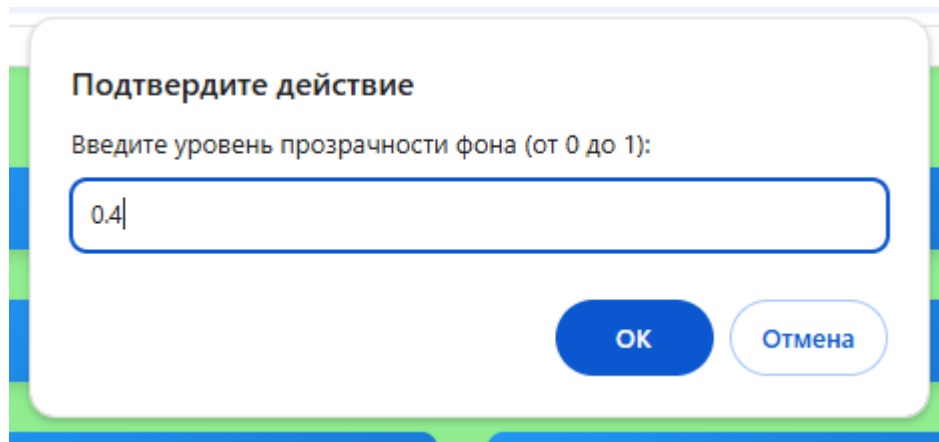


Рисунок 24.1 – изменение прозрачности фона

.. .. .

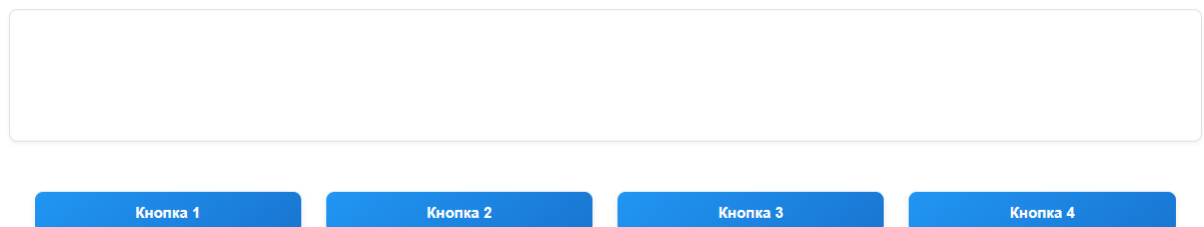


Рисунок 24.2 – изменение прозрачности фона

Листинг 25 – Сохранение текста в локальное хранилище

```
case 47: // Сохранение текста в локальное хранилище
    const saveText = prompt("Введите текст для сохранения в
    локальное хранилище:");
    if (saveText) {
        localStorage.setItem('savedText', saveText);
        alert('Текст сохранен в локальном хранилище.');
```

Результаты листинга 25 представлены на рисунках 25.1 и 25.2.

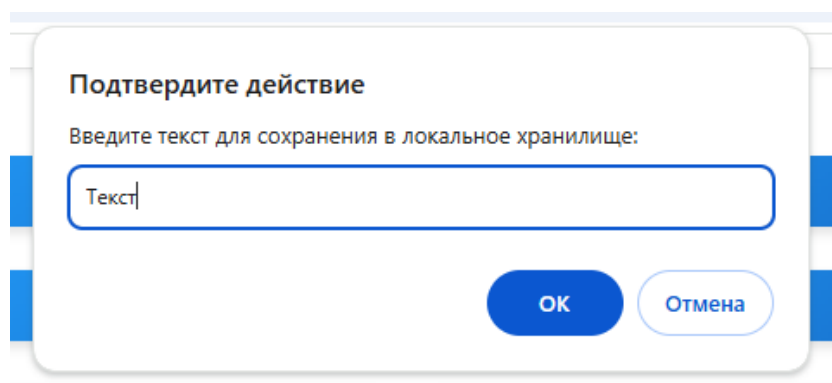


Рисунок 25.1 – сохранение текста в локальное хранилище

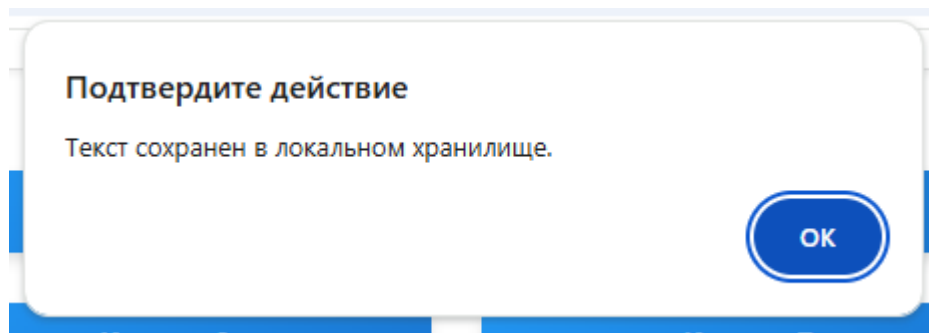


Рисунок 25.2 – сохранение текста в локальное хранилище

Листинг 26 – Заполнение outputDiv сохраненным текстом

```
case 48: // Заполнение outputDiv сохраненным текстом
    const loadedText = localStorage.getItem('savedText');
    if (loadedText) {
        outputDiv.innerText = loadedText;
    } else {
        alert('Нет сохраненного текста в локальном
хранилище.');
```

Результаты листинга 26 представлены на рисунке 26.

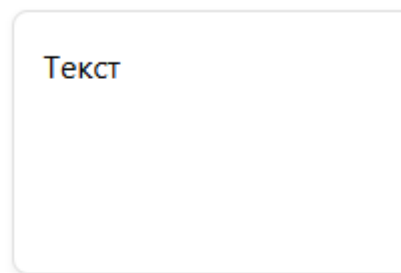


Рисунок 26 – заполнение outputDiv сохраненным текстом

Листинг 27 – Создание табуляции для переключения между контентом

```
case 49: // Создание табуляции для переключения между контентом
    const tabs = ['Контент 1', 'Контент 2', 'Контент 3'];
    tabs.forEach((tab, index) => {
        const tabButton = document.createElement('button');
        tabButton.innerText = tab;
        tabButton.onclick = () => {
            outputDiv.innerText = `Вы выбрали: ${tab}`;
        };
        outputDiv.appendChild(tabButton);
    });
    break;
```

Результаты листинга 27 представлены на рисунках 27.1 и 27.2.

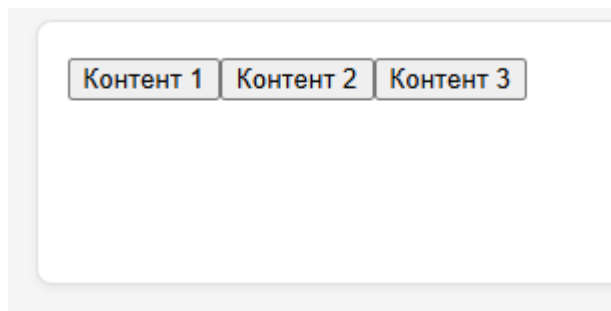


Рисунок 27.1 – создание табуляции для переключения между контентом

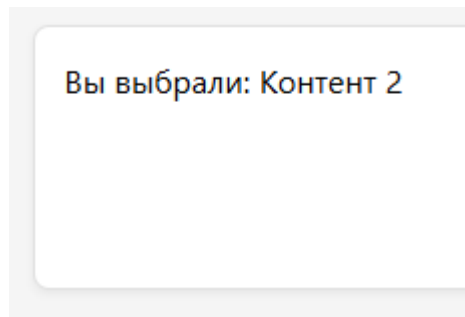


Рисунок 27.2 – создание табуляции для переключения между контентом

Листинг 28 – Смена цвета фона на выбранный

```
case 50:
    const selectedColor = prompt("Введите цвет (например,
    'red', '#00ff00', 'rgb(0, 0, 255)'):");
    if (selectedColor) {
        outputDiv.style.backgroundColor = selectedColor;
        alert(`Фон поля вывода изменен на ${selectedColor}`);
    } else {
        alert('Цвет не был задан.');
```

Результаты листинга 28 представлены на рисунках 28.1, 28.2 и 28.3.

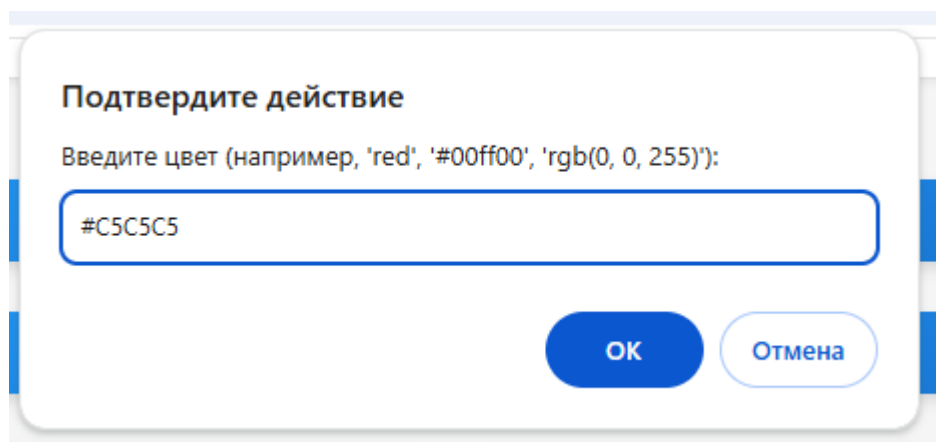


Рисунок 28.1 – смена цвета фона на выбранный

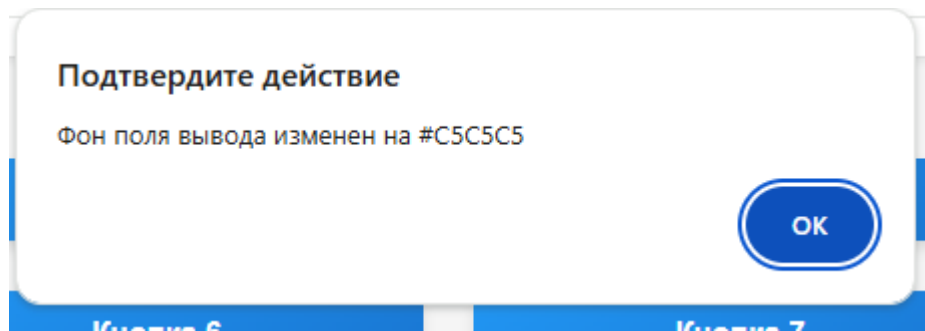


Рисунок 28.2 – смена цвета фона на выбранный

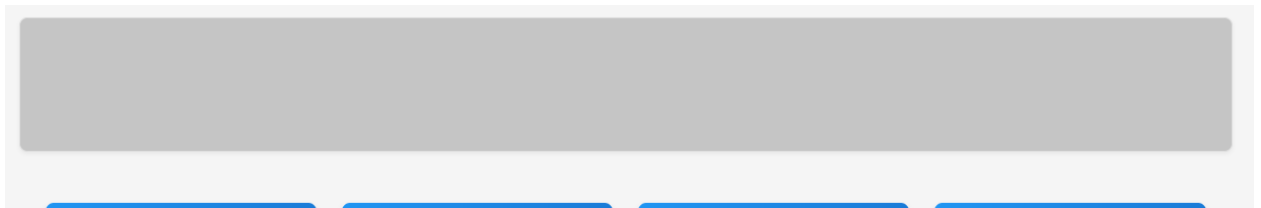


Рисунок 28.3 – смена цвета фона на выбранный

Контрольные вопросы

1. Что такое DOM и какова его роль в веб-разработке?

DOM (Document Object Model) — это программный интерфейс для HTML и XML документов. Он представляет структуру документа в виде дерева объектов, где каждый узел — это элемент, атрибут или текст.

Роль DOM:

- Позволяет JavaScript взаимодействовать с содержимым, структурой и стилями веб-страницы.
 - Дает возможность динамически изменять страницу без её перезагрузки.
2. Перечислите основные методы получения элементов в DOM. В чём их различия?
 - `getElementById(id)` — возвращает элемент по его уникальному `id`.
 - `getElementsByClassName(className)` — возвращает коллекцию элементов с указанным классом.

- `getElementsByTagName(tagName)` — возвращает коллекцию элементов по имени тега.
- `querySelector(selector)` — возвращает первый элемент, соответствующий CSS-селектору.
- `querySelectorAll(selector)` — возвращает все элементы, соответствующие CSS-селектору, в виде `NodeList`.

Различия:

- `getElementById` работает только с `id`.
- `getElementsByClassName` и `getElementsByTagName` возвращают "живые" коллекции (обновляются при изменении DOM).
- `querySelector` и `querySelectorAll` более гибкие, так как работают с любыми CSS-селекторами.

3. Какие существуют способы изменения стилей элементов через JavaScript?

Стили элементов можно изменять:

- Прямо через свойство, которое управляет стилями.
- Добавляя или удаляя классы, которые содержат стили.
- Изменяя атрибут, который хранит стили.

4. Объясните разницу между методами `querySelector()` и `querySelectorAll()`.

- `querySelector()` возвращает первый элемент, который соответствует указанному селектору.
- `querySelectorAll()` возвращает все элементы, которые соответствуют селектору, в виде списка.

5. Как создать новый элемент и добавить его на страницу с помощью JavaScript?

Чтобы добавить новый элемент на страницу, нужно:

- Создать элемент `.createElement()`.

- Настроить его (например, добавить текст или атрибуты).
- Вставить его в нужное место на странице, используя методы для работы с DOM `.appendChild(newElement)`.

6. Что такое события в JavaScript и как их обрабатывать?

События — это действия, которые происходят на странице (например, клик, наведение курсора, ввод текста). Обработка событий позволяет выполнять определённый код в ответ на эти действия. Это можно сделать:

- Через атрибуты в HTML `onclick`.
- Через свойства элементов в JavaScript `.onclick`.
- Через специальный метод, который добавляет обработчик события `.addEventListener`.

7. В чём разница между методами ``getElementsByClassName()`` и ``getElementsByTagName()``?

- `getElementsByClassName()` возвращает элементы по имени класса.
- `getElementsByTagName()` возвращает элементы по имени тега.

Оба метода возвращают "живые" коллекции, которые обновляются при изменении DOM.

8. Как удалить элемент из DOM?

Чтобы удалить элемент, нужно:

- Найти элемент, который нужно удалить.
- Использовать метод для его удаления либо напрямую `.remove()`, либо через его родительский элемент `.removeChild(element)`.

9. Объясните назначение метода ``addEventListener()``. Какие параметры он принимает?

`addEventListener()` позволяет добавить обработчик события к элементу.

Параметры:

- Тип события (например, 'click', 'mouseover').
- Функция-обработчик, которая выполняется при возникновении события.
- Опции (необязательно): например, `{ once: true }` для одноразового выполнения.

10. Что такое всплытие событий (event bubbling) в DOM?

Всплытие — это механизм, при котором событие, произошедшее на вложенном элементе, сначала обрабатывается на нём, а затем поднимается вверх по дереву DOM, вызывая обработчики на родительских элементах. Это позволяет обрабатывать события на разных уровнях вложенности.