МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

(КАФЕДРА 43)

ОЦЕНКА

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Н.А. Соловьева
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О КОНТР 1

«Основы JavaScript»

по дисциплине: Web-технологии

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4236		Л. Мвале
номер группы	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

Решение задач по JavaScript и веб-разработке

Вариант Задание 29

веб-технологии. Контрольная_1. вариант 29
Напишите сценарий на javascript с примером использования метода getElementsByTagName() для выбора нескольких (не менее 3) однотипных элементов на веб-странице. С каждым выбранным элементом нужно выполнить какое-либо простое действие. В ответе напишите сформулированную Вами задачу, код сценария, код страницы и скриншоты.
Объясните назначение и дайте пример использования свойства children
Перечислите все способы (теги, атрибуты, css), с помощью которых можно добавить графический файл на веб-страницу. Дайте краткий комментарий.
Опишите ситуацию, в которой можно применить селектор с атрибутом (селектор [атрибут]). Дайте пример
Для решения задачи написать функцию на javascript. Входные данные функции: размерность матрицы и она сама. Выходные данные: матрица или результат по заданию. Задание: Найти в матрице сумму элементов, меньших Z.

1. Скрипт JavaScript с использованием

`getElementsByTagName()`

Постановка задачи

Написать скрипт на JavaScript, который выбирает минимум три элемента <p> на веб-странице и изменяет каждый из них, меняя цвет текста и добавляя рамку.

Решение

HTML-код (index.html)

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Пример getElementsByTagName</title>
```

```
    <style>
        p { font-size: 18px; }
    </style>
</head>
<body>
    <h1>Пример использования getElementsByTagName</h1>
    <p>Первый параграф.</p>
    <p>Второй параграф.</p>
    <p>Третий параграф.</p>

    <script src="script.js"></script>
</body>
</html>
```

Пояснение:

- Стандартная HTML-структура с тремя элементами <p>.
- Подключен внешний JavaScript-файл script.js.

JavaScript-код (script.js)

```
// Выбираем все элементы <p>
const paragraphs = document.getElementsByTagName('p');

// Перебираем каждый элемент и изменяем его стиль
for (let i = 0; i < paragraphs.length; i++) {
```

```
    paragraphs[i].style.color = "blue";
    paragraphs[i].style.border = "2px solid red";
    paragraphs[i].style.padding = "10px";
}
```

Пояснение:

1. getElementsByTagName('p') возвращает коллекцию всех элементов <p>.

2. Цикл for проходит по каждому элементу и применяет стили:

- color: blue — синий текст.

- border: 2px solid red — красная рамка.

- padding: 10px — отступы для лучшего вида.

Теоретическое объяснение:

Метод `getElementsByTagName()` - это DOM-метод, который возвращает живую (live) коллекцию всех элементов с указанным именем тега. Особенности:

1) Возвращает HTMLCollection (не массив)

2) Коллекция "живая" - автоматически обновляется при изменении DOM

3) Чувствителен к регистру в XHTML-документах

4) Доступен для любого элемента DOM, не только document

Применяемые стили:

- `color: blue` - изменяет цвет текста на синий (HEX #0000FF)

- `border: 2px solid red` - добавляет сплошную красную рамку толщиной 2px

- `padding: 10px` - создает внутренние отступы для лучшей читаемости

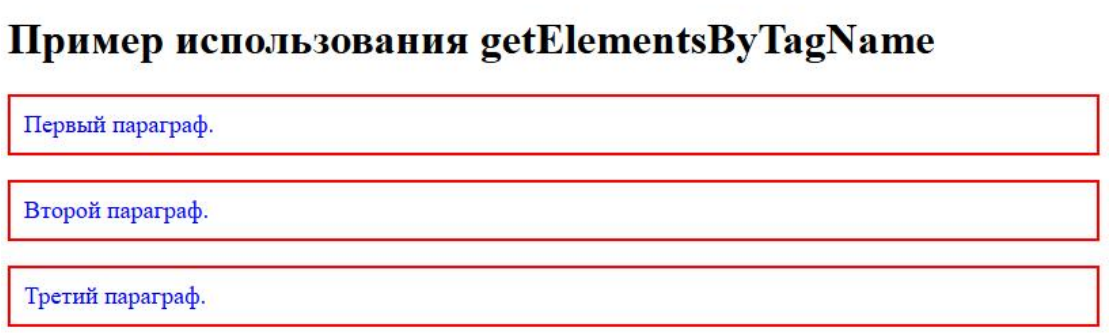


Рисунок 1 - Скриншот результата. у которых синий текст и красная рамка

2. Объяснение свойства children

Назначение

Свойство children возвращает **коллекцию дочерних элементов** (исключая текстовые узлы и комментарии).

Пример

```
<div id="parent">
    <p>Дочерний элемент 1</p>
    <span>Дочерний элемент 2</span>
```

```
    <div>Дочерний элемент 3</div>
</div>

<script>
    const parent = document.getElementById('parent');
    const childElements = parent.children; // Получаем [p, span, div]
    console.log(childElements);
</script>
```

Вывод в консоли:

HTMLCollection(3) [p, span, div]

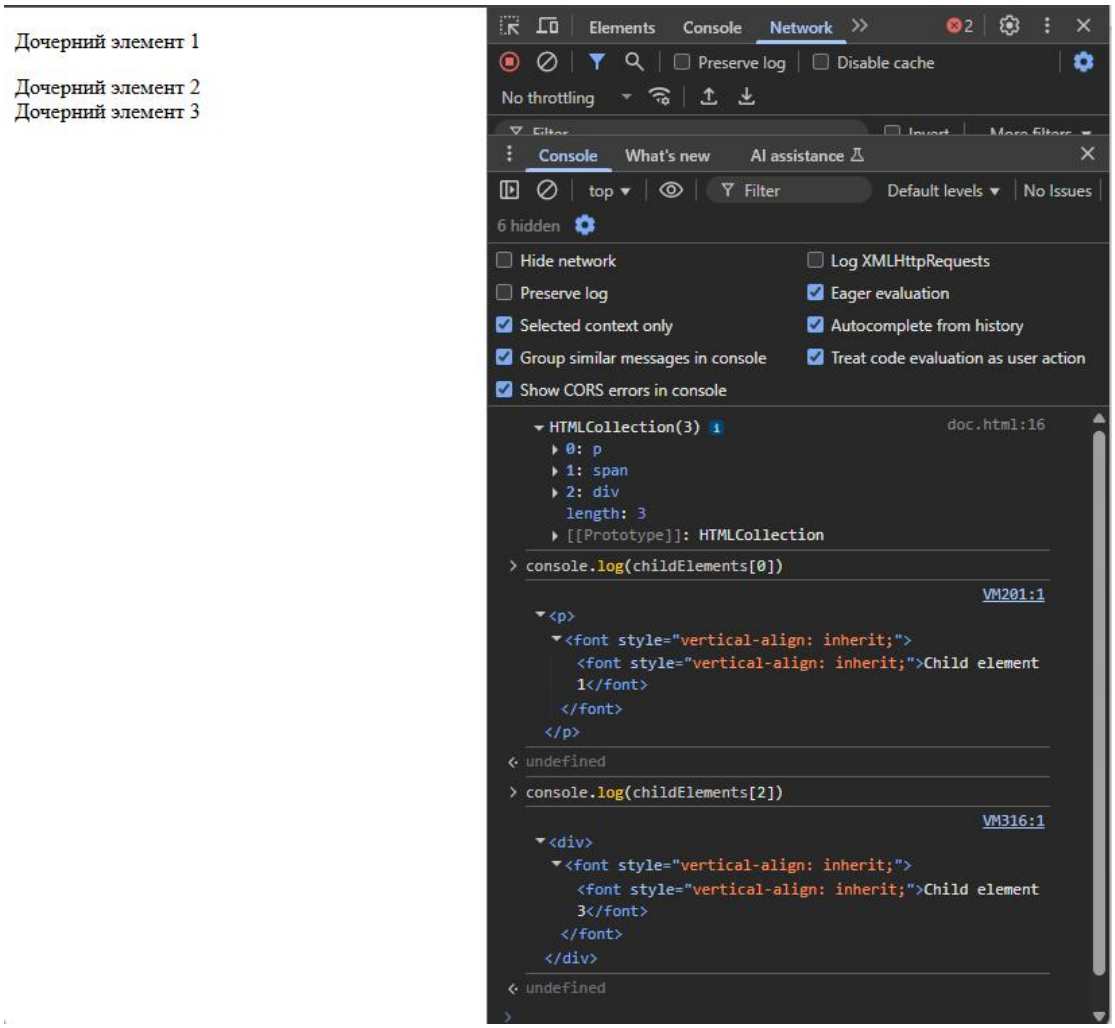


Рисунок 2 - Вывод в консоли: HTMLCollection(3) [p, span, div]

Пояснение:

-parent.children возвращает только элементы, вложенные в .

- В отличие от childNodes, children игнорирует текстовые узлы и комментарии.

3. Способы добавления изображения на веб-страницу

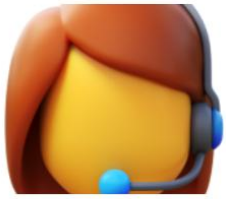
Способ	Пример	Комментарий
Тег <img>	<img src="image.jpg" alt="Пример">	Основной метод. Обязателен атрибут alt.
CSS background-image	div { background-image: url('image.jpg'); }	Используется для фоновых изображений.
Тег <picture>	<picture><source srcset="image.webp"><img src="image.jpg" alt="Запасной вариант"></picture>	Поддержка современных форматов с фолбэком.
SVG прямо в HTML	<svg width="100" height="100"><circle cx="50" cy="50" r="40" fill="red"/></svg>	Для векторной графики.
Base64-вставка	<img src="...">	Изображение встроено в код (увеличивает размер страницы).

Способы добавления изображений на веб-страницу

1. Тег <img>

Стандартный способ вставки изображений. Обязателен атрибут `alt` для доступности.

```
<img src="accounting.png" alt="Пример изображения" width="200">
```



2. CSS background-image

Используется для фоновых изображений и декоративных элементов.

```
div {
    background-image: url('image.jpg');
    background-size: cover;
    width: 200px;
    height: 150px;
}
```

3. Тег <picture>

Позволяет загружать разные изображения для разных условий (разрешение экрана, формат).

```
<picture>
    <source srcset="image.webp" type="image/webp">
    <source srcset="image.jpg" type="image/jpeg">
    <img src="fallback.jpg" alt="Пример picture">
</picture>
```

4. SVG прямо в HTML

Для векторной графики, которая масштабируется без потери качества.

```
<svg width="200" height="150">
    <rect x="10" y="10" width="180" height="130" fill="#9b59b6"/>
    <circle cx="100" cy="75" r="50" fill="#e67e22"/>
</svg>
```

5. Base64-вставка

Изображение встроено прямо в код, что увеличивает размер страницы, но уменьшает HTTP-запросы.

```
<img src="..." alt="Base64 example">
```

4. Пример использования селектора атрибута

[атрибут]

Сценарий

Нужно стилизовать все ссылки, открывающиеся в новой вкладке (target="_blank").

Пример

/* Выбирает все ссылки с target="_blank" */

a[target="_blank"] {

```
    color: red;
    text-decoration: none;
}
```

Google (открывается в новой вкладке)

Результат: Ссылка отображается красным цветом без подчеркивания.

Пояснение:

- Селектор a[target="_blank"] применяет стили только к ссылкам с указанным атрибутом.

- Полезно для визуального выделения внешних ссылок.

Примеры использования селекторов атрибутов в CSS

1. Базовый пример: стилизация ссылок с target="_blank"

Теоретическое объяснение:

Селектор атрибутов позволяет выбирать элементы на основе их атрибутов или значений атрибутов. В данном случае:

- `a[target="_blank"]` выбирает все элементы `<a>` с точным значением атрибута `target="_blank"`
- Это полезно для визуального выделения внешних ссылок, которые открываются в новой вкладке
- Такой подход улучшает UX, давая пользователю визуальную подсказку о поведении ссылки
- Селекторы атрибутов имеют специфичность 0-1-0 (класс=0-1-0, id=1-0-0)

Код:

```
/* CSS */
a[target="_blank"] {
    color: #e74c3c;
    text-decoration: none;
    font-weight: bold;
    border-left: 3px solid #e74c3c;
    padding-left: 10px;
    transition: all 0.3s ease;
}

/* HTML */
<a href="https://google.com" target="_blank">Google</a>
```

Результат:

Обычная ссылка (внутренняя)

Google (внешняя, откроется в новой вкладке) (безопасное соединение)

Яндекс (внешняя, откроется в новой вкладке) (безопасное соединение)

2. Другие полезные примеры селекторов атрибутов

2.1. Стилизация полей ввода по типу

```
input[type="text"] {
    border: 2px solid #3498db;
    padding: 8px;
}
```

Текстовое поле | Поле пароля | Email

2.2. Выбор изображений по значению alt

```
img[alt~="пример"] {
    border: 3px solid #2ecc71;
}
```

2.3. Выбор ссылок по протоколу

```
a[href^="https"]::after {
    content: " (безопасное соединение)";
    color: #27ae60;
    font-size: 0.8em;
}
```

HTTP ссылка
HTTPS ссылка (безопасное соединение)

2.4. Кастомные атрибуты данных

```
[data-tooltip]:hover::after {
    content: attr(data-tooltip);
    /* стили подсказки */
}
```

Наведи на меня

Таблица всех типов селекторов атрибутов

Селектор	Пример	Описание
`[attr]`	`a[title]`	Элементы с указанным атрибутом
`[attr=value]`	`input[type="text"]`	Точное совпадение значения атрибута
`[attr~=value]`	`img[alt~="пример"]`	Атрибут содержит указанное слово
`[attr|=value]`	`[lang|="ru"]`	Атрибут начинается с указанного значения (или равно)
`[attr^=value]`	`a[href^="https"]`	Атрибут начинается с указанного значения
`[attr$=value]`	`a[href$=".pdf"]`	Атрибут заканчивается указанным значением
`[attr*=value]`	`a[href*="google"]`	Атрибут содержит указанное значение

Эти скриншоты взяты из сгенерированного мной html-кода

5. Функция JavaScript: сумма элементов матрицы, меньших Z

Постановка задачи

Написать функцию, которая принимает матрицу и число Z, а возвращает сумму элементов, меньших Z.

Решение

```
function sumElementsLessThanZ(matrix, Z) {
    let sum = 0;
    for (let row of matrix) {
        for (let num of row) {
            if (num < Z) sum += num;
        }
    }
    return sum;
}


// Пример использования
const matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
];
const Z = 5;
```

```
console.log(sumElementsLessThanZ(matrix, Z)); // Вывод: 10 (1+2+3+4)
```

Пояснение:

-Функция принимает двумерный массив (matrix) и число Z.

-Два вложенных цикла for...of перебирают все элементы матрицы.

-Если элемент меньше Z, он добавляется к сумме.

-Результат выводится в консоль.