Лабораторная работа № 6

Тема: Поиск элементов и работа с коллекциями.

Цель: Освоить методы поиска элементов в DOM, работу с коллекциями HTMLCollection и NodeList, использование CSS-селекторов и data-атрибутов.

Для выполнения работы создайте HTML-файл со следующей структурой:

```html
<!DOCTYPE html>
<html lang="ru">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>DOM Laboratory</title>
    <style>
        .highlight { background-color: yellow; }
        .active { color: green; font-weight: bold; }
        .hidden { display: none; }
        .error { color: red; border: 1px solid red; }
    </style>
</head>
<body>
    <div id="container">
        <h1 id="main-title">Заголовок страницы</h1>

        <div class="content-block">
            <p class="text-item">Первый параграф</p>
            <p class="text-item active">Второй параграф</p>
            <p class="text-item">Третий параграф</p>
        </div>

        <ul class="menu">
            <li data-category="home" data-priority="high">Главная</li>
            <li data-category="about" data-priority="medium">О нас</li>
            <li data-category="services" data-priority="high">Услуги</li>
            <li data-category="contact" data-priority="low">Контакты</li>
        </ul>

        <div class="buttons">
            <button class="btn primary" data-action="save">Сохранить</button>
            <button class="btn secondary" data-action="cancel">Отмена</button>
            <button class="btn danger" data-action="delete">Удалить</button>
        </div>

        <form id="user-form">
            <input type="text" name="username" placeholder="Имя пользователя">
            <input type="email" name="email" placeholder="Email">
            <input type="password" name="password" placeholder="Пароль">
```

```
        </form>

        <div class="products">
            <div class="product-card" data-id="101" data-price="1500">
                <h3>Товар 1</h3>
                <span class="price">15000 тг.</span>
            </div>
            <div class="product-card" data-id="102" data-price="2300">
                <h3>Товар 2</h3>
                <span class="price">23000 тг.</span>
            </div>
            <div class="product-card" data-id="103" data-price="890">
                <h3>Товар 3</h3>
                <span class="price">8900 тг.</span>
            </div>
        </div>
    </div>

    <!-- Здесь будет ваш JavaScript код -->
    <script src="laba.js"></script>
</body>
</html>
```

Задание:

Задание 1. Поиск по data-атрибутам

- Найдите все элементы с атрибутом data-priority="high" и добавьте им класс "highlight"
- Найдите кнопку с data-action="save" и измените её текст на "Сохранено!"
- Выведите в консоль значения всех data-id атрибутов у товарных карточек

Задание 2. Работа с dataset

- Для всех товарных карточек получите значения data-price через свойство dataset
- Найдите товар с самой высокой ценой и добавьте ему класс "highlight"
- Добавьте новый data-атрибут data-status="popular" для товара с id="102"

Задание 3*. Комплексное задание
Создайте функцию filterProducts(minPrice, maxPrice), которая:

- Принимает минимальную и максимальную цену как параметры
- Находит все товары в указанном ценовом диапазоне

- Добавляет найденным товарам класс "active"
- Всем остальным товарам добавляет класс "hidden"
- Возвращает количество найденных товаров

Протестируйте функцию с параметрами (1000, 2000).

Шпаргалка: DOM API - Поиск элементов и работа с коллекциями

```
// ## 🔍 Методы поиска элементов

// ### getElementById(id)
// HTML: <div id="header">Заголовок</div>
const header = document.getElementById('header');
console.log(header.textContent); // "Заголовок"

// Если элемент не найден - возвращает null
const notFound = document.getElementById('nonexistent');
console.log(notFound); // null


// ### getElementsByClassName(className)
// HTML: <div class="card">1</div><div class="card">2</div>
const cards = document.getElementsByClassName('card');
console.log(cards.length); // 2

// Живая коллекция - обновляется автоматически
const newDiv = document.createElement('div');
newDiv.className = 'card';
document.body.appendChild(newDiv);
console.log(cards.length); // теперь 3!


// ### getElementsByTagName(tagName)
// Найти все параграфы
const paragraphs = document.getElementsByTagName('p');

// Поиск внутри конкретного элемента
const sidebar = document.getElementById('sidebar');
const sidebarLinks = sidebar.getElementsByTagName('a');


// ### querySelector(cssSelector)

// Первый элемент с классом
```

```
const firstCard = document.querySelector('.card');

// По сложному селектору
const activeButton =
document.querySelector('button.btn.active');

// Комбинированный селектор
const firstLinkInNav = document.querySelector('nav a:first-
child');

// Если не найден - возвращает null
const missing = document.querySelector('.not-exists');


// ### querySelectorAll(cssSelector)
// Все элементы с классом
const allCards = document.querySelectorAll('.card');

// Сложный селектор
const activeItems = document.querySelectorAll('li.menu-
item.active');

// Статическая коллекция - НЕ обновляется
console.log(allCards.length); // например, 3
// Добавим новый элемент
const newCard = document.createElement('div');
newCard.className = 'card';
document.body.appendChild(newCard);
console.log(allCards.length); // всё ещё 3!


// ## 📦 Работа с коллекциями
// ### HTMLCollection vs NodeList
// HTMLCollection (живая, только элементы)
const htmlCollection =
document.getElementsByClassName('item');

// NodeList (обычно статическая, может содержать любые узлы)
const nodeList = document.querySelectorAll('.item');
const liveNodeList = document.childNodes; // живая NodeList


// ### Обход коллекций
const items = document.querySelectorAll('.gallery-item');
```

```
// Классический for
for (let i = 0; i < items.length; i++) {
    items[i].style.opacity = '0.5';
}

// for...of (работает с коллекциями)
for (const item of items) {
    item.classList.add('processed');
}

// forEach (только у NodeList)
items.forEach(function(item, index) {
    item.textContent = `Элемент ${index + 1}`;
});

// Стрелочная функция
items.forEach((item, index) => {
    item.setAttribute('data-position', index);
});


// ### Преобразование в массив
const buttons = document.getElementsByTagName('button');

// Способ 1: Array.from()
const buttonsArray1 = Array.from(buttons);

// Способ 2: Spread оператор
const buttonsArray2 = [...buttons];

// Способ 3: Array.prototype.slice.call()
const buttonsArray3 = Array.prototype.slice.call(buttons);

// Теперь доступны методы массива
buttonsArray1.map(btn => btn.textContent.toUpperCase());
buttonsArray2.filter(btn => btn.disabled);


// ## 🎯 CSS-селекторы
// ### Простые селекторы
// По тегу
document.querySelector('h1');

// По классу
```

```
document.querySelector('.navigation');

// По ID
document.querySelector('#content');

// По атрибуту
document.querySelector('[type="submit"]');
document.querySelector('[href*="mailto"]');


// ### Комбинированные селекторы
// Потомок через пробел
document.querySelector('nav ul li');

// Непосредственный дочерний элемент
document.querySelector('div > p');

// Соседний элемент
document.querySelector('h1 + p');

// Класс + тег
document.querySelector('button.primary');

// Несколько классов
document.querySelector('.card.featured.large');


// ### Псевдоклассы
// Первый/последний ребёнок
document.querySelector('li:first-child');
document.querySelector('li:last-child');

// n-й элемент
document.querySelector('tr:nth-child(2n)'); // чётные строки

// Не содержащий класс
document.querySelector('div:not(.hidden)');

// По состоянию
document.querySelector('input:checked');
document.querySelector('input:disabled');


// ## 🏷️ Data-атрибуты
// ### Поиск по data-атрибутам
```

```
// HTML: <div data-user-id="123" data-
role="admin">User</div>
// Через селектор атрибута
const user = document.querySelector('[data-user-id="123"]');
const admins = document.querySelectorAll('[data-
role="admin"]');

// Частичное совпадение
const articles = document.querySelectorAll('[data-
category*="tech"]');

// Существование атрибута
const tracked = document.querySelectorAll('[data-
analytics]');


// ### Работа через getAttribute/setAttribute
const element = document.querySelector('.user-card');

// Получить значение
const userId = element.getAttribute('data-user-id');
const role = element.getAttribute('data-role');

// Установить значение
element.setAttribute('data-status', 'online');
element.setAttribute('data-last-seen', Date.now());

// Проверить существование
if (element.hasAttribute('data-premium')) {
    // Пользователь премиум
}

// Удалить атрибут
element.removeAttribute('data-temp-flag');


// ### Работа через dataset
// HTML: <div data-user-name="john" data-max-items="10"
data-is-active="true">

const element = document.querySelector('.config');

// Чтение (автоматическое преобразование в camelCase)
console.log(element.dataset.userName);    // "john"
```

```
console.log(element.dataset.maxItems);   // "10" (строка!)
console.log(element.dataset.isActive);   // "true" (строка!)

// Запись
element.dataset.userName = 'jane';
element.dataset.itemCount = 25;
element.dataset.newFlag = 'yes';

// Удаление
delete element.dataset.tempData;

// Перебор всех data-атрибутов
for (const key in element.dataset) {
    console.log(`${key}: ${element.dataset[key]}`);
}


// ## 🛠️ Полезные приёмы
// ### Проверка на существование
// Безопасная работа с элементами
const element = document.querySelector('#maybe-exists');

if (element) {
    element.style.display = 'none';
} else {
    console.log('Элемент не найден');
}

// Или через optional chaining (ES2020)
element?.classList.add('processed');


// ### Поиск внутри элемента
const container = document.querySelector('.products');

// Искать только внутри контейнера
const productCards = container.querySelectorAll('.card');
const firstButton = container.querySelector('button');


// ### Динамические селекторы
function findByDataId(id) {
    return document.querySelector(`[data-id="${id}"]`);
}

function findByClass(className) {
```

```
    return document.querySelectorAll(`.${className}`);
}

// Использование
const product = findByDataId('prod-123');
const warnings = findByClass('alert-warning');


// ### Комбинирование методов
// Найти все активные элементы и изменить их
document.querySelectorAll('.item.active')
    .forEach(item => {
        item.style.fontWeight = 'bold';
        item.dataset.processed = 'true';
    });

// Цепочка действий
Array.from(document.getElementsByClassName('card'))
    .filter(card => card.dataset.price > 1000)
    .forEach(expensiveCard => {
        expensiveCard.classList.add('premium');
    });


// ### Производительность
// ❌ Медленно - повторный поиск
for (let i = 0; i < 100; i++) {
    document.querySelector('.status').textContent = `Шаг ${i}`;
}

// ✅ Быстро - поиск один раз
const statusElement = document.querySelector('.status');
for (let i = 0; i < 100; i++) {
    statusElement.textContent = `Шаг ${i}`;
}


// ## 🤦 Частые ошибки
// ### 1. Обращение к коллекции как к элементу
// ❌ Неправильно
const items = document.querySelectorAll('.item');
items.style.color = 'red'; // Ошибка!

// ✅ Правильно
items.forEach(item => {
    item.style.color = 'red';
});


// ### 2. Забывание о том, что коллекция может быть пустой
// ❌ Может упасть, если элементов нет
```

```
const buttons = document.querySelectorAll('button');
buttons[0].click(); // Error если кнопок нет

// ✅ Безопасно
if (buttons.length > 0) {
    buttons[0].click();
}


// ### 3. Путаница с живыми и статическими коллекциями
// HTMLCollection - живая
const liveList = document.getElementsByClassName('item');

// NodeList от querySelectorAll - статическая
const staticList = document.querySelectorAll('.item');

// При добавлении нового элемента:
// liveList.length увеличится автоматически
// staticList.length останется прежним
```

Памятка по синтаксису:

Метод	Возвращает	Тип коллекции	Живая?
`getElementById`	Element или null	-	-
`getElementsByClassName`	HTMLCollection	HTMLCollection	Да
`getElementsByTagName`	HTMLCollection	HTMLCollection	Да
`querySelector`	Element или null	-	-
`querySelectorAll`	NodeList	NodeList	Нет