**Практическая №3**

**Теоретический материал по работе с DOM в JavaScript**

В этом материале мы рассмотрим основные методы получения и манипуляции элементами HTML с использованием JavaScript. Мы освоим, как взаимодействовать с элементами на веб-странице, изменять их стиль, добавлять и удалять элементы, а также добавлять обработчики событий.

## 1. Получение элементов

JavaScript предоставляет различные методы для получения элементов из DOM (Document Object Model). Ниже приведены наиболее распространенные из них:

### document.body

Используется для получения элемента <body> текущего документа. Это позволяет вам получать доступ к всему содержимому страницы, представленному в рамках тела документа.

### document.getElementById("id")

Позволяет получить элемент по его уникальному идентификатору (id). Метод возвращает первый элемент, найденный с заданным id, или null, если элемента с таким id не существует.

**Пример:**

**javascript**

```
let header = document.getElementById("header");

document.querySelector(".block")
```

Позволяет найти первый элемент, соответствующий указанному селектору CSS. В данном случае, селектор получает первый элемент с классом block.

**Пример:**

**javascript**

```
let blockElement = document.querySelector(".block");

document.querySelector("#block")
```

Находит первый элемент с указанным id. Использование # указывает, что мы ищем элемент с конкретным id.

**Пример:**

**javascript**

```
let block = document.querySelector("#block");

document.querySelectorAll(".list-item")
```

Возвращает NodeList (список узлов) всех элементов, соответствующих указанному селектору. Например, все элементы с классом list-item.

**Пример:**

**javascript**

```
let listItems = document.querySelectorAll(".list-item");
```

**2. Изменение стилей**

JavaScript позволяет изменять стили элементов в реальном времени. Вы можете получать доступ к свойствам стиля через свойство style элемента.

**Пример:**

**javascript**

```
let el = document.querySelector(".block");

el.style.background = "red"; // устанавливаем красный фон

el.style.fontSize = "8px"; // изменяем размер шрифта на 8px
```

Для изменения стилей с помощью CSS можно использовать:

**css**

```
font-size: 8px; /* в CSS */
```

### 3. Добавление элементов в HTML

JavaScript позволяет динамически изменять содержимое страницы. Например, вы можете добавлять новые элементы в список.

**Пример:**

Исходное состояние HTML:

**html**

```
<ul class="list">
  <li>1</li>
  <li>2</li>
  <li>3</li>
</ul>
```

Добавление нового элемента:

**javascript**

```
let ul = document.querySelector(".list");
// Добавление элемента перед списком
```

```
ul.insertAdjacentHTML('beforebegin', '<p>OOO</p>');
// Добавление нового элемента в начало списка
ul.insertAdjacentHTML('afterbegin', '<li>222</li>');
// Добавление нового элемента в конец списка
ul.insertAdjacentHTML('beforeend', '<li>333</li>');
// Добавление элемента после списка
ul.insertAdjacentHTML('afterend', '<p>KKK</p>');
```

**4. Удаление элементов в HTML**

Удаление элемента с помощью JavaScript также просто:

**javascript**

```
let ul = document.querySelector(".list");
ul.remove(); // удаляет элемент списка
```

**5. Добавление и удаление классов у элементов в HTML**

Переписывание класса:

**javascript**

```
let ul = document.querySelector(".list");
ul.className = "newClass"; // заменяет все классы на новый
```

Добавление класса:

**javascript**

```
ul.classList.add('active'); // добавляет новый класс 'active'
```

**Удаление класса:**

**javascript**

```
ul.classList.remove('active'); // удаляет класс 'active'
```

### 6. Добавление прослушки событий

JavaScript позволяет добавлять обработчики событий для реакции на пользовательские действия, такие как клики:

**javascript**

```
let ul = document.querySelector(".list");

// Добавление обработчика события "click"

ul.addEventListener('click', function(event) {

    // Действие при клике

});
```

Функция прослушки события

Пример функции, которая добавляет класс active при клике:

**javascript**

```
ul.addEventListener('click', (event) => {

    event.preventDefault(); // отключает стандартное поведение

    event.target.classList.add('active'); // добавляет класс к целевому элементу

});
```

event.preventDefault(): Отключает стандартное поведение элемента, например, переход по ссылке, если это элемент.

Выполнить задания, с каждым блоком увеличивается сложность заданий, выполнять по порядку.

| Блок | Количество заданий |
|---|---|
| 1 | 10 |
| 2 | 8 |
| 3 | 3 |
| 4 | 1 |

**Блок 1**

1. Изменение текста при клике на кнопку

Задача: Создайте кнопку с текстом "Изменить заголовок". При клике на неё измените текст заголовка с id `header` на "Заголовок изменен!".

2. Добавление класса при наведении мыши

Задача: При наведении мыши на элемент с классом `box`, добавьте ему класс `hovered`, а при уходе мыши — удалите этот класс.

3. Отслеживание изменений в текстовом поле

Задача: Отслеживайте изменения в текстовом поле с id `inputField`. Каждое изменение должно отображаться в элементе с id `output`.

4. Изменение фона при двойном клике

Задача: При двойном клике на элемент с id `backgroundBox` измените его фон на зеленый.

5. Включение/отключение кнопки при изменении чекбокса

Задача: Когда чекбокс с id `agreeCheckbox` отмечен, кнопка с id `submitButton` должна стать активной, если нет — оставаться неактивной.

6. Отслеживание изменения размера окна

Задача: При изменении размера окна (resize) измените текст внутри элемента с id `windowSize`, чтобы отображались текущие ширина и высота окна.

7. Изменение текста при наведении на ссылку

Задача: При наведении на ссылку с id `link`, текст ссылки должен измениться на "Вы навели мышь", а при уходе мыши — вернуться к исходному тексту.

### 8. Скрытие элемента при клике

Задача: При клике на элемент с id `hideMe` этот элемент должен скрыться (добавить стиль `display: none`).

### 9. Ограничение длины вводимого текста

Задача: Отслеживайте ввод текста в поле с id `textField`. Если длина текста превышает 10 символов, остановите дальнейший ввод (удалите дополнительные символы).

### 10. Переключение классов при клике

Задача: Создайте кнопку с id `toggleButton`. При каждом клике на неё элемент с id `toggleBox` должен переключать классы `active` и `inactive`.

## Блок 2

### 1. Создание элемента и добавление его в DOM

Задача: Создайте кнопку с текстом "Нажми меня" и добавьте ее в конец документа, используя JavaScript.

### 2. Добавление класса к элементу

Задача: Найдите элемент с id `example` и добавьте ему класс `highlight`.

### 3. Удаление класса у элемента

Задача: Удалите класс `active` у элемента с классом `button`.

### 4. Изменение стилей элемента

Задача: Найдите первый элемент с классом `box` и установите для него ширину 100px и высоту 100px с красным фоном.

### 5. Изменение текста элемента

Задача: Измените текст внутри элемента с id `header` на "Добро пожаловать!"

### 6. Добавление обработчика события

Задача: Добавьте обработчик клика на кнопку с id `myButton`, который выводит "Кнопка нажата" в консоль.

### 7. Добавление элементов списка

Задача: Добавьте новый элемент списка в конец `ul` с id `myList`.

### 8. Удаление элемента из DOM

Задача: Удалите элемент с id `deleteMe` из документа.

### 9. Манипуляция с атрибутами

Задача: Найдите элемент с id `link` и измените его атрибут `href` на `https://www.example.com`.

### 10. Поиск нескольких элементов и изменение их стилей

Задача: Найдите все элементы с классом `highlight` и измените их цвет текста на синий.

## Блок 3

### 1. Изменение текста нескольких элементов из массива

Задача: У вас есть массив строк. Напишите функцию, которая изменяет текст всех элементов с классом `item`, используя строки из массива.

### 2. Создание элементов списка на основе массива объектов

Задача: У вас есть массив объектов, представляющих товары. Напишите функцию, которая создает элементы списка (`li`) для каждого товара и добавляет их в `ul` с id `productList`.

### 3. Добавление классов к элементам из массива объектов

Задача: Есть массив объектов, в котором каждому объекту соответствует элемент с определенным id. Напишите функцию, которая добавляет класс `highlight` к каждому элементу, id которого есть в массиве.

### 4. Динамическое создание таблицы из массива объектов

Задача: Создайте таблицу на основе массива объектов. Каждый объект представляет строку таблицы, а его свойства — ячейки строки. Добавьте эту таблицу в `div` с id `tableContainer`.

### 5. Изменение стилей нескольких элементов из массива

Задача: У вас есть массив объектов, где каждому объекту соответствует элемент с определенным id и стилями. Напишите функцию, которая изменяет стили для каждого элемента на основе данных массива.

**Блок 4**

## 1. Динамическое создание галереи с переключением стилей и обработкой кликов

Задача: Создайте галерею из изображений на основе массива объектов. При клике на любое изображение оно должно увеличиваться, а остальные — уменьшаться. В каждой строке галереи должно быть не более 3 изображений.

Используемые методы и техники:

- Создание элементов на основе массива
- Добавление и удаление классов
- Обработчики событий (`addEventListener`)
- Изменение стилей

## 2. Динамическая форма с валидацией и отображением результата

Задача: Создайте форму с полями "Имя" и "Возраст". После отправки формы добавьте введенные данные в таблицу. Если возраст меньше 18, подсветите поле красным.

Используемые методы и техники:

- Создание элементов
- Добавление классов
- Обработка событий формы
- Изменение стилей
- Валидация данных

## 3. Таблица товаров с динамическим обновлением и фильтрацией

Задача: Создайте таблицу с товарами на основе массива объектов. Добавьте возможность фильтрации товаров по минимальной цене через текстовое поле. Когда пользователь вводит значение, отображаются только товары дороже указанной цены.

Используемые методы и техники:

- Создание элементов
- Массивы и объекты
- Обработчики событий
- Изменение DOM на основе пользовательского ввода

## Заключение

В этом материале мы рассмотрели основные методы работы с DOM в JavaScript. Мы научились получать элементы, изменять их стили, добавлять и удалять элементы, а также добавлять обработчики событий для взаимодействия с пользовательским вводом. Понимание этих основ является ключом к разработке современных интерактивных веб-приложений.