ДИСЦИПЛИНА	Фронтенд и бэкенд разработка
ИНСТИТУТ	Институт перспективных технологий и индустриального программирования
КАФЕДРА	Кафедра индустриального программирования
ВИД УЧЕБНОГО МАТЕРИАЛА	Практические занятия
ПРЕПОДАВАТЕЛЬ	Загородних Николай Анатольевич
СЕМЕСТР	1 семестр, 2024-2025 гг.

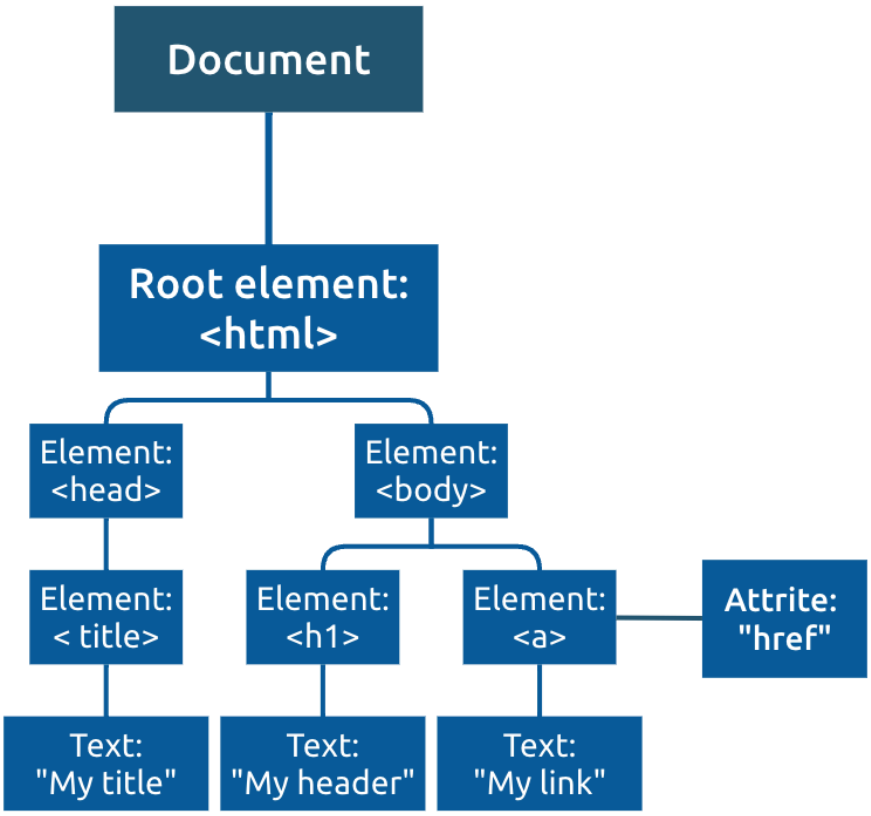
Практическое занятие 11. Работа с DOM-структурой веб-сайта и взаимодействие с объектами в JavaScript

Изучаемые вопросы

1. Основы работы с DOM.
2. Поиск элементов на странице.
3. Изменение свойств объектов.
4. Работа с событиями.
5. Создание и редактирование объектов.

Краткая теория

DOM (Document Object Model) — это программный интерфейс для HTML и XML документов. Он представляет структуру документа в виде дерева объектов, что позволяет динамически изменять содержимое, структуру и стиль веб-страницы.



С помощью модели объектов JavaScript получает всю необходимую мощность для создания динамического HTML:

• JavaScript может изменять все HTML-элементы на странице;

• JavaScript может изменять все HTML-атрибуты на странице;

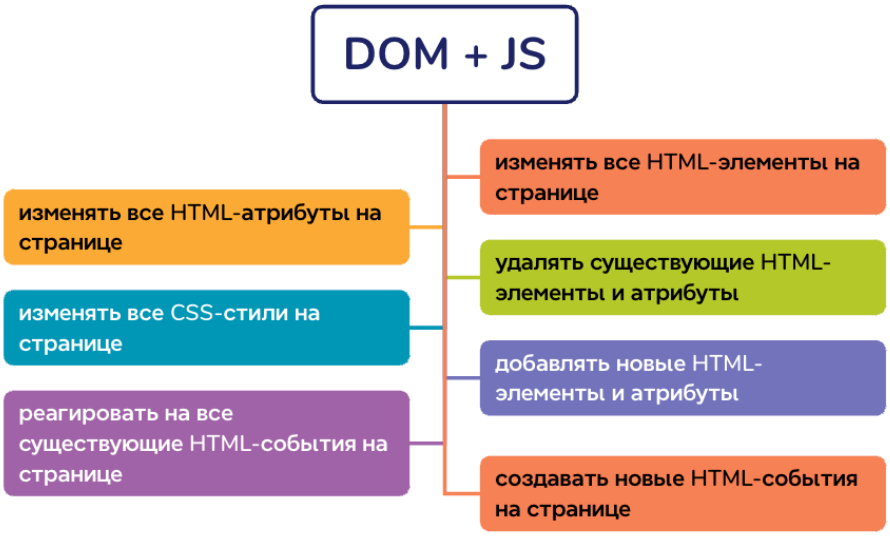
• JavaScript может изменять все CSS-стили на странице;

• JavaScript может удалять существующие HTML-элементы и атрибуты;

• JavaScript может добавлять новые HTML-элементы и атрибуты;

• JavaScript может реагировать на все существующие HTML-события на странице;

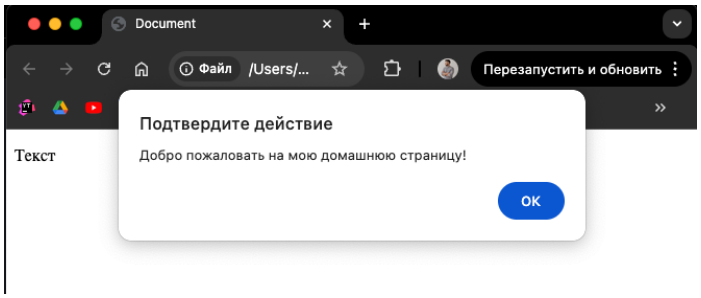
• JavaScript может создавать новые HTML-события на странице.



Вот простой пример, показывающий, как JavaScript может взаимодействовать с DOM:

```
index.html > html > body
1   <!DOCTYPE html>
2   <html lang="en">
3   <head>
4       <meta charset="UTF-8">
5       <meta name="viewport" content="width=device-width, initial-scale=1.0">
6       <title>Document</title>
7   </head>
8   <body onload="window.alert('Добро пожаловать на мою домашнюю страницу!');">
9       <p>Текст</p>
10
11  </body>
12  </html>
```

Этот код вызывает всплывающее окно при загрузке страницы.



Объекты в JavaScript — это коллекции значений и более сложных сущностей. Каждый объект имеет свойства и методы, которые могут быть изменены или добавлены в процессе выполнения программы.

Важные типы данных при работе с DOM

1. Document

document — это корневой объект, представляющий весь HTML или XML документ. Он предоставляет доступ ко всем элементам и узлам в документе.

Пример.

```
js > JS main.js > ...
1   // Получение заголовка документа
2   const title = document.title;
3   console.log(title); // Выведет заголовок текущей страницы
4
```

2. Element

Element представляет собой HTML-тег в документе. Каждый элемент имеет свои свойства и методы, которые позволяют манипулировать его содержимым и атрибутами.

Пример.

```
js > JS main.js > ...
1   // Получение элемента с id "myElement"
2   const myElement = document.getElementById("myElement");
3   myElement.textContent = "Новый текст!"; // Изменение текста элемента
4
```

3. NodeList

NodeList — это коллекция узлов, которая возвращается методами, такими как querySelectorAll(). Это не массив, но его можно перебирать с помощью циклов.

Пример.

```
js > JS main.js > ...
1   // Получение всех элементов <p> на странице
2   const paragraphs = document.querySelectorAll("p");
3   paragraphs.forEach((p) => {
4       console.log(p.textContent); // Вывод текста каждого параграфа
5   });
6
```

4. Attribute

Attribute представляет собой атрибут элемента, например, id, class, src и т.д. Атрибуты могут быть получены и изменены через свойства элемента.

Пример.

```
js > JS main.js > ...
1   // Получение элемента с id "myImage" и изменение его атрибута src
2   const myImage = document.getElementById("myImage");
3   myImage.setAttribute("src", "newImage.png"); // Изменение источника изображения
4
```

5. NamedNodeMap

NamedNodeMap — это коллекция атрибутов элемента. Она позволяет получить доступ к атрибутам по имени и использовать методы для работы с ними.

Пример.

```
js > JS main.js > ...
1   // Получение элемента с id "myElement" и его атрибутов
2   const myElement = document.getElementById("myElement");
3   const attributes = myElement.attributes; // Получение всех атрибутов элемента
4
5   // Перебор всех атрибутов
6   for (let i = 0; i < attributes.length; i++) {
7       console.log(${attributes[i].name}: ${attributes[i].value});
8   }
9
```

Эти типы данных являются основными при работе с DOM и позволяют эффективно взаимодействовать с элементами веб-страницы.

6. Интерфейсы и объекты

Объекты в DOM могут реализовывать несколько интерфейсов. Например, элемент таблицы реализует интерфейсы HTMLTableElement, Element и Node.

Пример.

```
js > JS main.js > ...
1   // Создаем таблицу
2   const table = document.createElement('table');
3
4   // Создаем строку и ячейки
5   const row = document.createElement('tr');
6   const cell = document.createElement('td');
7   cell.textContent = 'Ячейка таблицы';
8
9   // Добавляем ячейку в строку, а строку в таблицу
10  row.appendChild(cell);
11  table.appendChild(row);
12
13  // Добавляем таблицу в тело документа
14  document.body.appendChild(table);
15
16  // Теперь мы можем использовать методы HTMLTableElement
17  console.log(table.rows.length); // Выводит количество строк в таблице
18
```

Основные методы работы с DOM:

- document.getElementById() - получение элемента по ID.

- document.querySelector() - получение первого элемента, соответствующего CSS-селектору.

- element.innerHTML - изменение содержимого элемента.

- element.style - изменение стилей элемента.

Работа с Element

Возможности:

- Получение элементов по ID, классу или селектору.

- Изменение текста и атрибутов элементов.

Синтаксис:

const element = document.getElementById("id"); // по ID

const elements = document.getElementsByClassName("class"); // по классу

const element = document.querySelector("selector"); // по селектору

Примеры

В приведенном коде мы выполняем несколько манипуляций с DOM. Давайте разберем, что происходит в этом коде:

1. Изменение текста заголовка. Мы получаем элемент заголовка <h1> по его id с помощью document.getElementById("header").

Затем мы изменяем его текстовое содержимое на "Добро пожаловать!" с помощью header.textContent = "Добро пожаловать!";. Это заменяет текст внутри заголовка.

2. Получение элементов по классу. Мы получаем коллекцию элементов с классом item с помощью document.getElementsByClassName("item"). Это возвращает HTMLCollection, содержащую все элементы с указанным классом.

3. Изменение стиля первого элемента. Мы изменяем цвет текста первого элемента из коллекции items на синий с помощью items[0].style.color = "blue";. Это изменяет стиль первого элемента с классом item.

4. Изменение HTML первого элемента. Мы используем document.querySelector(".item"), чтобы получить первый элемент с классом item.

Затем мы изменяем его HTML-содержимое, заменяя текст на Измененный текст. Это делает текст жирным и заменяет предыдущее содержимое элемента.

Работа с Forms

Возможности:

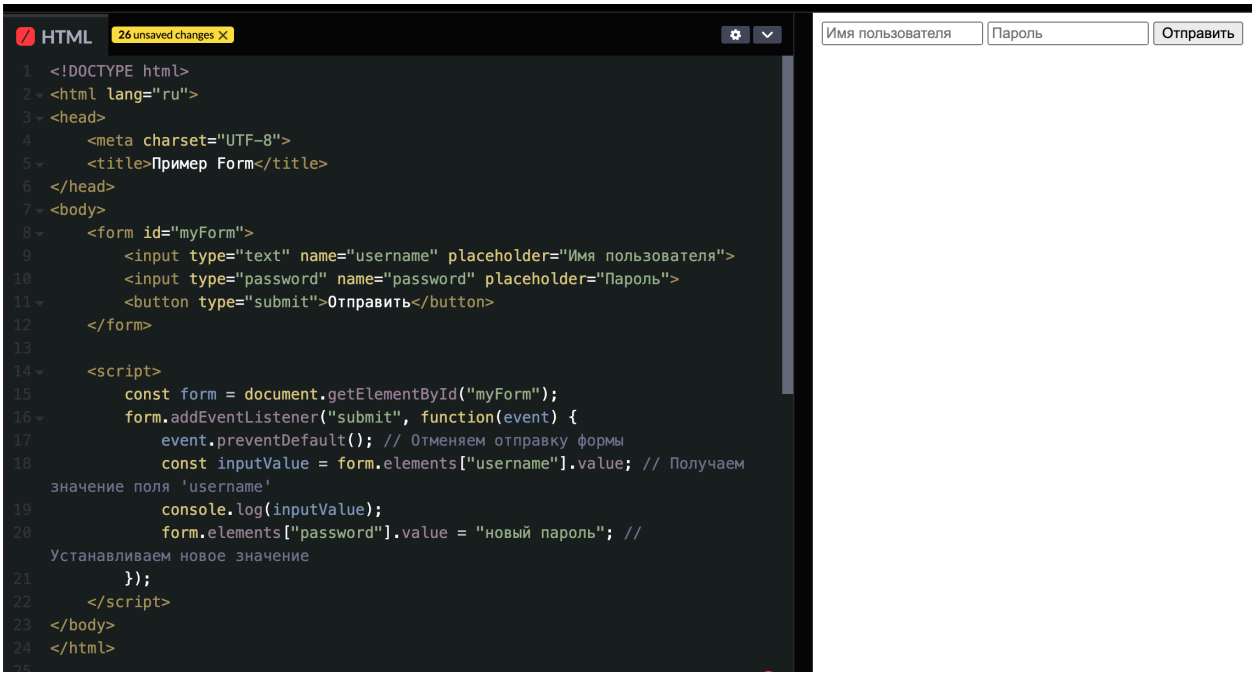
- Получение значений полей ввода.

- Изменение значений полей.

Синтаксис:

const value = form.elements["name"].value; // получение значения поля

Пример



В данном коде мы создаем простую HTML-форму и используем JavaScript для манипуляции с элементами этой формы через DOM. Давайте разберем, что именно происходит:

1. Создание формы. Мы создаем форму с id="myForm", которая содержит два поля ввода: одно для имени пользователя (username) и одно для пароля (password), а также кнопку для отправки.

2. Добавление обработчика события. Внутри тега <script> мы получаем элемент формы с помощью document.getElementById("myForm") и сохраняем его в переменной form.

Мы добавляем обработчик события submit к форме с помощью form.addEventListener("submit", function(event) {...}). Это означает, что когда форма будет отправлена (при нажатии на кнопку "Отправить"), будет выполнена указанная функция.

3. Отмена стандартного поведения. Внутри обработчика события мы используем event.preventDefault(), чтобы отменить стандартное поведение отправки формы. Это позволяет нам выполнять дополнительные действия (например, обработку данных), не перезагружая страницу.

4. Получение значения поля ввода. Мы получаем значение поля ввода для имени пользователя с помощью form.elements["username"].value. Это позволяет извлечь текст, введенный пользователем в поле.

5. Вывод значения в консоль. Полученное значение имени пользователя выводится в консоль с помощью console.log(inputValue). Это может быть полезно для отладки или проверки введенных данных.

6. Изменение значения поля ввода пароля. Мы устанавливаем новое значение для поля ввода пароля с помощью form.elements["password"].value = "новый пароль";. Это изменяет текст в поле пароля на "новый пароль".

Полезные источники информации

1. MDN Web Docs - Introduction to the DOM
https://developer.mozilla.org/ru/docs/Web/API/Document_Object_Model/Introduction
2. JS работа с DOM на w3school
https://www.w3schools.com/js/js_htmldom.asp
3. JavaScript.info - Объекты
https://learn.javascript.ru/object

4. Обработчик формы

https://learn.javascript.ru/forms-submit

Тренажеры

Codecademy - Learn JavaScript

https://www.codecademy.com/learn/introduction-to-javascript)

https://codepen.io/pen

Задание

Часть 1: Работа с DOM

1. Найдите заголовок H1 на странице и измените его текст на "Добро пожаловать на наш сайт!".

2. Измените цвет текста заголовка H2 на красный.

3. Найдите и измените текст первого параграфа на "Это новый текст параграфа.".

4. Скрыть встроенное видео, установив его стиль display на none.

Часть 2: Работа с объектами

1. Создайте объект formData, который будет содержать данные из формы (имя, e-mail, телефон, дата, комментарий).

2. Реализуйте функцию submitForm, которая будет собирать данные из формы и записывать их в объект formData.

3. Реализуйте проверки:

- Проверьте, что поля name, email и comment не пустые.

- Проверьте, что поле phone содержит только цифры.

- Проверьте корректность введенного e-mail (можно использовать простое регулярное выражение).

3. Добавьте метод printData в объект formData, который будет выводить собранные данные в консоль в формате:

Имя: [имя]

E-mail: [email]

Телефон: [телефон]

Дата: [дата]

Комментарий: [комментарий]

Часть 3: Обработчик формы

1. Добавьте обработчик события на кнопку отправки формы, который будет вызывать функцию submitForm и предотвращать стандартное поведение формы.

Критерии оценивания

1. Синтаксис (30%): Правильное использование синтаксиса JavaScript и методов работы с DOM.

2. Работа с DOM (30%): Корректное выполнение всех заданий по изменению элементов страницы.

3. Работа с объектами (20%): Создание и использование объекта formData, а также корректная реализация методов.

4. Логика программы (20%): Корректность выполнения задания и логика программного кода.

Форма сдачи

Студенты должны представить свой код в виде файлов .html .css .js, который можно открыть в браузере. Также важно прокомментировать код, чтобы объяснить, что делает каждый элемент.