The Document Object Model (DOM) is a programming interface for web documents.
It represents the page so that programs can change the document structure, style, and content. DOM manipulation is the process of changing the HTML, CSS, and content of a webpage using JavaScript.
we will explore how to manipulate the DOM using JavaScript.

Getting Started with DOM Manipulation

To manipulate the DOM using JavaScript, we need to access the DOM elements.
We can do that by using the document object provided by the browser.
The document object is the root node of the DOM tree and represents the entire HTML document.

To access a specific element on a web page, we use the getElementById() method.
This method retrieves an element based on its id attribute. Here's an example:

```
const element = document.getElementById('example');
```

The getElementById() method returns a reference to the DOM element with the specified id.
We can use this reference to change the element's content, attributes, and style.

Changing Element Content

To change the content of an element, we can use the innerHTML property.

This property sets or returns the HTML content within an element.

Here's an example:

```
const element = document.getElementById('example');
element.innerHTML = 'New content';
```

In this example, we're changing the content of an element with the id 'example' to 'New content'. The innerHTML property can also be used to add new HTML content to an element.

Changing Element Attributes

To change the attributes of an element, we can use the setAttribute() method.

This method sets the value of an attribute on the specified element.

Here's an example:

```
const element = document.getElementById('example');
element.setAttribute('class', 'new-class');
```

In this example, we're changing the class attribute of an element with the id 'example' to 'new-class'.

The setAttribute() method can be used to change any attribute of an element.

Changing Element Style

To change the style of an element, we can use the style property.
This property sets or returns the CSS style declaration of an element.
Here's an example:

```
const element = document.getElementById('example');
element.style.color = 'red';
```

In this example, we're changing the color style of an element with the id 'example' to red.
The style property can be used to change any style of an element.

Conclusion

DOM manipulation is a powerful technique for changing the HTML, CSS, and content of a webpage using JavaScript.
In this lesson, we explored how to access and manipulate DOM elements using JavaScript.
We learned how to change the content, attributes, and style of an element.

examples of DOM manipulation with JavaScript:

1. **Adding and Removing Classes**

To add or remove a class from an element, we can use the classList property.
The classList property returns a DOMTokenList object, which represents the list of classes of an element.
We can use the add() method to add a class and the remove() method

to remove a class.

Here's an example:

```
const element = document.getElementById('example');
element.classList.add('new-class');
element.classList.remove('old-class');
```

In this example, we're adding the class 'new-class' to an element with the id 'example' and removing the class 'old-class'.

2. Creating New Elements

To create a new element, we can use the createElement() method. This method creates a new element with the specified tag name. We can then add the new element to the DOM using the appendChild() method. Here's an example:

```
const newElement = document.createElement('div');
newElement.innerHTML = 'New element';
document.body.appendChild(newElement);
```

In this example, we're creating a new div element with the content 'New element' and adding it to the end of the body element.

3. Event Handling

To handle events on an element, we can use the addEventListener() method.

This method adds an event listener to an element and specifies the

function to be called when the event occurs.
Here's an example:

```
const button = document.getElementById('my-button');
button.addEventListener('click', function() {
console.log('Button clicked'); });
```

In this example, we're adding a click event listener to a button element with the id 'my-button' and logging a message to the console when the button is clicked.

4. Traversing the DOM

To traverse the DOM tree and access elements, we can use the parentElement, firstChild, lastChild, nextSibling, and previousSibling properties.
These properties return the parent, first child, last child, next sibling, and previous sibling of an element, respectively.
Here's an example:

```
const parent = document.getElementById('parent');
const firstChild = parent.firstChild;
const lastChild = parent.lastChild;
const nextSibling = firstChild.nextSibling;
const previousSibling = lastChild.previousSibling;
```

In this example, we're accessing the parent, first child, last child, next sibling, and previous sibling of an element with the id 'parent'.

These are just a few examples of the many ways you can manipulate the DOM using JavaScript. With practice and experimentation, you can create rich and dynamic web pages that respond to user input and events.

Exercises

Exercise 1 - Create a Basic HTML Page

Create a basic HTML page with a title, header, paragraph, and image. Use JavaScript to access the header element and change its text content to "Hello World!".

Exercise 2 - Change Element Attributes

Create a button element with the text "Click Me". Use JavaScript to access the button element and change its background color to blue and its text color to white.

Exercise 3 - Create and Append Elements

Create a list element with the text "Item 1". Use JavaScript to create a new list item element with the text "Item 2" and append it to the list.

Exercise 4 - Create a Dropdown Menu

Create a dropdown menu with three options: "Option 1", "Option 2", and "Option 3". Use JavaScript to add an event listener to the dropdown menu that logs the selected option to the console.

Exercise 5 - Create a Dynamic Quiz

Create a dynamic quiz that displays multiple-choice questions with options and allows users to select an answer and move to the next question. Use JavaScript to manipulate the DOM and keep track of the user's progress and score. This exercise will require more advanced DOM manipulation techniques, such as creating and updating elements dynamically and handling user input.

Exercise 6 - Dynamic To-Do List

In this exercise, you'll create a dynamic to-do list that allows users to add, delete, and complete tasks using JavaScript and DOM manipulation.

Instructions:

1. Create a basic HTML file with an input field, a button to add tasks, and an empty unordered list to display tasks.
2. Use JavaScript to access the input field and button elements and add an event listener to the button element. When the button is clicked, a new task should be added to the list with the content of the input field.
3. Add a delete button to each task in the list. When the delete button is clicked, the corresponding task should be removed from the list.
4. Add a checkbox to each task in the list. When the checkbox is checked, the corresponding task should be marked as completed and styled differently.
5. Add a clear button to the page. When the clear button is clicked, all completed tasks should be removed from the list.