DOM Manipulation Part II

Dynamically modify content, change attributes, manage classes and create elements

Introduction

Today, we'll explore more advanced techniques for manipulating the Document Object Model (DOM).

Introduction

We'll learn how to modify element content, change attributes, manage CSS classes, and dynamically manipulate HTML elements.

What You'll Learn

- Modify element content
- Change element attributes
- Manage CSS classes
- Create new elements
- Manipulate DOM elements

Getting Started

We'll work with HTML, CSS and JavaScript in this lecture. Our basic HTML will look like this…

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0
    <title>DOM manipulation part 2</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div id="myDiv">Initial content of the div.</div>
    <a id="myLink" href="#">Click me!</a>
    <div id="parentDiv">
        <div id="referenceNode">Reference Node</div>
    </div>
    <button id="toggleButton">Toggle Hidden Class</button>
    <script src="./script.js"></script>
</body>
</html>
```

Getting Started

And the page will look like this…

Initial content of the div.

Click me!

Reference Node

Toggle Hidden Class

This div can be toggled.

Modifying Content

innerHTML: The innerHTML property allows us to get or set the HTML content inside an element.

Modifying Content

It can be used to insert text, HTML markup, or both into an element.

```
const div = document.getElementById('myDiv');
div.innerHTML = '<strong>This is bold text</strong> and normal text.';
```

Modifying Content

And we can see that text on our page has changed.

This is bold text and normal text.

Click me!

Reference Node

Toggle Hidden Class

This div can be toggled.

Modifying Content

textContent: The textContent property allows you to get or set the text content of an element and its descendants.

Modifying Content

Unlike innerHTML, it treats the content purely as text, not HTML.

```
const div = document.getElementById('myDiv');
div.textContent = 'This is some text content.';
```

Modifying Content

Now, we can see this…

This is some text content.

Click me!

Reference Node

Toggle Hidden Class

This div can be toggled.

Changing Attributes

setAttribute: The setAttribute method is used to add a new attribute to an element or change the value of an existing attribute.

Changing Attributes

We can use it like this…

```
const link = document.getElementById('myLink');
link.setAttribute('href', 'https://www.perseverenow.org/');
```

Changing Attributes

And now, our link is live. If we click it, we go to the Persevere website.

Changing Attributes

We can also remove attributes.

```
const link = document.getElementById('myLink');
link.removeAttribute('href');
```

Changing Attributes

Now, it’s no longer a clickable link.

This is some text content.
Click me!
Reference Node
Toggle Hidden Class
This div can be toggled.

Add and Remove Classes

classList: The classList property provides methods to add, remove, and toggle CSS classes on an element.

Add and Remove Classes

We can add a class to an element.

```
const div = document.getElementById('myDiv');

// Adding a class
div.classList.add('new-class');
```

Add and Remove Classes

And if we have our class defined in our CSS…

```
.new-class { font-weight: bold;
}
```

Add and Remove Classes

We'll see its effects on our page...

Initial content of the div.

Click me!

Reference Node

Toggle Hidden Class

Add and Remove Classes

We can also remove a class by using the remove method on the classList property and passing in a class as a string.

Add and Remove Classes

We can use the toggle method to add a class if it doesn’t exist or remove it if it does.

Add and Remove Classes

Our JavaScript…

```
function toggle() {
    const div = document.getElementById('myDiv');
    div.classList.toggle('active');
}
```

Add and Remove Classes

Update our HTML...

```
<button id="toggleButton" onclick="toggle()">
```

Add and Remove Classes

And our CSS…

```
.active {
    background-color: aqua;
}
```

Add and Remove Classes

When we click the button, what will happen?

Add and Remove Classes

It will add, or remove that class.

Initial content of the div.

Click me!

Reference Node

Toggle Hidden Class

Create Elements

createElement: The createElement method is used to create a new HTML element.

Create Elements

We can use it like this…

```
const newDiv = document.createElement('div');
newDiv.textContent = 'This is a new div.';
```

Create Elements

At this point, our page will look like…

Initial content of the div.

Click me!

Reference Node

Toggle Hidden Class

Create Elements

Why? Where is that new div we just created?

Create Elements

To be able to see it on our page, we have to give it a place on the DOM to live.

Create Elements

appendChild: Adds a new child element to the end of the list of children of a specified parent element.

Create Elements

Here, we're adding our new div.

```
const parent = document.getElementById('parentDiv');
parent.appendChild(newDiv);
```

Create Elements

Now we can see it on our page.

Initial content of the div.
Click me!
Reference Node
This is a new div.
Toggle Hidden Class

Create Elements

If we inspect the page, we'll see it is a child of our parent div, below referenceNode.

Create Elements

```
▼<div id="parentDiv">
    <div id="referenceNode">Reference Node</div>
    <div>This is a new div.</div> == $0
  </div>
```

Create Elements

insertBefore: Inserts a new node before a reference node as a child of a specified parent node.

Create Elements

We can use that like this…

```
const referenceNode = document.getElementById('referenceNode');
parent.insertBefore(newDiv, referenceNode);
```

Create Elements

In this case, our page will look…

Initial content of the div.
Click me!
This is a new div.
Reference Node
Toggle Hidden Class

Create Elements

And if we inspect our page...

```
▼<div id="parentDiv">
    <div>This is a new div.</div> == $0
    <div id="referenceNode">Reference Node</div>
  </div>
```

Create Elements

We can see that we call the insert before method as a property of the parent and then specify what element to place our new element before.

Remove elements

removeChild: Removes a child node from the DOM.

```
const child = document.getElementById('referenceNode');
parent.removeChild(child);
```

Create Elements

On our page, we'll see referenceNode is gone.

Initial content of the div.

Click me!

This is a new div.

Toggle Hidden Class

What You Learned

- Modify element content
- Change element attributes
- Manage CSS classes
- Create new elements
- Manipulate DOM elements