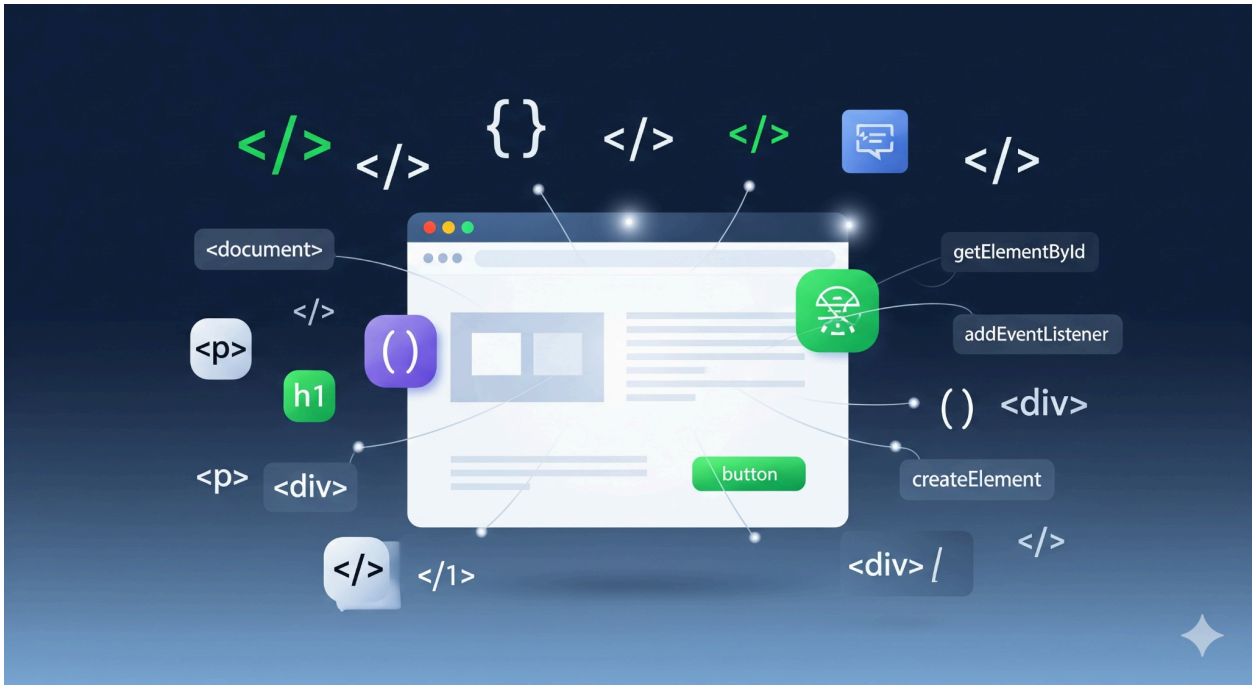
JavaScript DOM Coding Exercises 🚀



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. The following exercises are designed to help you understand and practice key DOM manipulation concepts.

Exercise 1: Selecting Elements

Objective: Learn how to select single and multiple elements from the DOM.

What will be accomplished: You'll select a heading by its ID and a list of items by their class.

Instructions:

1. Create an HTML file with an `<h1>` tag with an ID of "main-heading" and a `<ul>` with three `<li>` elements, each having the class "list-item."
2. Use `document.getElementById()` to select the heading and `document.getElementsByClassName()` to select the list items.
3. Log the selected elements to the console to verify your selections.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>DOM Exercise 1</title>
</head>
<body>
  <h1 id="main-heading">Hello World!</h1>
  <ul>
    <li class="list-item">Item 1</li>
    <li class="list-item">Item 2</li>
    <li class="list-item">Item 3</li>
  </ul>
  <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

JavaScript

```
// Select the h1 element by its ID
const heading = document.getElementById('main-heading');
console.log(heading); // This will log the h1 element

// Select all li elements by their class name
const listItems = document.getElementsByClassName('list-item');
console.log(listItems); // This will log an HTMLCollection of the li elements
```

Exercise 2: Modifying Element Content

Objective: Change the text and HTML content of an element.

What will be accomplished: You will change the text of a paragraph and the HTML content of a division.

Instructions:

1. Create a paragraph with an ID of "info-para" and a `<div>` with an ID of "content-div."
2. Use the `.innerText` property to change the text of the paragraph.
3. Use the `.innerHTML` property to add bold text to the `<div>`.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 2</title>
</head>
<body>
```

```
    <p id="info-para">This is a paragraph.</p>
    <div id="content-div">Current content.</div>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

```
JavaScript
// Select the paragraph
const para = document.getElementById('info-para');
para.innerText = 'The paragraph text has been changed!';

// Select the div
const div = document.getElementById('content-div');
div.innerHTML = 'This div now contains <b>bold text</b>.';
```

Exercise 3: Styling Elements

Objective: Learn to change the CSS styles of an element.

What will be accomplished: You will change the color, font size, and background color of an element using the `.style` property.

Instructions:

1. Create a `<h2>` element with an ID of "style-me."
2. Use the `.style` property to change the color to blue, the `fontSize` to 24px, and the `backgroundColor` to light gray. Remember to use **camelCase** for CSS properties with hyphens.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 3</title>
</head>
<body>
    <h2 id="style-me">Style Me!</h2>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

JavaScript

```
// Select the h2 element
const header = document.getElementById('style-me');

// Change multiple styles
header.style.color = 'blue';
header.style.fontSize = '24px';
header.style.backgroundColor = 'lightgray';
header.style.padding = '10px';
```

Exercise 4: Creating and Appending Elements

Objective: Programmatically create new elements and add them to the page.

What will be accomplished: You will create a new list item and append it to an existing unordered list.

Instructions:

1. Create an empty `<ul>` in your HTML with an ID of "my-list."
2. In JavaScript, use `document.createElement()` to create a new `<li>` element.
3. Set the `innerText` of the new list item.
4. Use `document.appendChild()` to add the new list item to the `<ul>`.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>DOM Exercise 4</title>
</head>
<body>
  <ul id="my-list">
    <li>Existing Item 1</li>
  </ul>
  <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

JavaScript

```
// Select the ul element
const list = document.getElementById('my-list');

// Create a new li element
const newItem = document.createElement('li');

// Set the content of the new item
```

```
newItem.innerText = 'New Item added with JavaScript!';

// Append the new item to the ul
list.appendChild(newItem);
```

Exercise 5: Removing Elements

Objective: Remove an element from the DOM.

What will be accomplished: You will remove a specific element from the page.

Instructions:

1. Create a `<div>` with an ID of "remove-me" inside a parent `<div>` with an ID of "container."
2. Select the element you want to remove.
3. Use the `.remove()` method on the selected element to remove it from the document.

Code Breakdown:

HTML:

```
HTML
<!DOCTYPE html>
<html lang="en">
<head>
  <title>DOM Exercise 5</title>
</head>
<body>
  <div id="container">
    <p>This is the parent container.</p>
    <div id="remove-me">This element will be removed.</div>
  </div>
  <script src="script.js"></script>
</body>
```

</html>

JavaScript (`script.js`):

JavaScript

```
// Select the element to be removed
const elementToRemove = document.getElementById('remove-me');

// Remove the element from the DOM
elementToRemove.remove();
```

Exercise 6: Handling User Events (Click)

Objective: Respond to user interactions, specifically a button click.

What will be accomplished: When a button is clicked, a message will appear on the page.

Instructions:

1. Add a `<button>` to your HTML with an ID of "my-button." Also, add an empty paragraph with an ID of "message."
2. Select both the button and the paragraph.
3. Use the `.addEventListener()` method to listen for a `click` event on the button.
4. Inside the event handler function, change the `innerText` of the message paragraph.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 6</title>
</head>
```

```
<body>
    <button id="my-button">Click Me!</button>
    <p id="message"></p>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

```
JavaScript
// Select the button and the message paragraph
const button = document.getElementById('my-button');
const message = document.getElementById('message');

// Add a click event listener to the button
button.addEventListener('click', () => {
    message.innerText = 'Button was clicked!';
});
```

Exercise 7: Changing Element Attributes

Objective: Learn to get, set, and remove attributes on an element.

What will be accomplished: You will change the `src` attribute of an image and remove its `alt` attribute.

Instructions:

1. Add an `<img>` tag to your HTML with a placeholder image source and an `alt` attribute.
2. Use `.setAttribute()` to change the `src` of the image to a new image URL.
3. Use `.getAttribute()` to get the new `src` and log it.
4. Use `.removeAttribute()` to remove the `alt` attribute.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 7</title>
</head>
<body>
    <img id="my-image" src="https://via.placeholder.com/150" alt="Placeholder image">
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

JavaScript

```
// Select the image element
const image = document.getElementById('my-image');

// Change the src attribute
image.setAttribute('src', 'https://picsum.photos/200/300');

// Log the new src attribute
const newSrc = image.getAttribute('src');
console.log('New image source:', newSrc);

// Remove the alt attribute
image.removeAttribute('alt');
```

Exercise 8: Working with Forms (Input Value)

Objective: Get and use the value from a form input field.

What will be accomplished: When a button is clicked, the text from a text input will be displayed.

Instructions:

1. Create a `<input>` of type `text` and a button. Give them IDs like "name-input" and "submit-btn."
2. Select both the input and the button.
3. Add a `click` event listener to the button.
4. Inside the event handler, get the value of the input using the `.value` property and display it in an alert or a paragraph.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 8</title>
</head>
<body>
    <input type="text" id="name-input" placeholder="Enter your name">
    <button id="submit-btn">Submit</button>
    <p id="display-name"></p>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

JavaScript

```
// Select the input, button, and display paragraph
const input = document.getElementById('name-input');
const button = document.getElementById('submit-btn');
const displayPara = document.getElementById('display-name');

// Add a click event listener
button.addEventListener('click', () => {
    const name = input.value;
    if (name) {
        displayPara.innerText = `Hello, ${name}!`;
    } else {
        displayPara.innerText = 'Please enter your name.';
    }
});
```

Exercise 9: Toggling Classes

Objective: Add or remove a CSS class to an element to change its style.

What will be accomplished: When a button is clicked, an element will toggle between two different visual states using CSS classes.

Instructions:

1. Create an HTML `<div>` with an ID of "toggle-box" and a button with an ID of "toggle-btn."
2. Define two simple CSS classes, `.active` and `.inactive`, with different background colors.
3. Select the `<div>` and the button.
4. Add a `click` event listener to the button.
5. Inside the event handler, use the `.classList.toggle()` method on the `<div>` to add or remove the `.active` class.

Code Breakdown:

HTML:

HTML

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 9</title>
    <style>
        #toggle-box {
            width: 100px;
            height: 100px;
            background-color: lightgray;
            transition: background-color 0.3s;
        }
        .active {
            background-color: dodgerblue;
        }
    </style>
</head>
<body>
    <div id="toggle-box"></div>
    <button id="toggle-btn">Toggle Color</button>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

JavaScript

```
// Select the box and the button
const box = document.getElementById('toggle-box');
const button = document.getElementById('toggle-btn');
```

```
// Add a click event listener
button.addEventListener('click', () => {
    box.classList.toggle('active');
});
```

Exercise 10: Traversing the DOM

Objective: Navigate between parent, child, and sibling elements.

What will be accomplished: You will select a specific element and then access its parent, children, and sibling elements.

Instructions:

1. Create an HTML structure with a parent `<div>` containing a `<ul>` and a `<p>`. The `<ul>` should have multiple `<li>` children.
2. Select one of the `<li>` elements.
3. From that `<li>`, use properties like `.parentNode`, `.children`, `.nextElementSibling`, and `.previousElementSibling` to access other elements.
4. Log the results to the console to see the relationships.

Code Breakdown:

HTML:

```
HTML
<!DOCTYPE html>
<html lang="en">
<head>
    <title>DOM Exercise 10</title>
</head>
<body>
    <div id="container">
        <p>A paragraph before the list.</p>
```

```
    <ul id="my-list">
      <li>Item 1</li>
      <li id="target-item">Item 2</li>
      <li>Item 3</li>
    </ul>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

JavaScript (`script.js`):

```
JavaScript
// Select the target li element
const targetItem = document.getElementById('target-item');

// Get the parent of the target li
const parent = targetItem.parentNode;
console.log('Parent of target item:', parent);

// Get the next sibling
const nextSibling = targetItem.nextElementSibling;
console.log('Next sibling:', nextSibling);

// Get the previous sibling
const previousSibling = targetItem.previousElementSibling;
console.log('Previous sibling:', previousSibling);

// Get the children of the parent ul
const children = parent.children;
console.log('Children of the list:', children);
```