# 🌟 Mastering DOM Manipulation in JavaScript 🌟

In web development, knowing how to access and manipulate elements in the Document Object Model (DOM) is 🔑! Here are some essential techniques to level up your DOM game:

- Change Page Title 📝
- Modify Background Colors 🎨
- Update Text Content 🖋️
- Create Dynamic Elements 🧩
- Remove Elements Dynamically ✂️
- Build Lists Like a Pro 📋
- Customize Images Effortlessly 🖼️
- Create Interactive Links 🔗
- Add Event Listeners with Ease 📢

Effortlessly Create and Append Elements 🆕

These skills are crucial for web developers working on dynamic and interactive websites. Keep learning, keep coding! 💪

## Accessing Elements in the DOM

In web development, accessing elements in the Document Object Model (DOM) is a fundamental skill. The DOM represents the structured content of a web page, and being able to manipulate it allows you to create dynamic and interactive web applications. In JavaScript, there are various methods to access DOM elements.

### 1. Accessing Elements by ID:

To access an element with a specific id attribute, use document.getElementById(id).

```
const myElement = document.getElementById('myId');
```

### 2. Accessing Elements by Class Name:

To access elements with a specific class name, use document.getElementsByClassName(className).

```
const elements =
document.getElementsByClassName('myClass');
```

### 3. Accessing Elements by Tag Name:

To access elements with a specific tag name, use document.getElementsByTagName(tagName).

```
const paragraphs = document.getElementsByTagName('p');
```

### 4. Accessing Elements by Selector:

To access elements using CSS selectors, use document.querySelector(selector) for the first matching element and document.querySelectorAll(selector) for all matching elements.

```
const element = document.querySelector('.myClass');
const elements =
document.querySelectorAll('p.myClass');
```

### 5. Accessing Parent and Child Elements:

You can access a parent element's children and a child element's parent using the parentNode and childNodes properties.

```
const parent =
document.getElementById('parentElement');
const children = parent.childNodes;
```

## 6. Accessing Form Elements:

To access form elements, you can use their name, id, or type attributes.

```
const usernameInput =
document.forms['myForm'].elements['username'];
```

## 7. Modifying Element Content:

You can change the content of an element using the textContent or innerHTML properties.

```
const element = document.getElementById('myElement');
element.textContent = 'New Content';
```

## 8. Modifying Element Attributes:

To change attributes like src, href, or class, access them and assign new values.

```
const image = document.getElementById('myImage');
image.src = 'new-image.jpg';
```

## 9. Creating New Elements:

You can create new elements and add them to the DOM using methods like `document.createElement(tagName)` and `element.appendChild(newElement)`.

```
const newDiv = document.createElement('div');
document.body.appendChild(newDiv);
```

### 10. Removing Elements:

To remove an element, access its parent and use the removeChild(childElement) method.

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

Mastering these techniques will empower you to build dynamic and interactive web applications with JavaScript. Whether you're changing content, attributes, or even creating entirely new elements, understanding how to access elements in the DOM is crucial.

## Coding Exercises

### Exercise 1: Change Page Title

Description: Change the title of the web page to "New Title."

Steps:

1. Access the <title> element in the DOM.
2. Change its textContent property to "New Title."

### Exercise 2: Change Background Color

Description: Change the background color of a specific <div> element to red.

Steps:

1. Access the <div> element using its id.
2. Change its style.backgroundColor property to "red."

### Exercise 3: Modify Text Content

Description: Change the text content of a paragraph to "Hello, JavaScript!"

Steps:

1. Access the <p> element by its id.
2. Change its textContent property to "Hello, JavaScript!"

### Exercise 4: Create a Button

Description: Create a new button element with the text "Click Me" and add it to the DOM.

Steps:

1. Create a new <button> element using document.createElement.
2. Set its textContent to "Click Me."
3. Append the button to an existing element in the DOM.

### Exercise 5: Remove an Element

Description: Remove a specific paragraph from the DOM.

Steps:

1. Access the <p> element to be removed using its id.
2. Access its parent element using parentNode.
3. Remove the paragraph using removeChild.

## Exercise 6: Create a List

Description: Create an unordered list (<ul>) and add three list items (<li>) with different text to it.

Steps:

1. Create a new <ul> element using document.createElement.
2. Create three <li> elements using document.createElement.
3. Set the text content of each <li> element.
4. Append the <li> elements to the <ul>.
5. Append the <ul> to an existing element in the DOM.

## Exercise 7: Modify an Image

Description: Change the src attribute of an image to a new image file.

Steps:

1. Access the <img> element using its id.
2. Change its src attribute to the path of the new image.

## Exercise 8: Create a Link

Description: Create an anchor (<a>) element with the text "Visit Google" and a link to Google.

Steps:

1. Create a new <a> element using document.createElement.
2. Set its textContent to "Visit Google."
3. Set its href attribute to "https://www.google.com."

4. Append the <a> element to an existing element in the DOM.

## Exercise 9: Add Event Listener

Description: Add a click event listener to a button element to display an alert when clicked.

Steps:

1. Access the button element using its id.
2. Use the addEventListener method to attach a click event listener.
3. Define a function to display an alert.

## Exercise 10: Create and Append Elements

Description: Create a new <div> element, set its class to "box," and append it to the body of the web page.

Steps:

1. Create a new <div> element using document.createElement.
2. Set its className to "box."
3. Append the <div> element to the body of the web page using appendChild.

These exercises will help you practice accessing and manipulating elements in the DOM using JavaScript.

## Coding exercise solutions

### Exercise 1: Change Page Title

```
// Access the <title> element
const pageTitle = document.querySelector('title');

// Change its textContent property
pageTitle.textContent = 'New Title';
```

### Exercise 2: Change Background Color

```
// Access the <div> element by its id
const myDiv = document.getElementById('myDiv');

// Change its background color property
myDiv.style.backgroundColor = 'red';
```

### Exercise 3: Modify Text Content

```
// Access the <p> element by its id
const myParagraph =
document.getElementById('myParagraph');
```

```
// Change its textContent property
myParagraph.textContent = 'Hello, JavaScript!';
```

Exercise 4: Create a Button

```
// Create a new <button> element
const newButton = document.createElement('button');

// Set its textContent
newButton.textContent = 'Click Me';

// Append it to an existing element (e.g., <body>)
document.body.appendChild(newButton);
```

Exercise 5: Remove an Element

```
// Access the <p> element to be removed by its id
const paragraphToRemove =
document.getElementById('paragraphToRemove');

// Access its parent element
const parentElement = paragraphToRemove.parentNode;

// Remove the paragraph
parentElement.removeChild(paragraphToRemove);
```

Exercise 6: Create a List

```
// Create a new <ul> element
const newList = document.createElement('ul');

// Create three <li> elements and set their text
content
for (let i = 1; i <= 3; i++) {
  const listItem = document.createElement('li');
  listItem.textContent = `List Item ${i}`;
  newList.appendChild(listItem);
}

// Append the <ul> to an existing element (e.g.,
<body>)
document.body.appendChild(newList);
```

Exercise 7: Modify an Image

```
// Access the <img> element by its id
const myImage = document.getElementById('myImage');

// Change its src attribute to a new image file
myImage.src = 'new-image.jpg';
```

Exercise 8: Create a Link

```
// Create a new <a> element
const newLink = document.createElement('a');

// Set its textContent
newLink.textContent = 'Visit Google';

// Set its href attribute
newLink.href = 'https://www.google.com';

// Append the <a> element to an existing element (e.g.,
<body>)
document.body.appendChild(newLink);
```

Exercise 9: Add Event Listener

```
// Access the button element by its id
const myButton = document.getElementById('myButton');

// Add a click event listener
myButton.addEventListener('click', () => {
  alert('Button Clicked!');
});
```

Exercise 10: Create and Append Elements

```
// Create a new <div> element
const newDiv = document.createElement('div');

// Set its className
newDiv.className = 'box';

// Append the <div> element to the body of the web page
document.body.appendChild(newDiv);
```

These solutions demonstrate how to access, manipulate, and interact with DOM elements using JavaScript.