Manipulating DOM Elements with JavaScript:

1. **innerHTML Property:** You can use the **innerHTML** property to set or get the HTML content of an element. For example, to change the content of a **<div>** element with the ID "myDiv," you can do:

   ```
   document.getElementById("myDiv").innerHTML = "New Content";
   ```

   This allows you to update the content, including text and HTML structure, dynamically.

2. **textContent Property:** If you only want to change the text content of an element and not the HTML structure, you can use the **textContent** property:

   ```
   document.getElementById("myDiv").textContent = "New Text Content";
   ```

   This is especially useful when you want to update text within elements like paragraphs, headings, or spans.

3. **setAttribute and getAttribute:** You can use the **setAttribute** method to set attributes on an element and **getAttribute** to retrieve attribute values. For example:

   ```
   const link = document.getElementById("myLink");
   link.setAttribute("href", "https://example.com");
   const hrefValue = link.getAttribute("href");
   ```

   This allows you to modify attributes like **href**, **src**, **class**, or custom data attributes.

4. **Style Property:** To change CSS properties of an element, you can access the **style** property of the element and set individual CSS properties:

   ```
   const myElement = document.getElementById("myElement");
   myElement.style.backgroundColor = "blue";
   myElement.style.fontSize = "18px";
   ```

   This technique is suitable for dynamically applying styles to elements.

5. **Create and Append Elements:** You can create new DOM elements and append them to the document using methods like **createElement** and **appendChild**. For example, to create a new paragraph and append it to a **<div>**:

   ```
   const newParagraph = document.createElement("p");
   newParagraph.textContent = "New Paragraph";
   document.getElementById("myDiv").appendChild(newParagraph);
   ```

   This is helpful for adding dynamic content to your page.

6. **Remove Elements:** To remove elements from the DOM, you can use the **remove()** method or access the parent element and use **removeChild()**. For example:

   ```
   const elementToRemove = document.getElementById("elementToRemove");

   elementToRemove.remove();
   ```

```
// or
      const parentElement = document.getElementById("parentElement");
       const childElement = document.getElementById("childElement");
       parentElement.removeChild(childElement);
```