CREATING HTML ELEMENTS DYNAMICALLY

LAB 1

1) Create an element: Use the document.createElement() method to create a new HTML element. You can then set its properties, such as its id, class, and innerHTML, and append it to an existing element using the appendChild() method.

```
const newDiv = document.createElement("div");
newDiv.innerHTML = "Hello World!";
document.body.appendChild(newDiv);
```

2) Modify an existing element: Use the getElementById() method to grab a reference to an existing element, and then modify its properties as needed.

```
const existingDiv = document.getElementById("existing-div");
existingDiv.innerHTML = "New Text";
```

3) Remove an element: Use the removeChild() method on the parent element to remove an existing element.

```
const existingDiv = document.getElementById("existing-div");
document.body.removeChild(existingDiv);
```

4) Add an event listener: Add an event listener to the element using the addEventListener() method. This can be used to execute a function when an event such as a click or a hover occurs.

```
const newDiv = document.createElement("div");
newDiv.innerHTML = "Hello World!";
newDiv.addEventListener("click", () => alert("You clicked the div"));
document.body.appendChild(newDiv);
```

LAB 2

Create a simple web application that allows users to add, edit, and delete todo items from a list. The application should have the following features:

CREATING HTML ELEMENTS DYNAMICALLY

1) A form for adding new todo items, with fields for the task name and a completion status.

2) The ability to view, edit, and delete existing todo items using dynamic HTML manipulation.

3) Use of LocalStorage or IndexedDB to save the todo items, so that the items are not lost when the user closes the browser or refreshes the page.

4) Use of JSON format to store and retrieve the todo items.

5) A filter feature that will allow the user to filter the list by completed or active items.

6) Use of CSS to style the list items and form.

7) Proper error handling mechanism

Technical requirements:

- Use of JavaScript and the DOM to create and manipulate the elements of the application.

- Use of JSON to store and retrieve the todo items data.

- Use of LocalStorage or IndexedDB to save the todo items data.

- Use of dynamic HTML manipulation to add, edit, and delete list items.

- Use of CSS and JavaScript to style the elements and add interactivity

- Use of Event listeners for handling user interactions and form submissions