Day 4 → JavaScript DOM Manipulation and Events:

The Document Object Model (DOM) represents the structure of an HTML document as a tree-like structure, where each element in the HTML document is represented as a node in the DOM tree. JavaScript allows you to interact with the DOM to modify the content, structure, and styling of webpages dynamically. Additionally, JavaScript provides event handling mechanisms to respond to user interactions and trigger actions based on those events.

Here are the key concepts and techniques related to JavaScript DOM manipulation and events:

1. Accessing DOM Elements:
   - Use methods like `getElementById`, `getElementsByClassName`, or `getElementsByTagName` to retrieve specific elements from the DOM.
   - Assign elements to variables for further manipulation.

2. Modifying DOM Elements:
   - Update the content of an element using the `textContent` or `innerHTML` properties.
   - Modify attributes of elements using properties like `src`, `href`, `className`, etc.
   - Change the styling of elements by modifying their CSS properties using the `style` property.

3. Creating and Inserting DOM Elements:

- Use the `createElement` method to create new DOM elements.
- Manipulate the newly created element's properties and content.
- Insert elements into the DOM using methods like `appendChild`, `insertBefore`, or `replaceChild`.

4. Event Handling:
- Register event listeners on elements using methods like `addEventListener`.
- Specify the event type (e.g., 'click', 'submit', 'mouseover') and the function to be executed when the event occurs.
- Access event-related information through the event object (e.g., target element, event type).

5. Responding to Events:
- Perform actions when events are triggered, such as validating form input, updating content, or invoking functions.
- Use conditional statements and loops to implement dynamic behavior based on user interactions.

Hands-on Project: Form Manipulation with Insert, Edit, and Delete Functionality:

For the project on form manipulation with insert, edit, and delete functionality, you'll need to apply the concepts of JavaScript DOM manipulation and events. Here's an outline of the steps you can follow:

1. Create the HTML structure:
   - Design a form using HTML elements like `<form>`, `<input>`, `<button>`, etc.
   - Include appropriate attributes (e.g., `id`, `class`, `name`) to identify elements uniquely.

2. Access DOM elements:
   - Use JavaScript to access the form and its input fields by their IDs or other relevant selectors.
   - Assign them to variables for further manipulation.

3. Handle form submission:
   - Register an event listener on the form's `submit` event.
   - Implement a function that executes when the form is submitted.
   - Retrieve the input values from the form fields.
   - Perform appropriate actions, such as validating the inputs, creating a new entry, and updating the DOM.

4. Insert new entries:
   - Create a function that takes the input values as parameters.
   - Use DOM manipulation techniques to create new elements representing the entry.
   - Insert the new entry into the appropriate section of the DOM (e.g., a table, a list).

5. Edit existing entries:
   - Add an edit button or mechanism to each entry in the form.

   - Register event listeners for the edit action.
   - Implement a function that allows editing of the selected entry.
   - Update the corresponding DOM elements with the edited values.

6. Delete entries:
   - Add a delete button or mechanism to each entry in the form.
   - Register event listeners for the delete action.
   - Implement a function that removes the selected entry from the DOM.

7. Apply styling and UX considerations:
   - Utilize CSS to style the form,

```
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      font-family: Arial, sans-serif;
    }

    form {
      display: flex;
      flex-direction: column;
      max-width: 300px;
    }

    label {
      margin-bottom: 10px;
    }

    input[type="text"], input[type="email"], input[type="date"] {
      padding: 5px;
      margin-bottom: 10px;
    }

    input[type="submit"], input[type="button"] {
      padding: 10px;
      cursor: pointer;
    }

    input[type="submit"] {
      background-color: #4CAF50;
      color: white;
      border: none;
    }

    input[type="button"] {
      background-color: #008CBA;
      color: white;
      border: none;
    }

    table {
      border-collapse: collapse;
      margin-top: 20px;
    }

    th, td {
      padding: 8px;
      border: 1px solid #000000;
    }

    th {
      background-color: #4CAF50;
      color: white;
    }
  </style>
</head>
```

```
<body>
  <h2>Form</h2>

  <form id="myForm">
    <label for="name">Name:</label>
    <input type="text" id="name" name="name" required>

    <label for="email">Email:</label>
    <input type="email" id="email" name="email" required>

    <label for="dob">Date of Birth:</label>
    <input type="date" id="dob" name="dob" required>

    <input type="submit" value="Submit">
    <input type="button" id="editButton" value="Edit" style="display:
none;">
  </form>

  <h2>Table</h2>

  <table id="dataTable">
    <tr>
      <th>Name</th>
      <th>Email</th>
      <th>Date of Birth</th>
      <th>Action</th>
    </tr>
  </table>

  <script>
    var selectedRow = null;

    document.getElementById("myForm").addEventListener("submit",
function(event) {
      event.preventDefault();

      var name = document.getElementById("name").value;
      var email = document.getElementById("email").value;
      var dob = document.getElementById("dob").value;
      if (!validateForm(name, email, dob)) {
        return;
      }

      if (selectedRow === null) {
        insertData(name, email, dob);
      } else {
        updateData(name, email, dob);
      }

      resetForm();
    });

    function validateForm(name, email, dob) {
      if (!name || !email || !dob) {
        alert("Please fill in all fields.");
        return false;
      }
      return true;
    }

    function insertData(name, email, dob) {
      var table = document.getElementById("dataTable");
      var newRow = table.insertRow(1);

      var cell1 = newRow.insertCell(0);
      cell1.innerHTML = name;

      var cell2 = newRow.insertCell(1);
      cell2.innerHTML = email;

      var cell3 = newRow.insertCell(2);
      cell3.innerHTML = dob;

      var cell4 = newRow.insertCell(3);
      cell4.innerHTML = '<input type="button" value="Edit"
onclick="editData(this)">' + '   ' +
                        '<input type="button" value="Delete"
onclick="deleteData(this)">';
    }
```

```
function updateData(name, email, dob) {
  selectedRow.cells[0].innerHTML = name;
  selectedRow.cells[1].innerHTML = email;
  selectedRow.cells[2].innerHTML = dob;

  var submitButton = document.querySelector("input[type=submit]");
  submitButton.value = "Submit";
  selectedRow = null;
}

function resetForm() {
  document.getElementById("myForm").reset();

  var submitButton = document.querySelector("input[type=submit]");
  submitButton.value = "Submit";

  var editButton = document.getElementById("editButton");
  editButton.style.display = "none";
}

function editData(btn) {
  console.log("btn",btn.parentNode,btn, btn.parentNode.parentNode)
  var row = btn.parentNode.parentNode;
  selectedRow = row;

  document.getElementById("name").value = row.cells[0].innerHTML;
  document.getElementById("email").value = row.cells[1].innerHTML;
  document.getElementById("dob").value = row.cells[2].innerHTML;

  var submitButton = document.querySelector("input[type=submit]");
  submitButton.value = "Update";

  // var editButton = document.getElementById("editButton");
  // editButton.style.display = "inline-block";
}

function deleteData(btn) {
  var row = btn.parentNode.parentNode;
```

```
      var confirmation = confirm("Are you sure you want to delete this
entry?");

      if (confirmation) {
        row.parentNode.removeChild(row);
        resetForm();
      }
    }
  </script>
</body>
</html>
```