DOM_Role.txt

1. Explain the DOM and its role in Web development.

Ans: The Document Object Model (DOM) is a programming interface for web documents, representing the structure of an HTML or XML document as a tree of objects. Each element, attribute, and text is a node in this tree, allowing developers to access and manipulate content, structure, and styling via scripting languages like JavaScript. The DOM's dynamic nature enables the creation of interactive web pages by modifying elements in response to user actions or events. In web development, it plays a crucial role in enabling dynamic updates, form validation, animations, and real-time data rendering without reloading the entire page.

EventDelegation.txt

2. Explain the concept of event delegation and provide a scenario where it is beneficial.

Ans: Event delegation is a technique in JavaScript where a single event listener is used on a parent element to manage events for multiple child elements. Instead of attaching individual event listeners to each child, the event bubbles up to the parent, where the listener detects it and determines the source of the event using the event.target property. This approach reduces memory usage and improves performance, especially when handling many elements.

A beneficial scenario is managing click events on a list of items. Instead of adding separate click listeners to each item, a single listener on the parent list can detect clicks on any child item, making it easier to manage dynamic changes like adding or removing items.

EventBubbling.txt

3. Explain the concept of Event Bubbling in the DOM.

Ans: Event bubbling in the DOM is a process where an event triggered on a child element first executes its event handlers, then "bubbles up" to execute handlers on its parent elements, up to the root. For example, if a button inside a div is clicked, the button’s click handler runs first, followed by the div’s handler, and so on up the hierarchy. This allows higher-level elements to handle events occurring within their child elements. Event bubbling is the default behavior, but it can be stopped using event.stopPropagation() to prevent the event from reaching parent elements.

EventListeners.txt

4. Explain the purpose of the addEventListener method in JavaScript and how it facilitates event handling in the DOM.

Ans: The addEventListener method in JavaScript attaches event handlers to DOM elements, allowing the execution of a specified function when an event occurs, such as a click or keypress. It provides flexibility by enabling multiple event listeners for the same event type on a single element. The method supports various options, including capturing or bubbling phases and once-only execution. By separating JavaScript code from HTML, addEventListener promotes cleaner code and enhances maintainability. It also allows the dynamic addition and removal of event handlers, making it essential for creating interactive and responsive web applications.

index.html

```html
<!-- Alert Function Demo -->

<!-- 5. Create an HTML page with a button. Use JavaScript to display an alert when the button is clicked -->

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Alert Function Demo</title>
</head>

<body>

    <h1>Alert Function Demo</h1>

    <button id="click-btn">Click Me!</button>

    <script src="script.js"></script>

</body>

</html>
```

script.js

```
const clickBtn = document.getElementById('click-btn');

clickBtn.addEventListener('click', () => {

    alert('Hello World!');
});
```

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Simple Image Carousel</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>
    <div class="carousel">

        <div class="carousel-images">

            <img src="images/web-1.jpg" alt="Image 1">

            <img src="images/web-2.jpg" alt="Image 2">

            <img src="images/web-3.jpg" alt="Image 3">

        </div>

        <button class="prev-button">&#10094;</button>

        <button class="next-button">&#10095;</button>

    </div>

    <script src="script.js"></script>

</body>

</html>
```

style.css

```css
body {
    font-family: Arial, sans-serif;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
    margin: 0;
}

.carousel {
    position: relative;
    width: 80%;
    max-width: 600px;
    overflow: hidden;
}

.carousel-images {
    display: flex;
    transition: transform 0.5s ease-in-out;
}

.carousel-images img {
    width: 100%;
    height: auto;
    flex-shrink: 0;
}

.prev-button,
.next-button {
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    background-color: rgba(0, 0, 0, 0.5);
    color: white
    border: none;
```

```
    font-size: 24px;
    cursor: pointer;
    padding: 10px;
    user-select: none;
}

.prev-button {
    left: 10px;
}

.next-button {
    right: 10px;
}
```

script.js

```
const carouselImages = document.querySelector('.carousel-images');

const images = document.querySelectorAll('.carousel-images img');

const prevButton = document.querySelector('.prev-button');

const nextButton = document.querySelector('.next-button');

let currentIndex = 0;

function showImage(index) {

    const offset = -index * 100;

    carouselImages.style.transform = `translateX(${offset}%)`;
}

nextButton.addEventListener('click', () => {

    currentIndex = (currentIndex + 1) % images.length;

    showImage(currentIndex);
});

prevButton.addEventListener('click', () => {

    currentIndex = (currentIndex - 1 + images.length) % images.length;

    showImage(currentIndex);
});

// Show the first image initially

showImage(currentIndex);
```

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dynamic Dropdown Menu</title>
</head>

<body>

    <ul class="menu">

        <li class="menu-item">Home</li>

        <li class="menu-item"> Services

            <ul class="dropdown">

                <li class="dropdown-item">Sub Menu 1</li>
                <li class="dropdown-item">Sub Menu 2</li>
                <li class="dropdown-item">Sub Menu 3</li>

            </ul>

        </li>

        <li class="menu-item">Contact</li>

    </ul>

    <script src="script.js"></script>

</body>

</html>
```

script.js

```
// Get all menu items with dropdowns

const menuItems = document.querySelectorAll('.menu-item');

// Add hover event listeners to toggle dropdown visibility

menuItems.forEach(item => {

    const dropdown = item.querySelector('.dropdown');

    if (dropdown) {

        item.addEventListener('mouseenter', () => {

            dropdown.style.display = 'block';

        });

        item.addEventListener('mouseleave', () => {

            dropdown.style.display = 'none';

        });
    }
});
```

index.html

```html
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Dynamic Shopping List</title>
    <link rel="stylesheet" href="style.css">
</head>

<body>

    <div class="container">

        <h1>Shopping List</h1>

        <input type="text" id="itemInput" placeholder="Add a new item">

        <button id="addItemButton">Add Item</button>

        <ul id="shoppingList"></ul>

    </div>

    <script src="script.js"></script>

</body>

</html>
```

script.js

```
document.addEventListener('DOMContentLoaded', () => {

    // Get DOM elements

    const addItemButton = document.getElementById('addItemButton');

    const itemInput = document.getElementById('itemInput');

    const shoppingList = document.getElementById('shoppingList');

    // Add item to the list

    addItemButton.addEventListener('click', addItem);

    itemInput.addEventListener('keypress', function (event) {

        // If the enter key was pressed then run the addItem function

        if (event.key === 'Enter') {

            addItem();
        }
    });

    function addItem() {

        // Validate input

        const itemText = itemInput.value.trim();

        // Check if input is not empty

        if (itemText !== '') {

            // Create list item

```

```
            const listItem = document.createElement('li');

            // Add text to list item

            listItem.textContent = itemText;

            // Create delete button

            const deleteButton = document.createElement('button');

            // Add text to delete button

            deleteButton.textContent = 'Delete';

            // Add delete button to list item

            deleteButton.className = 'delete-button';

            // Remove item from the list when delete button is clicked

            deleteButton.addEventListener('click', () => {

                shoppingList.removeChild(listItem);
            });

            listItem.appendChild(deleteButton);

            shoppingList.appendChild(listItem);

            // Clear input field and focus

            itemInput.value = '';

            itemInput.focus();
        }
    }
});
```