Universidad Tecnológica Metropolitana

Nombre de la asignatura Aplicaciones WEB

Nombre de la maestra: Ruth

Alumnos:

Joel Arcángel Canul Chan

Cuatrimestre: 3ro

Grupo: A

Fecha: 18/06/2024

Nombre de la actividad:

DOM

Parcial: Segundo parcial

Introduccion

El Modelo de Objetos del Documento (DOM, por sus siglas en inglés) es una interfaz de programación esencial para los documentos HTML y XML. Su función principal es proporcionar una representación estructurada del documento, permitiendo a los desarrolladores web acceder y manipular su contenido, estructura y estilo de manera programática.

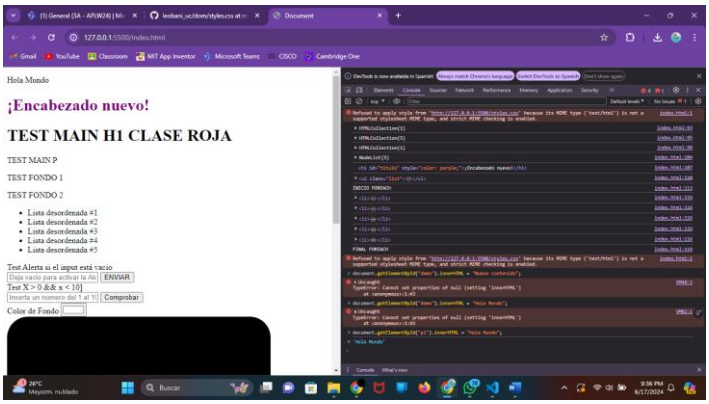
El DOM organiza el documento en una jerarquía de nodos, donde cada nodo representa un componente del documento, como elementos, atributos, texto o comentarios. Esta jerarquía facilita la interacción y modificación del documento a través de lenguajes de programación, siendo JavaScript el más comúnmente utilizado.

Una de las características más importantes del DOM es su capacidad para definir propiedades y métodos que permiten la manipulación de estos nodos. Por ejemplo, el método getElementById permite seleccionar un elemento específico del documento utilizando su ID, mientras que la propiedad innerHTML permite cambiar el contenido de dicho elemento.

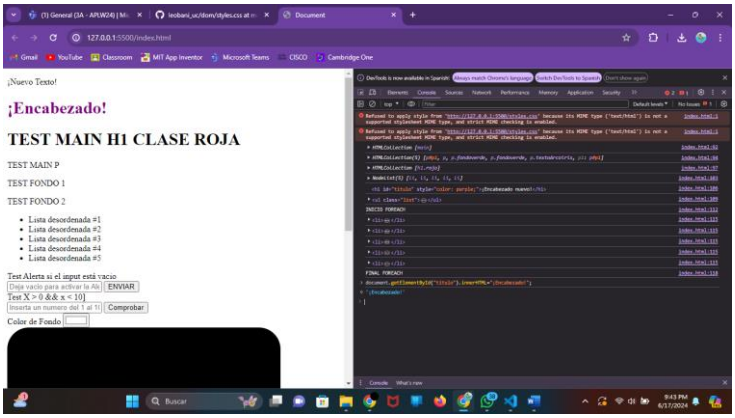
Además de manipular la estructura y el contenido, el DOM también facilita la gestión de eventos. Los eventos, como clics del ratón, entradas de teclado y movimientos del cursor, pueden ser manejados mediante el DOM para crear interactividad en las páginas web. Esto permite que las páginas web sean dinámicas y reaccionen a las acciones del usuario sin necesidad de recargar la página completa.

El DOM es fundamental para la creación de aplicaciones web modernas y dinámicas. Permite la actualización del contenido en tiempo real, mejora la interactividad y ofrece una experiencia de usuario más atractiva. A través del DOM, los desarrolladores pueden transformar simples documentos estáticos en aplicaciones web interactivas y completas.

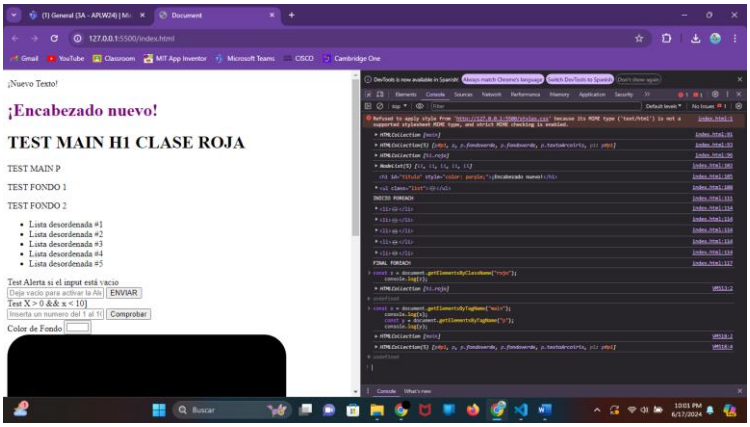
1. Uso del método getElementById y de la propiedad innerHTML.
   A. Cambia el contenido (el innerHTML) del elemento <p> con id=”p1”.



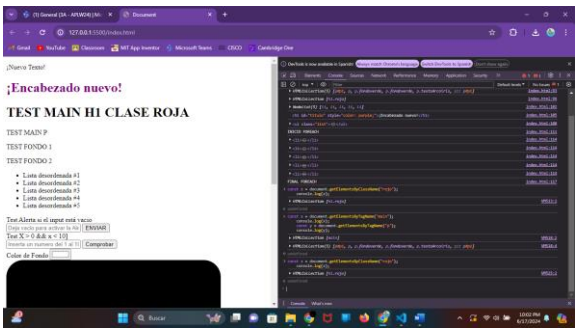
   B. Cambia el contenido de un elemento <h1>.



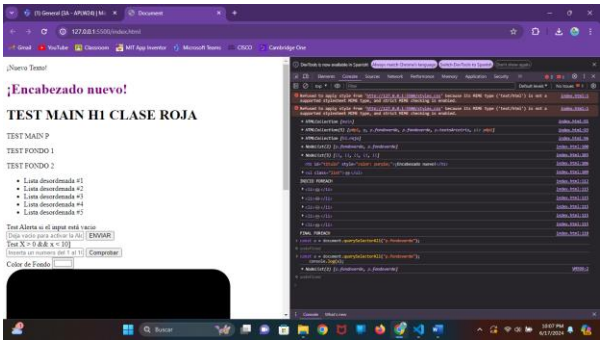
2. Búsqueda de elementos HTML
   A. Encuentra el elemento con id=”main” y luego encuentra todos los elementos <p>dentro del main.



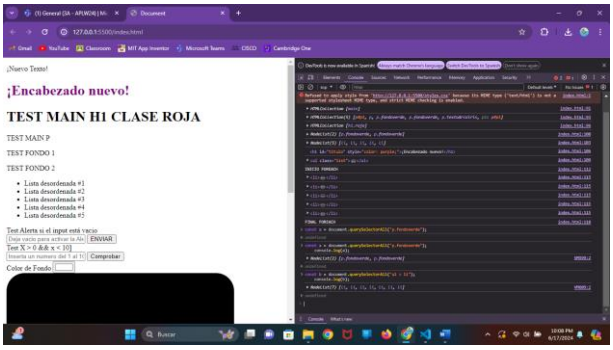
3. Encontrar elementos por nombre de clases.
   A. Si desea encontrar todos los elementos con el mismo nombre de clase, use getElementsByClassName().



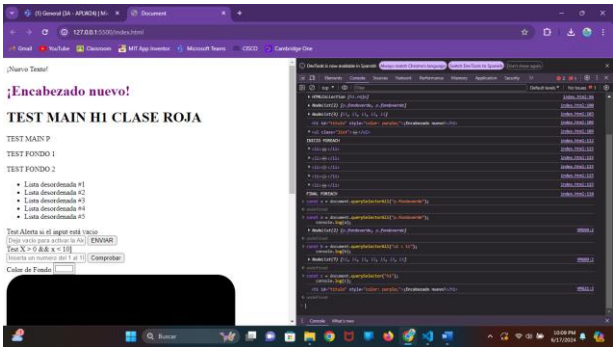
4. Búsqueda de elementos HTML mediante selectores de CSS
   A. Devuelve una lista de todos los elementos<p>con class =”intro”.



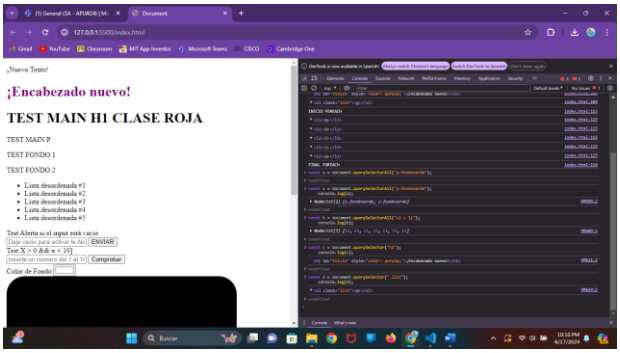
   B. Encuentra los elementos hijos de <ul>, para ello debes crear una lista desordenada con al menos 5 elementos dentro de la lista.



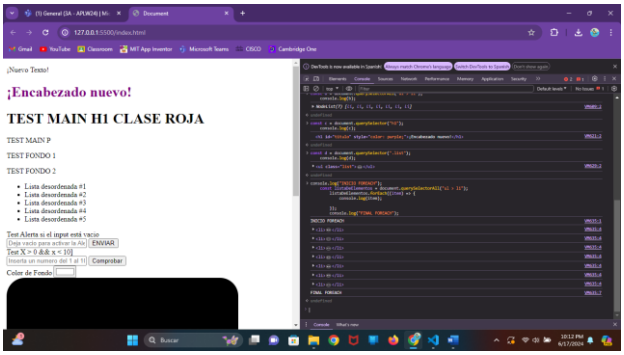
   C. Encontrar en la consola e imprimir el elemento<h1>.



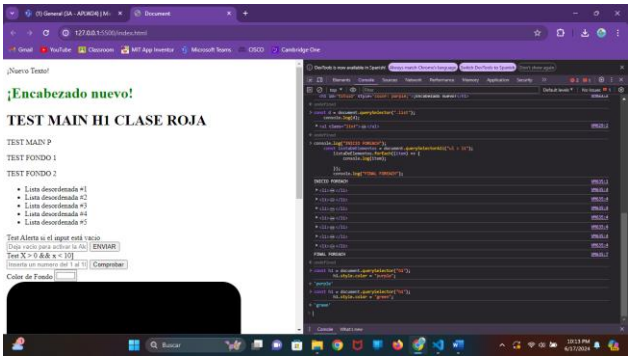
D. Encontrar la clase listusando querySelector().



E. Imprimir los elementos haciendo uso del ciclo forEach()para iterar sobre la NodeList e imprimir cada uno de los elementos.

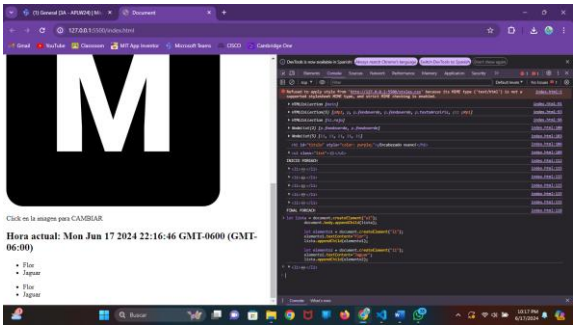


F. Uso de la propiedad style para cambiar estilos en línea CSS.

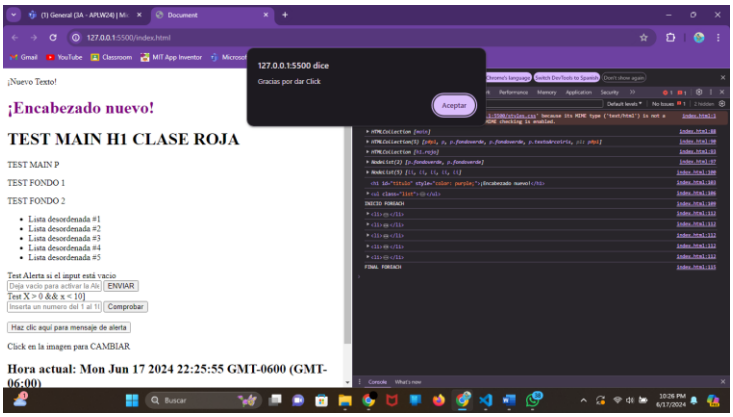


5. Búsqueda de elementos HTML por colecciones de objetos HTML.
   A. Encuentra el elemento de formulario con id=”frm1”, en la colección de formularios, y muestra todos los valores de los elementos.

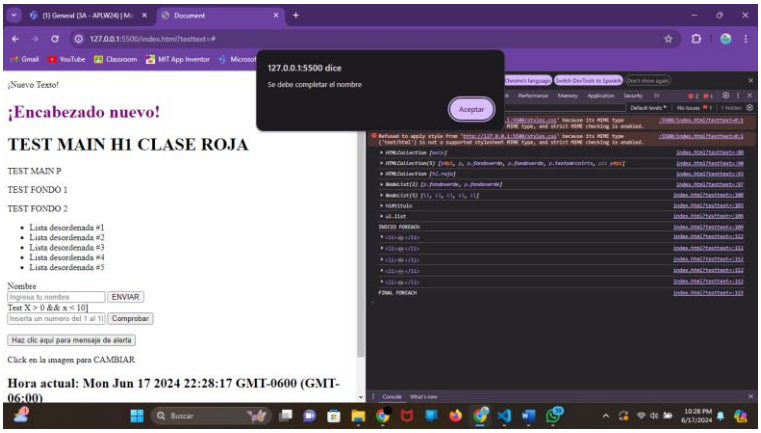
6. Agregar nuevos elementos al documento HTML.
   A. Agregar elementos al árbol del DOM usando losmétodosdocument.createElement(), appendChild()y haciendo uso de la propiedad textContent.



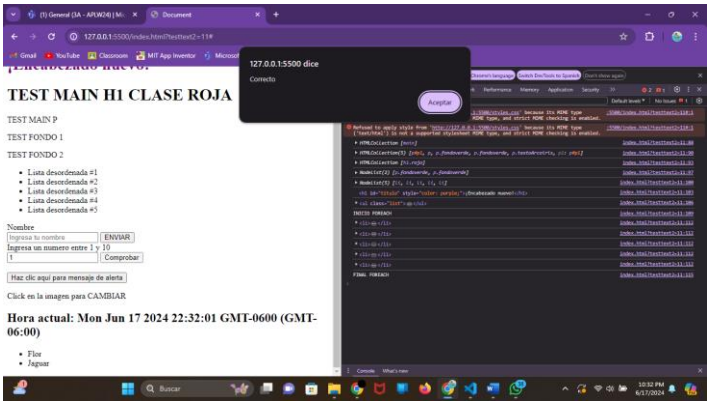
7. Usando el método addEventListener() para escuchar eventos en la página.
   A. Crea un botón en el documento html, donde al hacer click aparezca un mensaje de alerta.

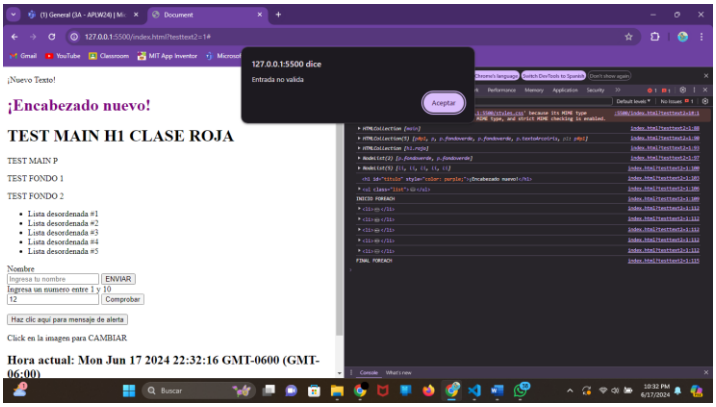


8. Validación de formulario.
   A. Si un campo de formulario (name) esta vacío, se muestra una alerta con un mensaje y devuelve falso para evitar que se envíe y se redireccione a otra página.

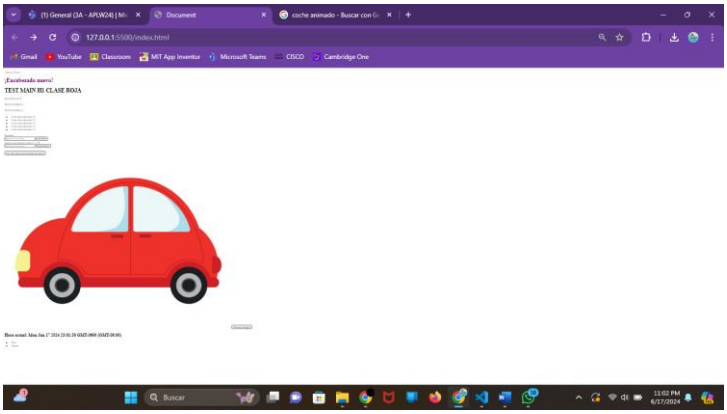


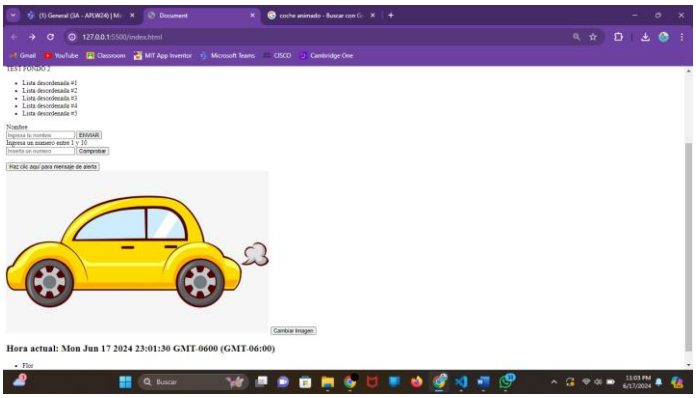
B. Valida la entrada numérica que valide que la entrada sea numérica y que este en un rango entre 1 y 10. Para ello crea un input con id=”numero”, un botón de type=”button”y que contenga el atributo onclick donde se le asigne el valor de la función. Crea también una etiqueta <p> con id=”demo”.



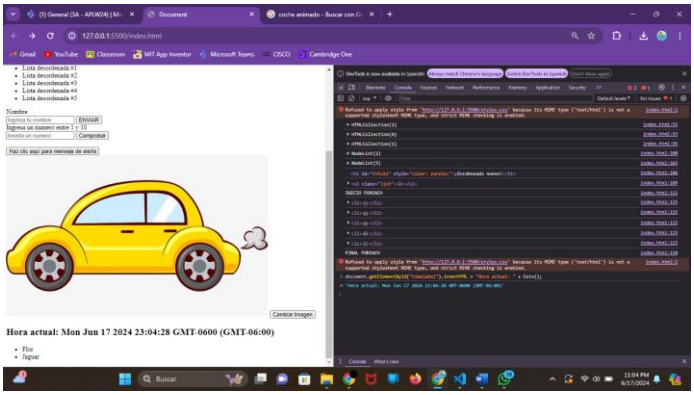


9. Cambiar el valor de un atributo
   A. Cambiar el valor de un atributo src de un elemento <img>.





10. Contenido dinámico
    A. Agregar la hora actual a una etiqueta con id=”demo”.



CODIGO

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <p id="p1">¡Hola Mundo!</p>
    <h1 id="titulo">Encabezado Viejo</h1>
    <main>
        <h1 class="rojo">H1 CLASS RED</h1>
        <p>TEST MAIN P</p>
    </main>
    <p class="fondoverde">TEST FONDO 1</p>
    <p class="fondoverde">TEST FONDO 2</p>
    <ul class="list">
        <li>Lista desordenada #1</li>
        <li>Lista desordenada #2</li>
        <li>Lista desordenada #3</li>
        <li>Lista desordenada #4</li>
        <li>Lista desordenada #5</li>
    </ul>
    <form action="#" method="get" id="form1">
        <label for="testtext">Nombre</label><br>
        <input type="text" name="testtext" id="testtext"
placeholder="Ingresa tu nombre">
        <input type="submit" value="ENVIAR" id="btnSubmit">
    </form>
    <form action="#" method="get" id="form2">
        <label for="testtext2">Ingresa un numero entre 1 y 10</label> <br>
        <input type="text" name="testtext2" id="testtext2"
placeholder="Inserta un numero">
        <input type="submit" value="Comprobar" id="btnSubmit2">
    </form>
    <br>
    <button id="btn">Haz clic aquí para mensaje de alerta</button>
    <br>
    <img id="miImagen" src="imagen_an.jpg" alt="Imagen Antigua">
    <button id="btn2">Cambiar Imagen</button>
    <br>
```

```
    <h2 id="timelabel">HORA ACTUAL:</h2>
</body>

<script>
    document.addEventListener("DOMContentLoaded", () => {
    /*Busqueda por nombre de etiqueta*/
    document.getElementById("p1").innerHTML="¡Nuevo Texto!";
    document.getElementById("titulo").innerHTML="¡Encabezado nuevo!";
    /* Busqueda por nombre de clases*/
    const x = document.getElementsByTagName("main");
    console.log(x);
    const y = document.getElementsByTagName("p");
    console.log(y);
    /*Encontrar por nombrede clases*/
    const z = document.getElementsByClassName("rojo");
    console.log(z);
    /*Busqueda de elementos HTML mediante selectores de CSS*/
    //Devuelve solo donde TagName(p) y Class(fondoverde)
    const a = document.querySelectorAll("p.fondoverde");
    console.log(a);
    //Devuelve todos los HIJOS de una lista desordenada
    const b = document.querySelectorAll("ul > li");
    console.log(b);
    //Imprime el primer elemento H1
    const c = document.querySelector("h1");
    console.log(c);
    //Encuentra un elemento con la clase list
    const d = document.querySelector(".list");
    console.log(d);

    //Iterar los elementos <li> usando el ciclo ForEach()
        console.log("INICIO FOREACH");
    const listaDeElementos = document.querySelectorAll("ul > li");
        listaDeElementos.forEach((item) => {
            console.log(item);

        });
        console.log("FINAL FOREACH");

        //CAMBIAR ESTILOS DE CSS CON JS
        const h1 = document.querySelector("h1");
        h1.style.color = "purple";

        //Agregar nuevos elementos HTML.
        let lista = document.createElement("ul");
```

```
document.body.appendChild(lista);

let elemento1 = document.createElement("li");
elemento1.textContent="Flor";
lista.appendChild(elemento1);

let elemento2 = document.createElement("li");
elemento2.textContent="Jaguar";
lista.appendChild(elemento2);

//BOTON CLICK
const button = document.getElementById("btn");
button.addEventListener("click", () => {
    alert("Gracias por dar Click");
});

//Validación input vacio
let botonSubmit = document.getElementById('btnSubmit');
botonSubmit.addEventListener("click", () =>{
    let x = document.forms["form1"]["testtext"].value;
    if(x === ""){
        alert("Se debe completar el nombre");
        return false;
    }
});

//Validación numero
let botonSubmit2 = document.getElementById('btnSubmit2');
botonSubmit2.addEventListener("click", () =>{
    let x = document.forms["form2"]["testtext2"].value
    if (isNaN(x) || x < 1 || x > 10){
        alert("Entrada no valida");
        return false;
    }else{
        alert("Correcto");
        return false
    }
    return false;
})
 // Cambiar imagen
 document.getElementById("btn2").addEventListener("click", () => {
    document.getElementById("miImagen").src = "car.jpg";
});
```

```
        //Poner fecha actual
        document.getElementById("timelabel").innerHTML = "Hora actual: " +
Date();



    });
</script>
</html>
```

Conclusión

El código proporcionado demuestra cómo JavaScript interactúa dinámicamente con elementos HTML a través del Document Object Model (DOM),

Se utilizan métodos como `getElementById`, `getElementsByTagName`, y `getElementsByClassName` para seleccionar y manipular elementos específicos por su ID, etiqueta o clase. Esto permite modificar el contenido de manera dinámica, como cambiar el texto de un párrafo o un encabezado (`h1`).

Además, se muestra cómo JavaScript puede ajustar los estilos de los elementos utilizando la propiedad `.style`, como cambiar el color de un encabezado (`h1`) a morado directamente desde el script.

El código también incluye la creación dinámica de nuevos elementos HTML con `createElement` y su inserción en el documento usando `appendChild`. Esto facilita la generación de contenido en tiempo real, como listas desordenadas (`ul`) con elementos de lista (`li`).

Se implementan funciones de manejo de eventos con `addEventListener` para asignar acciones específicas a eventos como clics de botón (`click`). Por ejemplo, un botón puede cambiar una imagen al ser clicado, proporcionando una interactividad instantánea y eficaz.

Finalmente, se aborda la validación de formularios mediante funciones que aseguran que los datos introducidos sean válidos antes de ser procesados. Esto incluye verificar campos de texto no vacíos o asegurarse de que un número esté dentro de un rango específico, mejorando la experiencia del usuario al garantizar la integridad de los datos enviados.

En resumen, el código muestra cómo JavaScript puede dinamizar una página web al interactuar directamente con su contenido HTML, proporcionando una experiencia más interactiva y adaptativa para los usuarios.

Referencias

docs, m. w. (s.f.). Obtenido de https://developer.mozilla.org/es/docs/Web/API/Document_Object_Model/Introduction

manz.dev. (s.f.). Obtenido de https://lenguajejs.com/javascript/dom/que-es/