PRACTICA °2

Nombre: David Hugo Davila Cruz

¿Qué es el DOM y cómo se relaciona con HTML?

```
<!DOCTYPE html>
<html>
<head>
  <title>Ejemplo</title>
</head>
<body>
  <h1>Hola, mundo!</h1>
  <p>Este es un párrafo.</p>
</body>
</html>
```

2.-

3.- ¿Para qué sirve addEventListener? Proporciona un ejemplo.

Un addEventListener sirve para registrar un "listener" de eventos, es decir, una función que se ejecutará cuando ocurra un evento específico en un elemento HTML, como un clic de ratón o un cambio de entrada. Permite que un elemento responda a eventos sin necesidad de modificar el código HTML.

Ejemplo

<!DOCTYPE html>

<html lang="es">

<head>

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ejemplo de addEventListener</title>
</head>
<body>

<button id="miBoton">Haz clic aquí</button>
<p id="mensaje"></p>

<script>
    const boton = document.getElementById('miBoton');
    const mensaje = document.getElementById('mensaje');

    boton.addEventListener('click', function() {
        mensaje.textContent = '¡Has hecho clic en el botón!';
    });
</script>

</body>
</html>
```

4.- ¿Qué métodos del DOM se usan para capturar valores de un formulario?

Para capturar valores de un formulario en JavaScript, se utilizan principalmente los siguientes métodos del DOM:

document.getElementById(),document.querySelector(),document.querySelectorAll()

Form.elements-

5.- Explica cómo prevenir el envío por defecto de un formulario con JavaScript.

Para prevenir el envío por defecto de un formulario en JavaScript, se puede utilizar el método preventDefault() del objeto del evento. Esto es útil cuando deseas realizar validaciones, manipular datos antes de enviarlos o manejar el envío de manera personalizada.

6.- ¿Qué es el "almacenamiento en memoria" y en qué se diferencia de `localStorage` ?

El "almacenamiento en memoria" (también conocido como almacenamiento "in-memory") se refiere a la forma en que una aplicación almacena datos en la memoria RAM del dispositivo, mientras que localStorage es un método de almacenamiento persistente que se utiliza para guardar datos en el navegador web del usuario. La principal diferencia es que el almacenamiento en memoria se pierde cuando se cierra el navegador o la página se reinicia, mientras que localStorage persiste hasta que se borra manualmente.

2. Análisis de Código

Dado el siguiente código:

```
<button id="btn">Haz clic</button>
<p id="mensaje"></p>
document.getElementById("btn").addEventListener("click", () => {
 document.getElementById("mensaje").textContent = "¡Botón presionado!";
});
```

Preguntas:

- ¿Qué hace el código?
- ¿Qué pasaría si cambiamos `textContent` por `innerHTML`?

El código proporciona una funcionalidad que hace una interacción con el sitio web donde el código hace una interacción con HTML y con java script y el html tiene un botón que cumple una función con un <button> y donde un texto y un párrafo vacio <p> donde se mostra un mensaje y donde java script donde el botón con id= btn y va agregando un evento un clic en un botón y se cambia un contenido de párrafo con id="mensaje" a: **"¡Botón presionado!"** usando textContent.

3. Análisis de Código

Dado el siguiente formulario:

```
<form id="form-usuario">
 <input type="text" id="nombre" placeholder="Nombre">
 <button type="submit">Guardar</button>
</form>
<ul id="lista-usuarios"></ul>
const usuarios = []; // Array para simular persistencia

document.getElementById("form-usuario").addEventListener("submit", (e) => {
 e.preventDefault(); // ¿Por qué es importante esta línea?
 const nombre = document.getElementById("nombre").value;
 usuarios.push(nombre); // Almacena en memoria
 actualizarListaUsuarios();
});

function actualizarListaUsuarios() {
 const lista = document.getElementById("lista-usuarios");
 lista.innerHTML = usuarios.map(user => `<li>${user}</li>`).join("");
}
```

Preguntas:

- ¿Qué hace el código al enviar el formulario?

Al enviar el formulario, el evento de submit se activa. La línea e.preventDefault(); evita que la página se recargue, permitiendo que el código JavaScript continúe ejecutándose. Luego, se obtiene el valor del campo de texto (nombre) y se añade al array usuarios. Finalmente, se

llama a la función actualizarListaUsuarios() para actualizar la lista visible en la página con los nombres almacenados.

- ¿Cómo se simula la "persistencia de datos" aquí?

La "persistencia de datos" se simula mediante el uso de un array (usuarios) que almacena los nombres ingresados en memoria. Sin embargo, esta persistencia es temporal, ya que los datos se perderán si se recarga la página. En un entorno real, se podría utilizar almacenamiento local (como localStorage) o una base de datos para mantener los datos incluso después de recargar la página