PARTE 1 – Parte Teórica

1. Preguntas Conceptuales

a) ¿Qué es el DOM?
El DOM (Document Object Model) es básicamente como una versión en forma de árbol del HTML de una página. Cada cosa (etiqueta, texto, atributos) es como una "ramita" o nodo. Gracias al DOM, podemos usar JavaScript para cambiar cosas de la página mientras está cargada, como el texto, los estilos o agregar elementos nuevos.

b) Diferencias entre métodos y propiedades comunes del DOM

- document.getElementById("id"): Esto se usa para agarrar un elemento por su ID. Solo te devuelve uno porque el ID es único.
- document.querySelector("selector"): Es más flexible, porque usa los mismos selectores que el CSS (como .clase, #id, div p, etc.). También te da solo el primer elemento que coincida.
- textContent: Solo mete o saca texto. Si hay etiquetas HTML, las muestra como texto tal cual.
- innerHTML: Este sí interpreta HTML. Por ejemplo, si metes un hola, lo muestra en negrita.

c) ¿Para qué sirve addEventListener?
Este método sirve para decirle al navegador: "Oye, si pasa tal cosa (como que alguien haga clic), ejecuta esta función". Ejemplo:

```
document.getElementById("miBoton").addEventListener("click", () => {
  alert("¡Clic detectado!");
});
```

Cuando le das clic al botón, aparece una alerta.

d) ¿Cómo saco el valor de un formulario con JavaScript?

- Con .value, usando cosas como getElementById("campo").value o querySelector("selector").value.
- También puedes usar FormData, que es como una forma de agarrar todos los datos del formulario de una vez:

```
const form = document.getElementById("miFormulario");
const datos = new FormData(form);
console.log(datos.get("nombre"));
```

e) ¿Para qué sirve e.preventDefault()?
Evita que el formulario se envíe y recargue la página. Básicamente te da chance de manejar el formulario a tu manera con JavaScript.

f) ¿Qué diferencias hay entre guardar datos en variables y en localStorage?

- Las **variables** solo guardan los datos mientras la página está abierta. Si la recargas, ¡adiós datos!
- **localStorage** es como una mini base de datos del navegador. Guarda info que se queda ahí incluso si cierras la pestaña o el navegador.

2. Análisis de Código

```
<button id="btn">Haz clic</button>
<p id="mensaje"></p>

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

¿Qué pasa aquí?
Cuando haces clic en el botón, el texto dentro del <p> cambia y aparece "¡Botón presionado!".

¿Y si uso innerHTML en vez de textContent?
Funciona igual si solo estás mostrando texto. Pero si le metes etiquetas HTML, innerHTML las interpreta.
Ejemplo:

```
mensaje.innerHTML = "<b>¡Botón presionado!</b>";
```

Esto haría que el texto salga en negrita.

3. Análisis de Código

```
<form id="form-usuario">
  <input type="text" id="nombre" placeholder="Nombre">
  <button type="submit">Guardar</button>
</form>
<ul id="lista-usuarios"></ul>

<script>
const usuarios = [];

document.getElementById("form-usuario").addEventListener("submit", (e) => {
  e.preventDefault();
  const nombre = document.getElementById("nombre").value;
  usuarios.push(nombre);
  actualizarListaUsuarios();
});

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

</script>

¿Qué hace este código?

- Evita que se recargue la página cuando se manda el formulario.
- Agarra el nombre que escribiste.
- Lo mete en un arreglo llamado usuarios.
- Luego actualiza la lista (<ul>) para mostrar todos los nombres que se han guardado.

¿Los datos se guardan para siempre?
No. El arreglo usuarios solo vive mientras la página está abierta. Si la recargas, se borra. Si quieres que no se pierdan, tendrías que usar localStorage.