Estrategias de almacenamiento en localStorage y sessionStorage en CSR

Objetivo

Que el estudiante comprenda, identifique y aplique las estrategias de almacenamiento en el cliente dentro de aplicaciones CSR (Client Side Rendering), usando localStorage y sessionStorage en JavaScript.

1. Introducción al CSR (Client Side Rendering)

En una aplicación CSR, toda la lógica de interfaz y actualización de datos ocurre en el navegador del usuario, no en el servidor. Por eso, es importante poder guardar información localmente, sin tener que comunicarse con el servidor constantemente. Aquí entran las API de almacenamiento web: localStorage y sessionStorage.

2. ¿Qué son localStorage y sessionStorage?

Característica	localStorage	sessionStorage	Uso típico
Persistencia	Permanente (hasta que se borre manualmente)	Temporal (se borra al cerrar la pestaña o ventana)	Configuraciones o sesión persistente / Datos temporales
Alcance	Global para todo el dominio	Solo la pestaña actual	
Capacidad	~5MB por dominio	~5MB por dominio	

3. Estrategias de almacenamiento en CSR

A) Estrategias con localStorage

Se usa cuando los datos deben mantenerse entre sesiones. Estrategias comunes:

1. Guardar configuraciones de usuario (tema, idioma):

- localStorage.setItem("theme", "dark");
  const theme = localStorage.getItem("theme");

2. Mantener sesión del usuario (token o login):

- localStorage.setItem("token", "abc123xyz");

3. Cache de datos de API para evitar recargas:

- fetch("/api/datos")
  .then(res => res.json())
  .then(data => localStorage.setItem("cacheDatos", JSON.stringify(data)));

4. Guardar carrito de compras o preferencias persistentes:

- localStorage.setItem("carrito", JSON.stringify(productos));

B) Estrategias con sessionStorage

Se usa cuando los datos solo deben existir durante la sesión actual. Estrategias comunes:

1. Guardar progreso de formularios:

- sessionStorage.setItem("formData", JSON.stringify(datos));

2. Controlar la navegación (mostrar un mensaje solo una vez):

- if (!sessionStorage.getItem("introShown")) {
  alert("Bienvenido a la aplicación");
  sessionStorage.setItem("introShown", true);
  }
- 3. Guardar datos sensibles temporales: datos de verificación o filtros activos.

4. Buenas prácticas

✅ Usa JSON.stringify() y JSON.parse() para guardar objetos.
✅ Borra los datos cuando ya no se necesiten (removeItem() o clear()).
✅ Evita guardar información sensible (contraseñas, datos personales).
✅ Verifica si el navegador soporta la API:

- if (typeof(Storage) !== "undefined") {
  console.log("Compatible con almacenamiento local");
  }

5. Actividades prácticas

Ejercicio 1: Guardar tema del sitio usando localStorage.

Ejercicio 2: Guardar formulario temporalmente usando sessionStorage.

Ejercicio 3: Limpiar almacenamiento y observar el resultado.

6. Evaluación

1. ¿Cuál es la diferencia principal entre localStorage y sessionStorage?

2. ¿Qué tipo de datos no se recomienda guardar en localStorage?

3. ¿Qué función usamos para eliminar un elemento almacenado?

4. ¿Por qué son importantes estas estrategias en una aplicación CSR?

7. Conclusión

El almacenamiento en el cliente permite que las aplicaciones CSR sean más rápidas e interactivas, reduzcan la carga en el servidor y mantengan información relevante para el usuario sin depender de una base de datos. El uso correcto de localStorage y sessionStorage mejora tanto la experiencia de usuario como la eficiencia de las aplicaciones modernas.