FRONT-END JS

Clase nº 13 | LocalStorage, SessionStorage y Carrito de Compras

¡Les damos la bienvenida! 🚀

🎥 Vamos a comenzar a grabar la clase.

Índice

1

13. LocalStorage, SessionStorage y Carrito de Compras

- Introducción a LocalStorage y SessionStorage
- Diferencias entre LocalStorage y SessionStorage
- Implementación de un carrito de compras utilizando LocalStorage o SessionStorage

2

14. Asincronía

- Asincronía
- Consumo de API REST a través de fetch
- Procesamiento de los datos
- Incluir los datos consumidos y procesados por medio de fetch en nuestro proyecto

Introducción a LocalStorage y SessionStorage

LocalStorage vs SessionStorage

LocalStorage

- Es una API del navegador que permite guardar datos en pares clave-valor de manera persistente.
- Los datos almacenados permanecen incluso después de cerrar el navegador.

Ejemplo:

Podés almacenar configuraciones como el tema de la aplicación:

```
localStorage.setItem('tema', 'oscuro');
```

SessionStorage

- Almacena datos de forma temporal en el navegador.
- Los datos se eliminan al cerrar la pestaña o ventana del navegador.

Ejemplo:

Estado de un formulario en un proceso de compra:

```
sessionStorage.setItem('pasoActual', '2');
```

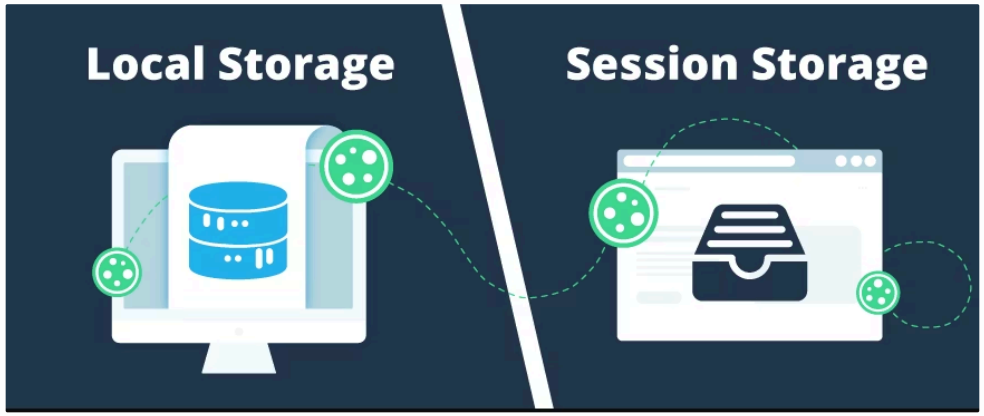
Diferencias:

LocalStorage

- Los datos se mantienen después de cerrar el navegador.
- Compartido entre todas las pestañas del mismo dominio.
- Ideal para almacenar preferencias o configuraciones persistentes.

SessionStorage

- Los datos se eliminan al cerrar la pestaña o ventana.
- Limitado a la pestaña o ventana donde se creó.
- Útil para datos temporales o que solo se necesiten durante la sesión.



Manipulación de LocalStorage

1 **Guardar:** para guardar datos en LocalStorage, usás el método `setItem()`:

```
localStorage.setItem('usuario', 'Pedro');
```

2 **Obtener:** para recuperar datos almacenados, usás el método `getItem()`:

```
let usuario = localStorage.getItem('usuario');
console.log(usuario); // Pedro
```

3 **Eliminar:** para eliminar un ítem específico, usás `removeItem()`:

```
localStorage.removeItem('usuario');
```

Manipulación de SessionStorage

---

1 **Guardar:** similar a LocalStorage, podés usar `setItem()` para guardar datos:

```
sessionStorage.setItem('usuario', 'Ana');
```

2

2 **Obtener:** podés obtener los datos con `getItem():`

```
let usuario = sessionStorage.getItem('usuario');
console.log(usuario); // Ana
```

3 **Eliminar:** para eliminar datos de SessionStorage, usás `removeItem():`

```
sessionStorage.removeItem('usuario');
```

Implementación de un Carrito de Compras con LocalStorage y SessionStorage

Estructura del Carrito de Compras en HTML

Podés implementar un carrito de compras usando una lista para mostrar los productos

```
<div id="carrito">
 <h2>Carrito de Compras</h2>
 <ul id="lista-carrito"></ul>
 <button id="vaciar-carrito">Vaciar Carrito</button>
</div>
```

Funcionalidades del Carrito con JavaScript

El carrito debe permitir agregar, eliminar productos y vaciar el carrito. Usaremos eventos y manipulación del DOM.

Agregar productos

Implementar función para añadir productos al carrito.

```
document.getElementById("boton-agregar").addEventListener("click", function () {
 let producto = { id: 1, nombre: "Producto 1", precio: 10 };
 let carrito = JSON.parse(localStorage.getItem("carrito")) || [];
 carrito.push(producto);
 localStorage.setItem("carrito", JSON.stringify(carrito));
 actualizarCarrito();
});
```

Funcionalidades del Carrito con JavaScript

Guardar productos

Guardá el estado del carrito en LocalStorage cada vez que se modifique

Recordà que JSON.stringify(carrito) convierte nuestro carrito, que es un array de objetos, en una cadena de texto. LocalStorage solo puede guardar strings, así que necesitamos convertir el array para poder almacenarlo. Cuando queramos recuperarlo, usaremos JSON.parse() para convertir esa cadena de vuelta en un array."

```
localStorage.setItem('carrito', JSON.stringify(carrito));
```

Funcionalidades del Carrito con JavaScript

Actualización del carrito

Cada vez que el usuario añada o elimine productos, actualizá el DOM y el LocalStorage para reflejar cambios.

```
function actualizarCarrito() {
 var carrito = JSON.parse(localStorage.getItem('carrito')) || [];
 var listaCarrito = document.getElementById('lista-carrito');
 listaCarrito.innerHTML = '';
 for (var i = 0; i < carrito.length; i++) {
   var producto = carrito[i];
   var li = document.createElement('li');
   li.textContent = producto.nombre + ' - $' + producto.precio;
   listaCarrito.appendChild(li);
 }
}
```

Ejercicio Práctico: Carrito de Compras

---

1 Desarrollá un carrito que permita:

- Añadir productos.
- Eliminar productos.
- Guardar el estado del carrito usando LocalStorage.
- Mantener el carrito funcional incluso al recargar la página.

2 Pasos a seguir

- Crear la estructura HTML del carrito.

- Implementar funcionalidad de agregar productos con JS.
- Utilizar LocalStorage para guardar los productos.
- Mostrar el contenido del carrito al cargar la página.
- Implementar la opción de vaciar el carrito.

3 Implementación

- Usamos JSON.stringify() y JSON.parse() para manejar objetos complejos en LocalStorage.
- Nos aseguramos de actualizar tanto el DOM como LocalStorage cada vez que se añadan o eliminen productos.
- Usamos eventos click para capturar las interacciones del usuario.

⚙️ Ejercicios Prácticos

⚙️ Ejercicio Práctico #1

Optativo | No entregable

⚙️ Ejercicio Práctico #2

Guardar Preferencias de Usuario

Pasos a seguir:

Crear una función que guarde y recupere las preferencias de un usuario, con su nombre y el color de fondo preferido, utilizando **LocalStorage**.

1. La función debe permitir al usuario ingresar su nombre y seleccionar
2. Los
3. Cada vez que la página se recargue, las preferencias deben recuperarse de **LocalStorage** y aplicarse automáticamente (mostrar nombre del usuario y cambiar el color de fondo)

Tips claves:

- Uso de LocalStorage para almacenar el nombre y el color.
- Manipulación del DOM para aplicar los cambios de color de fondo.
- Uso de eventos como submit para guardar las

Optativo | No entregable

Carrito de Compras con Conteo de Productos

Pasos a seguir:

Crear un carrito de compras utilizando **LocalStorage**, que permita a los usuarios agregar productos y muestre la cantidad total de productos en el carrito.

1. Los productos deben tener un botón para agregar al carrito.
2. Al agregar un producto, se debe mostrar el número total de productos en el carrito, almacenándolo en **LocalStorage**.
3. Al recargar la página, el número total de productos debe recuperarse de **LocalStorage** y mostrarse correctamente.

Tips claves:

- Uso de **LocalStorage** para guardar el número total de productos en el carrito.
- Manipulación del **DOM** para actualizar el contador de productos en tiempo real.
- Utilización de eventos click para agregar productos.