Buenos Aires aprende
Agencia de Habilidades para el Futuro

Talento Tech

Front-End JS

Clase 13 - “ LocalStorage, SessionStorage y Carrito de Compras”

Buenos Aires Ciudad

¡Les damos la bienvenida!

 Vamos a comenzar a grabar la clase

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Talento Tech

Clase 12.

JS 4 - Dom y eventos

1. Manipulación del DOM
2. Definición, alcance y su importancia para operar sobre elementos HTML
3. Eventos en JS
4. Eventos: ¿Que son, para qué sirven y cuáles son los más comunes?
5. Escuchar un evento sobre el DOM

Clase 13.

JS 5 - LocalStorage, SessionStorage y Carrito de Compras

1. Introducción a LocalStorage y SessionStorage
2. Diferencias entre LocalStorage y SessionStorage
3. Implementación de un carrito de compras utilizando LocalStorage o SessionStorage

Clase 14.

JS 6 - Asincronía

1. Asincronía
2. Consumo de API REST a través de fetch
3. Procesamiento de los datos
4. Incluir los datos consumidos y procesados por medio de fetch en nuestro proyecto

LocalStorage

SessionStorage

LocalStorage y SessionStorage

¡Les damos la bienvenida a la clase de LocalStorage y SessionStorage! Hoy vamos a ver cómo podés guardar datos en el navegador de una manera súper simple, y lo mejor de todo: ¡permanentes o temporales, según lo que necesites!

Breve introducción al tema: ¿Te imaginás que un usuario pueda guardar sus preferencias en tu aplicación y que, aunque cierre el navegador, esas preferencias sigan ahí? De eso se trata LocalStorage, y de su primo, el SessionStorage, que hace lo mismo pero con una duración más corta.

¿Qué son LocalStorage y SessionStorage?

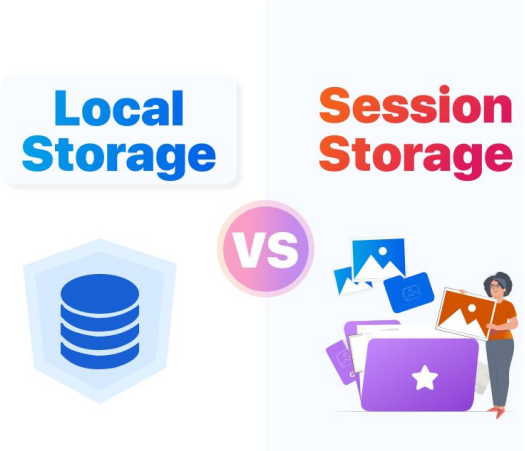
LocalStorage y SessionStorage son como pequeñas cajas que tiene el navegador, donde podés guardar datos clave-valor. La diferencia entre ellos es cuánto duran: LocalStorage es eterno, o al menos hasta que el usuario lo borre, y SessionStorage desaparece cuando cerrás la pestaña.

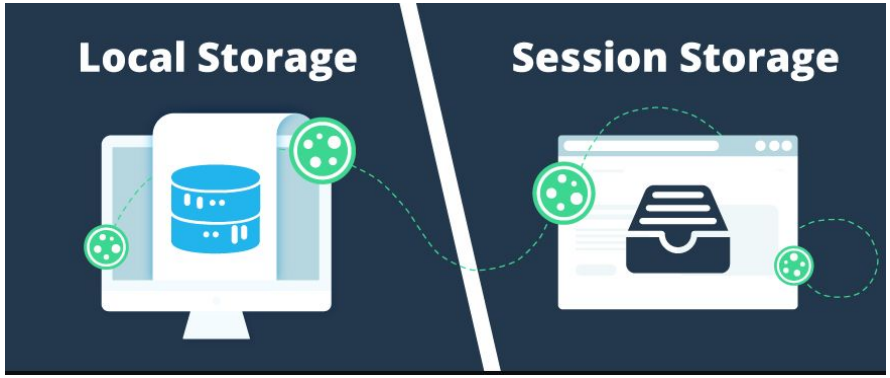
```
3
4
5   // Guardar datos en LocalStorage
6   localStorage.setItem('nombre', 'Pedro');
7
8
9   // Guardar datos en SessionStorage
10  sessionStorage.setItem('usuario', 'Ana');
11
```

- Fijate, con solo dos líneas de código ya podés empezar a guardar datos.

Diferencias entre LocalStorage y SessionStorage

Vamos a ver las diferencias clave de una manera sencilla. Imaginá que LocalStorage es como una libreta donde anotás cosas importantes y la guardás en un cajón. Aunque cierres la computadora, esos datos siguen ahí. SessionStorage, en cambio, es como escribir en un papelito que tirás a la basura cuando cerrás la pestaña del navegador.





Característica	LocalStorage	SessionStorage
Persistencia	Datos siguen al cerrar el navegador	Datos se eliminan al cerrar la pestaña
Alcance	Disponible en todas las pestañas del dominio	Solo en la pestaña actual

LocalStorage

¿Cómo guardar y obtener datos con LocalStorage?

- Ahora vamos a lo interesante, ¿cómo guardo algo? Supongamos que querés guardar el tema que eligió el usuario para tu página:

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

¿Y cómo lo recupero para que al volver a la página siga con su tema preferido?

```
let tema = localStorage.getItem('tema');
console.log(tema); // 'oscuro'
```

- ¡Facilísimo! Ahora el usuario no tiene que volver a cambiar el tema cada vez que entra.

Eliminar Datos en LocalStorage

Capaz que querés borrar datos, porque el usuario cambió de opinión o simplemente no los necesitás más. Podés hacerlo de dos maneras:

Eliminar un dato específico:

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

Eliminar todo de una vez:

```
localStorage.clear();
```

- Con estas funciones, limpiás todo lo que guardaste. ¡Adiós datos!


Session Storage

Guardar y Obtener Datos en SessionStorage

¿Cómo usar SessionStorage?

- Ahora, si en vez de guardar algo para siempre lo necesitás por un ratito, podés usar SessionStorage. Por ejemplo, si querés guardar el paso actual de un formulario de compra, esto sería perfecto:

Guardar datos:

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

Recuperar datos:

```
let paso = sessionStorage.getItem('pasoActual');
console.log(paso); // '2'
```

Carrito de Compras

Carrito de compras usando LocalStorage

- Llegó el momento de hacer algo que todos conocemos y usamos: un carrito de compras. Vamos a usar LocalStorage para que cuando el usuario agregue productos, queden guardados incluso si cierra la página
- El código que vas a usar va a ser algo como esto:

```
// Guardar carrito en LocalStorage
localStorage.setItem('carrito', JSON.stringify(listaDeProductos));


// Obtener carrito de LocalStorage
let carrito = JSON.parse(localStorage.getItem('carrito'));
```

Crear el Carrito de Compras (HTML)

Estructura básica para crear el Carrito de Compras

Empecemos con lo más básico: el HTML que va a mostrar nuestro carrito de compras. Vamos a crear una lista para los productos y un botón para vaciar el carrito.

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

Este bloque HTML será la estructura visual de nuestro carrito. Todo lo que agreguemos o quitemos lo vamos a manejar con JavaScript

Agregar Productos al Carrito

¿Cómo agregar productos al Carrito?

- Ahora, vamos a darle vida al carrito.
- Cada vez que el usuario haga clic en un botón de 'Agregar al carrito', vamos a guardar ese producto en LocalStorage y actualizar la lista del carrito en el HTML.

```
// Agregar producto al carrito
function agregarProducto(event) {
  var producto = {
    id: event.target.getAttribute('data-id'),
    nombre: event.target.getAttribute('data-nombre'),
    precio: event.target.getAttribute('data-precio')
  };

  var carrito = JSON.parse(localStorage.getItem('carrito')) || [];
  carrito.push(producto);
  localStorage.setItem('carrito', JSON.stringify(carrito));
  actualizarCarrito();
}
```

- Cada vez que un producto se agregue, lo guardamos en LocalStorage y actualizamos la lista del carrito que se muestra en la página

Eliminar un Producto del Carrito

¿Cómo eliminar un producto del Carrito de Compras?

A veces, el usuario se arrepiente y quiere sacar un producto. No hay problema, también lo podemos hacer. Vamos a crear un botón que permita eliminar un producto específico del carrito.

```
function eliminarProducto(idProducto) {
  var carrito = JSON.parse(localStorage.getItem("carrito")) || [];
  carrito = carrito.filter(function (producto) {
    return producto.id !== idProducto;
  });
  localStorage.setItem("carrito", JSON.stringify(carrito));
  actualizarCarrito();
}
```

Acá lo que hacemos es filtrar el carrito para eliminar el producto que tiene el id correspondiente. Luego, actualizamos el LocalStorage y la visualización.

Vaciar el Carrito de Compras

¿Cómo vaciar completamente el Carrito de Compras?

Por último, si el usuario decide que quiere empezar de cero, puede vaciar todo el carrito con un solo click.

```
document
  .getElementById("vaciar-carrito")
  .addEventListener("click", function () {
    localStorage.removeItem("carrito");
    actualizarCarrito();
  });
```

Con localStorage.removeItem(), eliminamos todo lo que estaba guardado en el carrito, dejando la página vacía. ¡Listo para empezar de nuevo!

Actualización del Carrito en Tiempo Real

Cada vez que agreguemos, eliminemos o vaciemos el carrito, tenemos que actualizar lo que se ve en la página. Acá te dejo cómo lo hacemos de manera sencilla:

Esta función toma el carrito de LocalStorage, lo recorre y vuelve a construir la lista de productos en el HTML.

```
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);
  }
}
```

¡Vamos a la práctica! 

Ejercicios Prácticos

Optativos | No entregables

Guardar preferencias de usuario

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

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

Tips:

- 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 preferencias.

Ejercicios Prácticos

Optativos | No entregables

Carrito de compras con conteo de productos

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:

- 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.