«Talento Tech»

Front End Js

CLASE 13

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase 13: JS 6 - LocalStorage, SessionStorage y Carrito de Compras

Buenos Aires aprende
Agencia de Habilidades para el Futuro

1. Introducción a 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.

1.1. ¿Qué es LocalStorage?

LocalStorage es una API del navegador que te permite guardar datos en pares clave-valor de manera persistente. Los datos almacenados permanecen incluso después de cerrar el navegador. Por ejemplo, podés almacenar configuraciones como el tema de la aplicación:

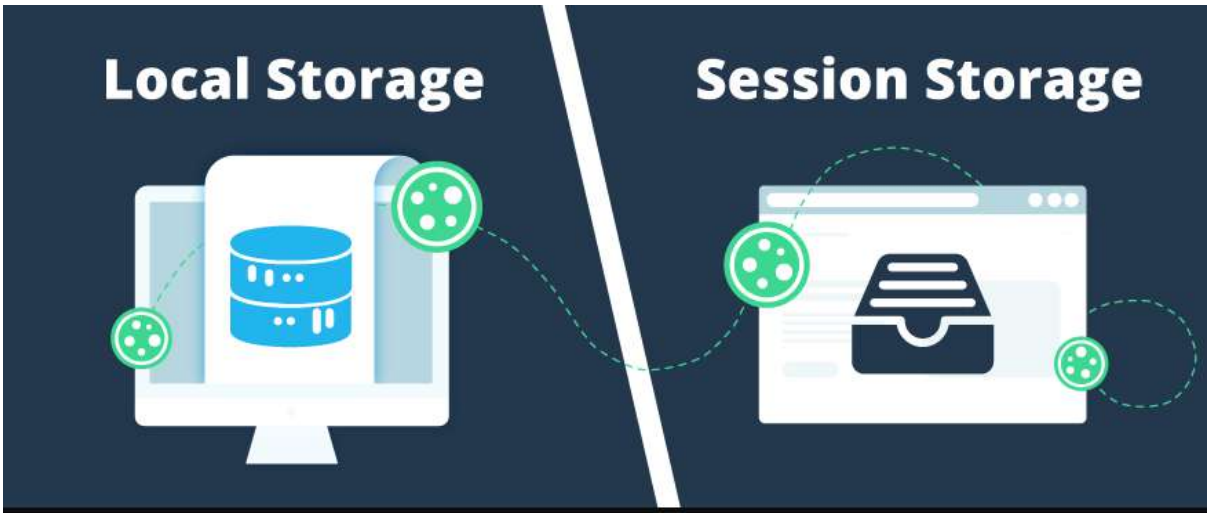
```
localStorage.setItem('tema', 'oscuro');
```

1.2. ¿Qué es SessionStorage?

SessionStorage es similar a LocalStorage pero sus datos son temporales. Los datos se eliminan cuando la pestaña o ventana del navegador se cierra. Esto es útil para almacenar información que solo es relevante durante la sesión actual, como el estado de un formulario en un proceso de compra:

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

1.3. Diferencias entre LocalStorage y SessionStorage



- **Persistencia**: LocalStorage mantiene los datos después de cerrar el navegador, mientras que SessionStorage los elimina al cerrar la sesión.
- **Alcance**: SessionStorage está limitado a la pestaña o ventana en la que se crea, mientras que LocalStorage se comparte entre todas las pestañas del mismo dominio.
- **Uso adecuado**: LocalStorage es ideal para datos persistentes como preferencias del usuario, mientras que SessionStorage es útil para datos temporales.

2. Manipulación de LocalStorage y SessionStorage

2.1. Guardar datos en LocalStorage

Para guardar datos en LocalStorage, usás el método `setItem()`:

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

2.2. Obtener datos de LocalStorage

Para recuperar datos almacenados, usás el método `getItem()`:

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

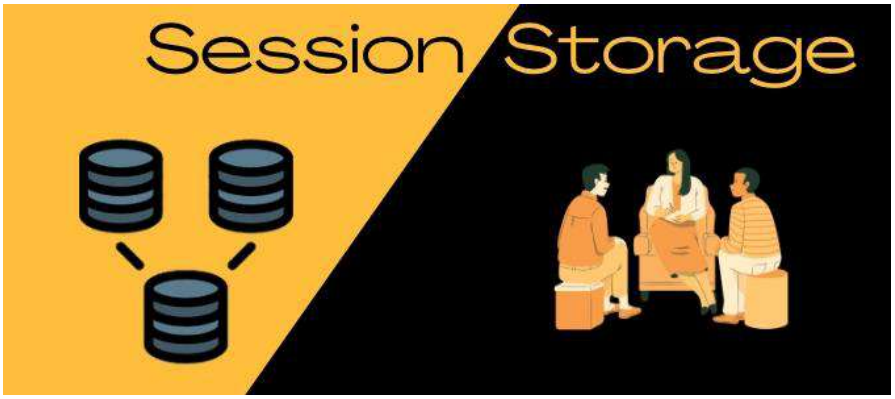
2.3. Eliminar datos de LocalStorage

Para eliminar un ítem específico, usás `removeItem()`:

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

También podés eliminar todos los datos del LocalStorage con `clear()`:

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



2.4. Guardar datos en SessionStorage

Similar a LocalStorage, podés usar `setItem()` para guardar datos:

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

2.5. Obtener datos de SessionStorage

Podés obtener los datos con `getItem()`:

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

2.6. Eliminar datos de SessionStorage

Para eliminar datos de SessionStorage, usás `removeItem()`:

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

Para borrar todos los datos de SessionStorage:

```
sessionStorage.clear();
```

3. Implementación de un Carrito de Compras con LocalStorage y SessionStorage

3.1. 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>
```

3.2. Funcionalidades del Carrito de Compras con JavaScript

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

```
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();
});
```

3.3. Uso de LocalStorage para guardar el estado del carrito

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

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

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."

3.4. Actualización del Carrito en tiempo real

Cada vez que el usuario añada o elimine un producto, actualizá el DOM y el LocalStorage para reflejar los 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);
 }
}
```

4. Ejercicio Práctico de ejemplo: Carrito de Compras

4.1. Enunciado del Ejercicio

Desarrollá un carrito de compras que permita:

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

4.2. Pasos a seguir

1. Crear la estructura HTML del carrito.
2. Implementar la funcionalidad de agregar productos con JavaScript.
3. Utilizar LocalStorage para guardar los productos.
4. Mostrar el contenido del carrito al cargar la página.
5. Implementar la opción de vaciar el carrito.

4.3. Tips para la 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.
-

Código Completo: Carrito de Compras

Te dejo el código HTML, CSS y JavaScript completo para que puedas crear tu propio carrito de compras.

Luego vos deberás hacer lo mismo en tu sitio, separando el js en un archivo aparte y adaptando todo a tu sitio. (Este ejercicio será el segundo de la ppt de ejercicios de esta semana)

Buenos Aires aprende
Agencia de Habilidades para el Futuro

```
<!DOCTYPE html>
<html lang="es">
<head>
 <meta charset="UTF-8">
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
 <title>Carrito de Compras</title>
 <style>
   body {
     font-family: Arial, sans-serif;
   }
   #carrito {
     margin-top: 20px;
   }
   #lista-carrito {
     list-style: none;
     padding: 0;
   }
   #lista-carrito li {
     background-color: #f4f4f4;
     margin: 5px 0;
     padding: 10px;
     border: 1px solid #ccc;
   }
   #vaciar-carrito {
     margin-top: 10px;
     background-color: red;
     color: white;
     border: none;
     padding: 10px 20px;
     cursor: pointer;
   }
   #vaciar-carrito:hover {
     background-color: darkred;
   }
```

```
 </style>
</head>
<body>

 <h1>Tienda Online</h1>

 <!-- Productos -->
 <div id="productos">
   <h2>Productos Disponibles</h2>
   <ul>
     <li>
       Producto 1 - $10
       <button class="agregar-carrito" data-id="1" data-
nombre="Producto 1" data-precio="10">Agregar al Carrito</button>
     </li>
     <li>
       Producto 2 - $15
       <button class="agregar-carrito" data-id="2" data-
nombre="Producto 2" data-precio="15">Agregar al Carrito</button>
     </li>
     <li>
       Producto 3 - $20
       <button class="agregar-carrito" data-id="3" data-
nombre="Producto 3" data-precio="20">Agregar al Carrito</button>
     </li>
   </ul>
 </div>

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

```
<!-- Este Script es buena practica que lo pongas en un archivo .js
separado -->

 <script>
   document.addEventListener('DOMContentLoaded', function() {
     cargarCarrito();
   });

   // Agregar producto al carrito
   var botonesAgregar = document.getElementsByClassName('agregar-
carrito');
   for (var i = 0; i < botonesAgregar.length; i++) {
     botonesAgregar[i].addEventListener('click', agregarProducto);
   }

   // Vaciar carrito
   document.getElementById('vaciar-carrito').addEventListener('click',
function() {
     localStorage.removeItem('carrito');
     cargarCarrito();
   });

   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));
```

```
    cargarCarrito();
  }

  function cargarCarrito() {
    var listaCarrito = document.getElementById('lista-carrito');
    listaCarrito.innerHTML = '';

    var carrito = JSON.parse(localStorage.getItem('carrito')) || [];

    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);
    }
  }
 </script>

</body>
</html>
```

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicios:

Ejercicio 1: Guardar Preferencias de Usuario

Enunciado:

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.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Ejercicio 2: Carrito de Compras con Conteo de Productos

Enunciado:

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.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad