«Talento Tech»

Front-End JS

Clase 13

Buenos Aires Ciudad

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

Objetivo de la clase:

En esta clase aprenderás a utilizar LocalStorage y SessionStorage para almacenar datos en el navegador de forma persistente o temporal. Conocerás sus diferencias, ventajas y limitaciones, y practicarás su uso en un proyecto concreto: la implementación de un carrito de compras. Al finalizar, serás capaz de guardar, recuperar y eliminar datos de manera local, brindando a tus proyectos una experiencia de usuario más completa y profesional.

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:

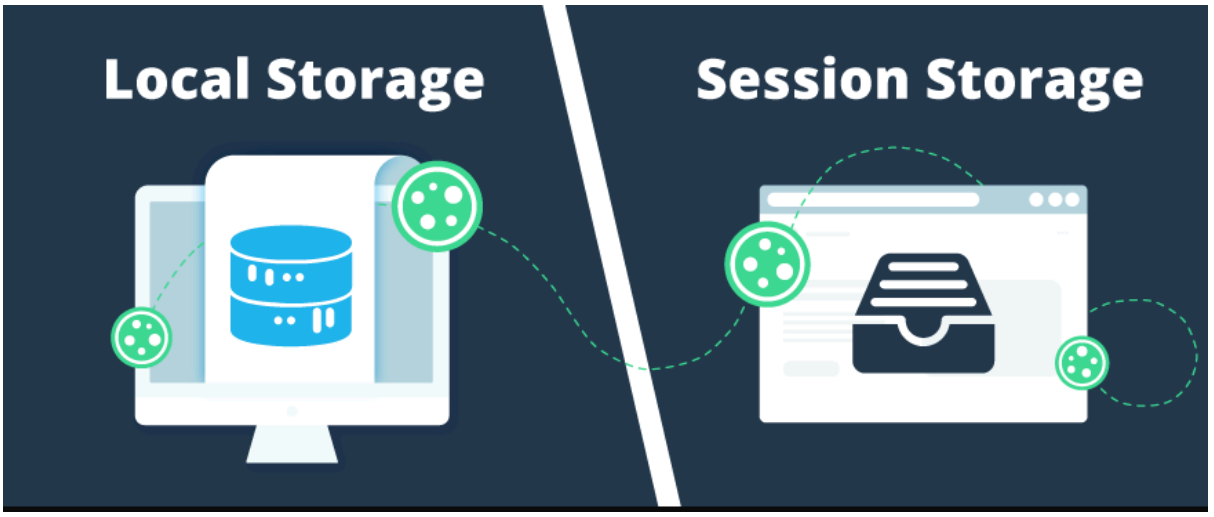
```
localStorage.setItem('tema', 'oscuro');
```

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 sólo 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)

```
<!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>
```

💭**Storytelling**

Persistencia de datos con LocalStorage y SessionStorage

Tomás (Desarrollador Senior)

Chicos y chicas, llegó el momento de dar un gran salto: **aprender a guardar información en el navegador**. Esto les va a permitir crear sitios más completos, donde las preferencias del usuario o los productos de un carrito no desaparezcan al actualizar la página.

Hoy vamos a trabajar con **LocalStorage**, una herramienta súper práctica para mantener datos persistentes entre sesiones, y aplicarla a casos reales.

Ejercicio práctico #1:

Guardar preferencias de usuario

Lucía (Product Owner)

Queremos que el usuario se sienta cómodo cada vez que entre al sitio. Por eso vamos a permitirle elegir su nombre y su color de fondo preferido, y recordar esas elecciones.

Pasos a seguir:

1. **Crear el formulario**

- Armá en tu HTML un formulario con dos campos:
  - un campo para el nombre (input).
  - un selector (select) con diferentes colores de fondo.
- Sumale un botón de enviar para guardar las preferencias.

2. **Capturar los datos**

- En JavaScript, usá un evento `submit` para capturar el nombre y el color cuando el usuario envía el formulario.
- Recordá usar `preventDefault()` para evitar que el formulario recargue la página.

3. **Guardar en LocalStorage**

- Usá `localStorage.setItem()` para guardar el nombre y el color.

4. **Recuperar y aplicar preferencias**

- Cada vez que la página se cargue, verificá si hay datos guardados en LocalStorage.
- Si existen, aplicá automáticamente:
  - el saludo con el nombre del usuario.
  - el color de fondo elegido.

Tips de Tomás:

Usá `localStorage.getItem()` para leer los datos

Manipulá el DOM con `document.body.style.backgroundColor` para cambiar el color

Probá recargando la página para chequear que la preferencia se mantiene

Ejercicio práctico #2:

Carrito de Compras con conteo de productos

Lucía (Product Owner)

Ahora necesitamos un carrito de compras básico, que **no pierda la información** si la página se recarga.

Pasos a seguir:

1. **Armar el catálogo**
- Creá un listado de productos en el HTML (pueden utilizar los que ya armaron anteriormente “Agregar al carrito”).

2. **Configurar el contador**
- Mostrá un contador (por ejemplo, en la esquina superior) que indique cuántos productos hay en el carrito.

3. **Programar el agregado**
- Cuando el usuario haga clic en “Agregar al carrito”, aumentá la cantidad de productos.
- Guardá el valor actualizado en LocalStorage.

4. **Recuperar los datos al cargar**
- Cada vez que se recargue la página, tomá el número guardado en LocalStorage.
- Mostralo en el contador del carrito.

Tips de Tomás:

Usá `addEventListener("click")` para detectar el clic en los botones

Recordá actualizar el DOM cada vez que cambie el contador

Verificá con `localStorage.getItem()` si ya existe un valor antes de empezar en 0

Ejercicios Práctico 1:

Integración

```
<!-- PREFERENCIAS DE USUARIO -->
<div id="preferencias">
  <h2>Preferencias de Usuario</h2>
  <form id="form-preferencias">
    <label>
      Nombre:
      <input type="text" id="nombre-usuario">
    </label>
    <label>
      Color de fondo:
      <select id="color-fondo">
        <option value="#ffffff">Blanco</option>
        <option value="#f8f9fa">Gris claro</option>
        <option value="#e0f7fa">Celeste</option>
        <option value="#fce4ec">Rosa</option>
      </select>
    </label>
    <button type="submit">Guardar Preferencias</button>
  </form>
</div>



<script>
```

```
  // ---------------------------
  // Funcionalidad Preferencias
  // ---------------------------
  document.addEventListener('DOMContentLoaded', function() {
    aplicarPreferencias();
  });

document.getElementById('form-preferencias').addEventListener('submi
t', function(e) {
    e.preventDefault();
    const nombre = document.getElementById('nombre-usuario').value;
    const color = document.getElementById('color-fondo').value;

    localStorage.setItem('nombreUsuario', nombre);
    localStorage.setItem('colorFondo', color);

    aplicarPreferencias();
  });

  function aplicarPreferencias() {
    const nombre = localStorage.getItem('nombreUsuario');
    const color = localStorage.getItem('colorFondo');

    if (nombre) {
      alert(`¡Bienvenido/a nuevamente, ${nombre}!`);
```

```
    }


    if (color) {
      document.body.style.backgroundColor = color;
    }
  }
</script>
```

Ejercicio Práctico 2:

Integración

Sumar al carrito de compras ya existente.

```
<h2>Carrito de Compras (<span id="contador-productos">0</span>
productos)</h2>
```

```
// dentro de  la función y como último llamado a cargarCarrito()
document.getElementById('contador-productos').textContent =
carrito.length;
```

Buenos Aires aprende

Agencia de Habilidades para el Futuro

Buenos Aires Ciudad