# Instituto tecnológico del valle de Oaxaca

## PROGRAMACION II

**UNIDAD 3**

**Actividad para contrastar**

**Tema: locaStorage Y sessionStorage**

**INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES**

**Alumnos: OSWALDO MIGUEL LÓPEZ**

**N° de control: 21920131**

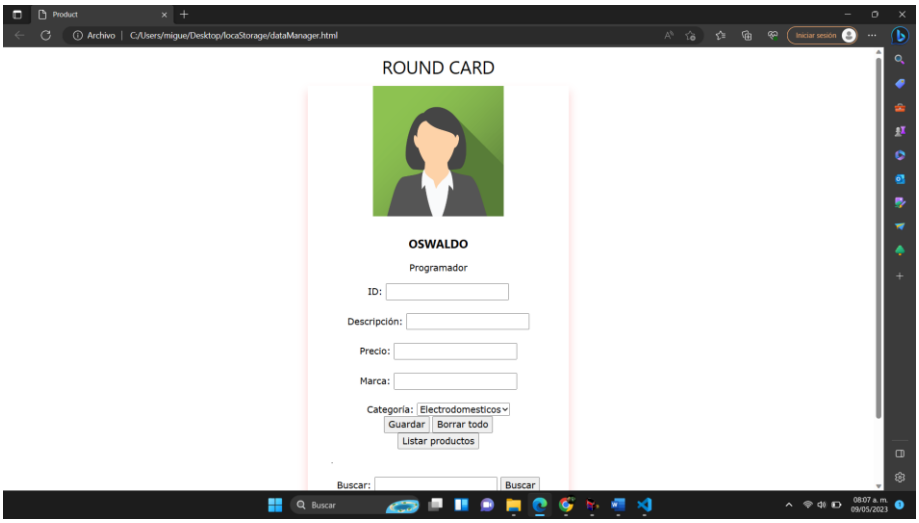
**Docente: Cardozo ambrosio Giménez**

**Fecha de entrega: 24 de Febrero del 2023**

# INTRODUCCIÓN:

HTML5 es una tecnología que ha revolucionado la forma en que se crean y consumen contenidos web. Con la introducción de HTML5, se ha hecho posible desarrollar aplicaciones web que antes solo eran posibles en plataformas de escritorio. Uno de los principales beneficios de HTML5 es su capacidad para permitir el acceso a datos locales del usuario a través del navegador, lo que ha abierto nuevas posibilidades en el desarrollo de aplicaciones web.

En este sentido, el acceso a datos HTML5 permite a los desarrolladores de aplicaciones web crear aplicaciones más ricas y dinámicas, capaces de ofrecer una experiencia de usuario mejorada. En esta breve guía, se explorará cómo se puede acceder a datos en HTML5, qué herramientas se pueden utilizar y cómo se pueden utilizar para crear aplicaciones web más potentes y flexibles.




# LocalStorage

El término "dataManager" puede referirse a diferentes cosas dependiendo del contexto. Por ejemplo, puede referirse a una clase o función en un programa de software que se utiliza para administrar datos, o puede referirse a una persona o equipo responsable de la gestión de datos en una organización.

- get: devuelve todos los datos de la colección.
- push: ingresa nuevos datos en la colección.
- set: ingresa una nueva colección completa.
- delete: elimina todos los datos de la colección.

```
let database = new dataManager('productos');

function dataManager(name) {

  let DB = (localStorage.getItem(name)) ? JSON.parse(localStorage.getItem(name)) : [];

  return {
    // obtener todos los datos de la colección
    get: () => {
      return DB;
    },
    // ingresar nuevos datos
    push: (obj) => {
      DB.push(obj);
      localStorage.setItem(name, JSON.stringify(DB));
    },
    // ingresar una nueva colección
    set: (collection) => {
      DB = collection;
      localStorage.setItem(name, JSON.stringify(DB));
    },
    // eliminar todos los datos de la colección
    delete: () => {
      DB = [];
      localStorage.setItem(name, JSON.stringify(DB));
    }
  }
}
```

## SessionStorage

Además del acceso a datos en HTML5, el uso de sessionStorage es otro concepto importante para el desarrollo de aplicaciones web modernas. El objeto global de JavaScript, sessionStorage, proporciona una forma conveniente para que los desarrolladores almacenen datos de manera temporal en el navegador web del usuario.
A diferencia de localStorage, que almacena los datos de forma permanente hasta que se borran manualmente, sessionStorage se utiliza para almacenar datos solo mientras la ventana del navegador está abierta. Esto significa que los datos se pierden cuando se cierra la ventana del navegador o se abandona la página.
En el código mostrado en la imagen, se utiliza sessionStorage para crear un administrador de datos llamado "dataManager" que se encarga de almacenar y recuperar información en una colección llamada "productos". La función dataManager recibe un nombre como parámetro que se utiliza como clave para almacenar y recuperar los datos en el sessionStorage.

```
function dataManager(name) {

  let DB = (sessionStorage.getItem(name)) ? JSON.parse(sessionStorage.getItem(name)) : [];

  return {
    // obtener todos los datos de la colección
    get: () => {
      return DB;
    },
    // ingresar nuevos datos
    push: (obj) => {
      DB.push(obj);
      sessionStorage.setItem(name, JSON.stringify(DB));
    },
    // ingresar una nueva colección
    set: (collection) => {
      DB = collection;
      sessionStorage.setItem(name, JSON.stringify(DB));
    },
    // eliminar todos los datos de la colección
    delete: () => {
      DB = [];
      sessionStorage.setItem(name, JSON.stringify(DB));
    }
  }
}
```

El código muestra una función llamada "saveData" que se encarga de guardar los datos de un producto ingresados por el usuario en el formulario en la base de datos. Primero, la función obtiene los valores ingresados por el usuario en el formulario (ID, descripción, precio, marca y categoría) y los asigna a las variables correspondientes. Luego, la función verifica si el ID ingresado es un número entero utilizando la función "Number.isInteger()". Si el ID no es un número entero, se muestra un mensaje de alerta y la función se detiene.
A continuación, se obtiene la lista de productos existentes en la base de datos utilizando la función "get()" del objeto "database" y se verifica si el producto ingresado ya existe en la base de datos utilizando el método "find()" de JavaScript. Si el producto ya existe, se muestra un mensaje de alerta y la función se detiene.
Luego, la función verifica si el precio ingresado es un número utilizando la función "isNaN()" de JavaScript. Si el precio no es un número, se muestra un mensaje de alerta y la función se detiene.
Finalmente, se crea un nuevo objeto "Producto" utilizando los valores ingresados por el usuario y se agrega a

```
function saveData() {
  const id = parseInt(document.getElementById("id").value);
  const descripcion = document.getElementById("descripcion").value;
  const precio = parseFloat(document.getElementById("precio").value);
  const marca = document.getElementById("marca").value;
  const categoria = document.getElementById("categoria").value;

  //validamos que el ID sea un número entero
  if (!Number.isInteger(id)) {
    alert('El ID debe ser un número entero');
    return;
  }

  //validamos que el ID no se repita con otro producto
  const productos = database.get();
  const productoExistente = productos.find(producto => producto.id === id);
  if (productoExistente) {
    alert('El ID que proporciono ya existe');
    return;
  }

  //validamos que el precio sea un número decimal o entero
  if (isNaN(precio)) {
    alert('El precio debe ser un número');
    return;
  }

  const producto = new Producto(id, descripcion, precio, marca, categoria);

  database.push(producto);
}
```

La función listData muestra todos los productos en la base de datos en una tabla HTML en la página web.

```
function listData() {
  let productos = database.get();
  let table = document.getElementById("products");
  table.innerHTML = "";
  let i = 0;
  productos.forEach(producto => {
    let row = table.insertRow(i);
    let cell = row.insertCell(0);
    cell.innerHTML = producto.id;
    cell = row.insertCell(1);
    cell.innerHTML = producto.descripcion;
    cell = row.insertCell(2);
    cell.innerHTML = producto.precio;
    cell = row.insertCell(3);
    cell.innerHTML = producto.marca;
    cell = row.insertCell(4);
    cell.innerHTML = producto.categoria;
    i++;
  });
}
```

function(producto: any): void

La función searchData busca un producto en la base de datos por ID y lo muestra en una tabla HTML en la página web. Si el producto no existe, se muestra un mensaje de error.

```
function searchData() {
  const id = parseInt(document.getElementById("search").value);
  const productos = database.get();
  const productoEncontrado = productos.find(producto => producto.id === id);
  let table = document.getElementById("product");
  if (productoEncontrado) {
    let row = table.insertRow(0);
    let cell = row.insertCell(0);
    cell.innerHTML = productoEncontrado.id;
    cell = row.insertCell(1);
    cell.innerHTML = productoEncontrado.descripcion;
    cell = row.insertCell(2);
    cell.innerHTML = productoEncontrado.precio;
    cell = row.insertCell(3);
    cell.innerHTML = productoEncontrado.marca;
    cell = row.insertCell(4);
    cell.innerHTML = productoEncontrado.categoria;
  } else {
    let row = table.insertRow(0);
    let cell = row.insertCell(0);
    cell.innerHTML = "Producto no encontrado";
  }
}
function clearSearch() {
  document.getElementById("product").innerHTML = "";
}
```

La función deleteProduct elimina un producto de la base de datos por ID. Si el producto no existe, se muestra un mensaje de error. Si el producto es eliminado correctamente, la tabla HTML se actualiza para mostrar la base de datos actualizada.

```
function deleteProduct() {
  const id = parseInt(document.getElementById("search").value);
  let productos = database.get();
  const index = productos.findIndex(producto => producto.id === id);
  if (index !== -1) {
    productos.splice(index, 1);
    database.set(productos);
    alert('Producto eliminado correctamente');
    clearSearch();
    listData();
  } else {
    alert('Producto no encontrado');
  }
}
```

Este código es útil si deseas dar formato y estilo a una página web y es especialmente útil si deseas crear secciones o secciones de contenido específicas en tu página.

```css
body {
  text-align: center;
}
.card {
  box-shadow: 0 10px 15px 0 rgba(238, 4, 4, 0.2);
  transition: 0.3s;
  width:30%;
  margin: auto;
}

.container {
  padding: 20px 40px;
}
```

El código que se muestra es un archivo HTML que representa la estructura de una página web. La página web es una aplicación que permite agregar, listar, buscar, editar y eliminar productos.
La estructura de la página se define dentro del elemento HTML y consta de varios elementos, como encabezados, imágenes y botones.
En la sección head del documento, se incluyen elementos como el título de la página, la descripción de la vista de la página y la referencia a archivos CSS y JavaScript externos. El archivo "storage.js" se carga en esta sección y contiene las funciones que manejan el almacenamiento de datos en el sessionStorage.

El cuerpo de la página contiene un encabezado principal, un contenedor de una tarjeta que contiene una imagen y el formulario para agregar nuevos productos. El formulario incluye campos para ingresar la información del producto, como el ID, descripción, precio, marca y categoría.

La página también incluye botones para guardar, borrar todo, listar, buscar, eliminar y editar productos. La tabla de productos se muestra debajo del formulario y se actualiza dinámicamente con los datos de los productos que se agregan.
En resumen, este código representa la estructura HTML de una página web que permite agregar, listar, buscar, editar y eliminar productos, utilizando sessionStorage para almacenar los datos de forma temporal en el navegador web del usuario.

```html
<!DOCTYPE html>
<html>
  <head>
    <title>Product </title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta charset="utf-8">
    <link rel="stylesheet" href="https://www.w3schools.com/w3css/4/w3.css">
    <link rel="stylesheet" href="style.css" type="text/css">
    <script src="storage.js"></script>
  </head>
  <body>
    <h4><h2>Tiendita </h2></h4>

    <div class="card">
      <img src="img_avatar.jfif" alt="Avatar">
      <div class="container">
        <form>
          <label for="id">ID:</label>
          <input type="text" id="id">
          <br>
          <label for="descripcion">Descripción:</label>
          <input type="text" id="descripcion">
          <br>
          <label for="precio">Precio:</label>
          <input type="text" id="precio">
          <br>
          <label for="marca">Marca:</label>
          <input type="text" id="marca">
          <br>
          <label for="categoria">Categoría:</label>
          <select id="categoria">
            <option value="Electrodomesticos">Electrodomesticos</option>
            <option value="Comida">Comida</option>
            <option value="Ropa">Ropa</option>
            <option value="Hogar">Hogar</option>
          </select>
          <br>
          <button id="btnSave" onclick="saveData();">Guardar</button>
          <button id="btnDelete" onclick="deleteData();">Borrar todo</button>
        </form>
        <br>
        <button id="btnList" onclick="listData();">Listar productos</button>
        <div>
          <table id="products" border="1"><br>
          </table>
        </div>
        <br>
        <label for="search">Buscar:</label>
        <input type="text" id="search">
        <button id="btnSearch" onclick="searchData();">Buscar</button>
        <button id="btnClearSearch" onclick="clearSearch();">Limpiar</button>
        <button id="btnDelete" onclick="deleteProduct()">Eliminar </button>
        <table id="product" border="1"><br>
        </table>
      </div>
      <div>
      <button id="btnDelete" onclick="editData()">Editar</button>
      <button id="btnSave" onclick="saveEdit(id) ">Guardar</button>
      <table id="product" border="1">
  </table>
</div>
      </div>
    <br><br>
  </body>
```

# Conclusion

El acceso a datos en HTML5 es una tecnología importante para el desarrollo de aplicaciones web modernas y dinámicas. La capacidad de acceder a los datos locales del usuario a través del navegador ha abierto nuevas posibilidades para los desarrolladores de aplicaciones web, lo que les permite crear aplicaciones que antes solo eran posibles en plataformas de escritorio.

Al utilizar herramientas como el almacenamiento local y las API de base de datos, los desarrolladores pueden crear aplicaciones web que ofrezcan una experiencia de usuario mejorada y más flexible. Además, al aprovechar la capacidad de HTML5 para trabajar sin conexión, las aplicaciones web pueden ser utilizadas en cualquier momento y en cualquier lugar.

En resumen, el acceso a datos HTML5 es una tecnología clave para el desarrollo de aplicaciones web modernas y puede ser utilizada para crear aplicaciones que ofrezcan una experiencia de usuario mejorada y más rica. Con las herramientas adecuadas y una comprensión sólida de cómo funciona la tecnología, los desarrolladores pueden crear aplicaciones web más potentes y flexibles que se adapten a las necesidades de sus usuarios.

## REPOSITORIO:

https://github.com/msdjjd/datos.git