Exercicio 51: engadir a unha lista da compra

Obxectivo: Este programa ten como obxectivo crear unha lista na páxina web que lle permite ao usuario engadir elementos á lista e garda o último elemento introducido no **almacenamento local do navegador (*localstorage*)** para que persista incluso despois de pechar e volver abrir a páxina.

Que é o localStorage?

localStorage é unha propiedade da interface ***window*** en JavaScript que **permite almacenar datos de xeito local no navegador do usuario**. Os datos almacenados en localStorage persisten incluso despois de pechar o navegador ou recargar a páxina, a menos que sexan eliminados manualmente polo usuario ou polo código da aplicación.

Características principais:

- **Persistencia**: Os datos non expiran e permanecen almacenados ata que se eliminen explicitamente.

- **Formato clave-valor**: A información gárdase como pares clave-valor en formato string (UTF-16). Se se desexa almacenar obxectos ou arrays, deben ser convertidos a JSON usando JSON.stringify() e recuperados con JSON.parse().

- **Tamaño limitado:** Xeralmente, permite almacenar entre 5 MB e 10 MB de datos.

Métodos principais:

1. **Gardar datos:**

   ```
   localStorage.setItem('clave', 'valor');
   ```

   - Exemplo con obxectos:

   ```
   const usuario = { nome: 'Ana', idade: 25 };
   localStorage.setItem('usuario', JSON.stringify(usuario));
   ```

2. **Recuperar datos:**

   ```
   const valor = localStorage.getItem('clave');
   ```

   - Para obxectos:

   ```
   const usuario = JSON.parse(localStorage.getItem('usuario'));
   ```

3. **Eliminar un dato específico**:

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

4. **Eliminar todos os datos**:

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

5. **Obter unha clave por índice para recuperar un dato nunha posición específica dentro da memoria:**

```
const clave = localStorage.key(0);
```

Cando usalo:

Utilízase cando é necesario almacenar información persistente no navegador, como configuracións de usuario, tokens de autenticación ou datos temporais que non requiren unha base de datos externa. A diferenza de sessionStorage, os datos en localStorage sobreviven ao peche do navegador.

Parte do HTML:

1. **Primeiros pasos:**

Para comezar co exercicio é necesario crear unha carpeta chamada *Exercicio51* esta será a carpeta onde gardaremos este traballo. Dentro da carpeta crearemos un ficheiro *index.html* e crearemos tamén unha carpeta chamada *js* onde crearemos o noso arquivo *script.js*

Creamos tamén unha carpeta para os ficheiros de CSS e creamos dentro un arquivo *style.css* para os estilos.
Unha vez feito isto crearemos a estrutura básica dun ficheiro no html e vincularemos o noso script, para isto escribiremos o seguinte antes do peche da etiqueta <body>:
<script src="js/script.js"></script>

No ficheiro de *script.js* escribiremos o código JavaScript deste exercicio para poder gardarnos unha copia do traballo.

Entrega: Unha vez completado todo **subiremos a carpeta *Exercicio50*** ao espazo para entrega de exercicios que temos no OneDrive de Microsoft Teams.

2. **Corpo (body):**
   - Creamos dúas divisións (divs) principais no corpo do documento.

3. **División para introducir datos**:
   - Creamos un div co id "ingresarDatos" que contén un campo de entrada de texto (`<input>`) co id "novaEntrada" e un marcador de posición ("Escribe aquí").
   - Tamén contén un botón co texto "Añadir" e un atributo `onclick` que chama á función `agregarElemento()` cando se fai clic.

4. **División para a lista:**
   - Un div que contén unha lista desordenada (`<ul>`) co id "lista". Inicialmente, esta lista está baleira.

O resultado debería de quedar así:

HTML:

```
<div id="ingresarDatos">
  <input type="text" id="novaEntrada" placeholder="Escribe aquí"/>
  <button onclick="agregarElemento()">Engadir</button>
</div>

<div>
  <ul id="lista">
  </ul>
</div>
```

Parte do JavaScript:

5. **Script (JavaScript):**
   - Decláranse tres variables globais (`entradaLista`, `listaHTML`, e `listaGuardada`) utilizando `var`.
   - `entradaLista` asignáselle ao elemento de entrada de texto co id "novaEntrada".
   - `listaHTML` asignáselle ao elemento de lista desordenada co id "lista".
   - `listaGuardada` inicialízase utilizando `**localStorage.getItem**("listaGuardada")`. Isto **recupera o último texto introducido e gardado localmente**.
   - Comprobamos cun condicional ìf` se hai unha `listaGuardada` e se agrega os elementos a lista almacenada al HTML.

6. **Función `agregarElemento()`:**
   - Esta función chámase cando se fai clic no botón "Engadir".
   - Decláranse duas variables locais (`novoTexto` e `novoElemento`).
   - `novoTexto` asignáselle ao valor actual do campo de entrada (`entradaLista.value`).
   - `novoElemento` créase como un novo elemento de lista (`<li>`) e o seu texto interno establécese como `novoTexto`.
   - O novo elemento engádeselle á lista desordenada (`listaHTML.**appendChild(novoElemento)**`).
   - O último texto introducido **gardase no almacenamento local** do navegador `**localStorage.setItem**("listaGuardada", listaHTML.innerHTML)`.

JavaScript:

```
var entradaLista;

var listaHTML;

entradaLista=document.getElementById("novaEntrada")

listaHTML=document.getElementById("lista")
```

```
ultimoTexto=localStorage.getItem("ultimoTexto")
entradaLista.value=ultimoTexto;


var listaGuardada = localStorage.getItem("listaGuardada");
if (listaGuardada) {
  listaHTML.innerHTML = listaGuardada;
}


 function agregarElemento () {
 var novoTexto;
 var novoElemento;
 novoTexto=entradaLista.value
 novoElemento=document.createElement("li");
 novoElemento.innerText=novoTexto;
 listaHTML.appendChild(novoElemento);
 localStorage.setItem("listaGuardada", listaHTML.innerHTML);
entradaLista.value = ""; //Borramos o texto do input
 }
```

Este código HTML con JavaScript crea unha interface sinxela que lle permite ao usuario engadir elementos a unha lista, e o último elemento engadido gárdase e recupérase do almacenamento local para persistencia.

EXTRA: Engade estilos segundo o voso gusto e preferencia para facer máis agradable o aspecto deste programa.

Ejercicio 51: Añadir a una lista de la compra

Objetivo: Este programa tiene como objetivo crear una lista en la página web que permita al usuario añadir elementos a la lista y guardar el último elemento introducido en el **almacenamiento local del navegador (localStorage)** para que persista incluso después de cerrar y volver a abrir la página.

¿Qué es el localStorage?

localStorage es una propiedad de la interfaz ***window*** en JavaScript que **permite almacenar datos de forma local en el navegador del usuario.** Los datos almacenados en localStorage persisten incluso después de cerrar el navegador o recargar la página, a menos que sean eliminados manualmente por el usuario o por el código de la aplicación.

Características principales:

- **Persistencia**: Los datos no expiran y permanecen almacenados hasta que se eliminen explícitamente.

- **Formato clave-valor:** La información se guarda como pares clave-valor en formato string (UTF-16). Si se desea almacenar objetos o arrays, deben ser convertidos a JSON usando JSON.stringify() y recuperados con JSON.parse().

- **Tamaño limitado**: Generalmente, permite almacenar entre 5 MB y 10 MB de datos.

Métodos principales:

1. **Guardar datos:**

   localStorage.setItem('clave', 'valor');

- **Ejemplo con objetos:**

  const usuario = { nombre: 'Ana', edad: 25 };
  localStorage.setItem('usuario', JSON.stringify(usuario));

2. **Recuperar datos:**

   const valor = localStorage.getItem('clave');

- **Para objetos:**

  const usuario = JSON.parse(localStorage.getItem('usuario'));

3. **Eliminar un dato específico:**

   localStorage.removeItem('clave');

4. **Eliminar todos los datos:**

   localStorage.clear();

5. Obtener una clave por índice para recuperar un dato en una posición específica dentro de la memoria:

   const clave = localStorage.key(0);

Cuándo usarlo:

Se utiliza **cuando es necesario almacenar información persistente en el navegador**, como configuraciones de usuario, tokens de autenticación o datos temporales que no requieren una base de datos externa. A diferencia de sessionStorage, los datos en localStorage sobreviven al cierre del navegador

Parte del HTML

1. **Primeros pasos:**

- Para comenzar con el ejercicio, es necesario crear una carpeta llamada **Ejercicio51**, que será donde guardaremos este trabajo. Dentro de la carpeta, crearemos un archivo index.html y también una carpeta llamada js, donde crearemos nuestro archivo script.js.

- También crearemos una carpeta para los archivos de CSS y dentro de ella un archivo style.css para los estilos.

- Una vez hecho esto, crearemos la estructura básica de un archivo HTML y vincularemos nuestro script. Para ello, escribiremos lo siguiente antes del cierre de la etiqueta <body>:
  <script src=”js/script.js”></script>

En el archivo script.js, escribiremos el código JavaScript de este ejercicio para poder guardar una copia del trabajo.

Entrega: Una vez completado todo, subiremos la carpeta ***Ejercicio51*** al espacio para entrega de ejercicios que tenemos en el OneDrive de Microsoft Teams.

2. Cuerpo (body):
- Creamos dos divisiones (divs) principales en el cuerpo del documento.

3. División para introducir datos:
- Creamos un div con el id "ingresarDatos", que contiene un campo de entrada de texto (<input>) con el id "novaEntrada" y un marcador de posición ("Escribe aquí").
- También contiene un botón con el texto "Añadir" y un atributo onclick que llama a la función agregarElemento() cuando se hace clic.

4. División para la lista:
- Un div que contiene una lista desordenada (<ul>) con el id "lista". Inicialmente, esta lista está vacía.

El resultado debería quedar así:

HTML:

```
<div id="ingresarDatos">
  <input type="text" id="novaEntrada" placeholder="Escribe aquí"/>
  <button onclick="agregarElemento()">Engadir</button>
</div>

<div>
  <ul id="lista">
  </ul>
</div>
```

Parte del JavaScript

5. Script (JavaScript):

- Se declaran tres variables globales (entradaLista, listaHTML y listaGuardada) utilizando var.
- A entradaLista se le asigna el elemento de entrada de texto con el id "novaEntrada".
- A listaHTML se le asigna el elemento de lista desordenada con el id "lista".
- A listaGuardada se le inicializa utilizando **localStorage.getItem("listaGuardada").** Esto **recupera los elementos guardados localmente.**
- Comprobamos con un condicional if si hay una listaGuardada. Si existe, se agregan los elementos almacenados al HTML.

6. Función agregarElemento():

- Esta función se llama cuando se hace clic en el botón "Añadir".
- Se declaran dos variables locales (novoTexto y novoElemento).
- A novoTexto se le asigna el valor actual del campo de entrada (entradaLista.value).
- A novoElemento se le crea como un nuevo elemento de lista (<li>), y su texto interno se establece como novoTexto.
- El nuevo elemento se añade a la lista desordenada (listaHTML.appendChild(novoElemento)).
- El contenido actualizado de **la lista se guarda en el almacenamiento local** del navegador mediante **localStorage.setItem("listaGuardada", listaHTML.innerHTML).**

Código JavaScript completo:

JavaScript:

```
var entradaLista;

var listaHTML;

entradaLista=document.getElementById("novaEntrada")

listaHTML=document.getElementById("lista")
```

```
ultimoTexto=localStorage.getItem("ultimoTexto")

entradaLista.value=ultimoTexto;


var listaGuardada = localStorage.getItem("listaGuardada");

if (listaGuardada) {

 listaHTML.innerHTML = listaGuardada;

}


 function agregarElemento () {

 var novoTexto;

 var novoElemento;

 novoTexto=entradaLista.value

 novoElemento=document.createElement("li");

 novoElemento.innerText=novoTexto;

 listaHTML.appendChild(novoElemento);

 localStorage.setItem("listaGuardada", listaHTML.innerHTML);

entradaLista.value = ""; //Borramos o texto do input

 }
```

Explicación del código:

1. **Variables globales:**
   - Se definen las variables necesarias para interactuar con los elementos HTML.
   - Se recuperan datos almacenados previamente en localStorage (si existen).

2. **Función principal:**
   - La función agregarElemento() añade nuevos elementos a la lista desordenada.
   - Cada vez que se añade un nuevo elemento, toda la lista se guarda en localStorage para mantener su estado incluso después de recargar o cerrar la página.

3. **Persistencia:**
   - Al cargar la página, si hay datos guardados en **localStorage**, estos se cargan automáticamente en la lista.

Este código HTML con JavaScript crea una interfaz sencilla que permite al usuario añadir elementos a una lista. Los elementos añadidos persisten gracias al uso del almacenamiento local (localStorage).

EXTRA: Añade estilos a tu gusto y preferencia para hacer más agradable el aspecto de este programa.