LocalStorage y SessionStorage ¿Qué son y para qué se utilizan?

Los objetos de almacenamiento web `localStorage` y `sessionStorage` permiten guardar pares de clave/valor en el navegador.
Lo que es interesante sobre ellos es que los datos sobreviven a una recarga de página (en el caso de `sessionStorage`) y hasta un reinicio completo de navegador (en el caso de `localStorage`).

El objeto `sessionStorage` se utiliza mucho menos que `localStorage`.
Las propiedades y métodos son los mismos, pero es mucho más limitado:

- `sessionStorage` solo existe dentro de la pestaña actual del navegador (Otra pestaña con la misma página tendrá un almacenaje distinto).
- Los datos sobreviven un refresco de página, pero no cerrar/abrir la pestaña.

La propiedad de sólo lectura localStorage te permite acceder al objeto local Storage; los datos persisten almacenados entre de las diferentes sesiones de navegación.Los datos almacenados en localStorage no tienen fecha de expiración.

Las claves y los valores son siempre cadenas de texto (al igual que con los objetos, las claves de enteros se convertirán automáticamente en cadenas de texto).

¿Cómo guardamos datos en LocalStorage/SessionStorage?

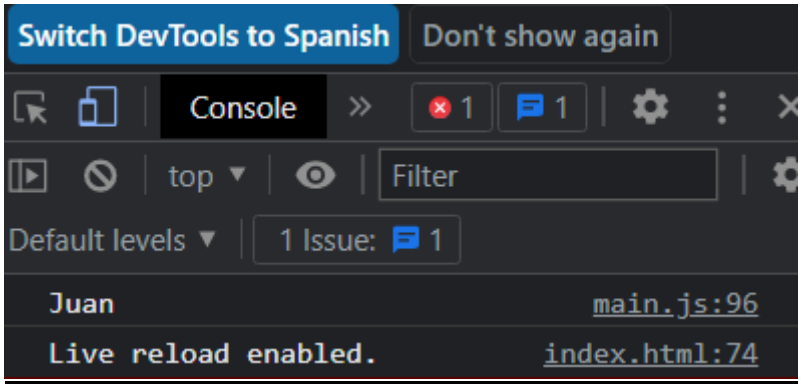
Tanto en uno, como en otro, utilizamos los mismos métodos para almacenar o eliminar información.
Para guardar un dato, accedemos al objeto localStorage actual, seguido del método setItem() que recibe por parámetro una clave y un valor. Ejemplo:
localStorage.setItem('nombre', 'Juan');
Luego para poder leer el valor almacenado, utilizaremos el método getItem(), y va a recibir por parámetro el nombre de la clave a la cual queremos leer el valor. Ejemplo:
let nombre Guardado = localStorage.getItem(‘nombre’);

```
localStorage.setItem('nombre', 'Juan');

let nombreGuardado = localStorage.getItem('nombre');

console.log(nombreGuardado)
```

si mostramos por consola la variable nombre guardado, vamos a poder ver el nombre ‘Juan’

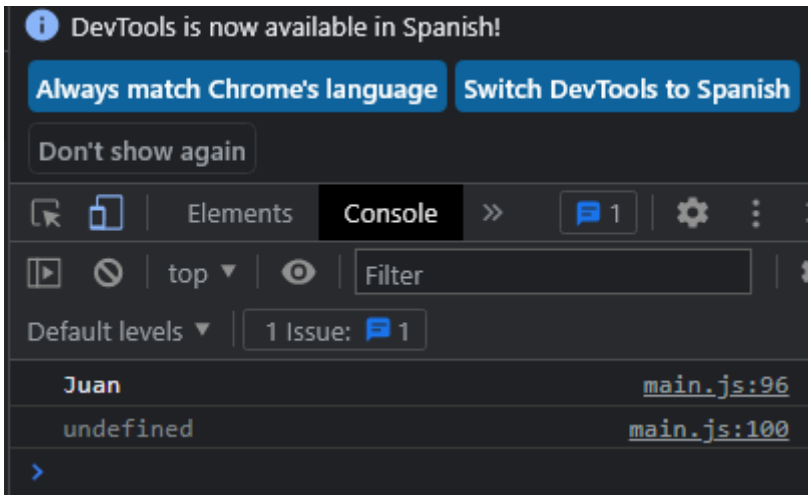


Ahora, si queremos borrar ese mismo ítem, vamos a utilizar el método removeItem(), y va a recibir por parámetro el nombre de la clave que queremos eliminar:

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

console.log(localStorage.removeItem('nombre'));
```

Y si vemos por consola, nos daremos cuenta que esa clave, vuelve a estar indefinida:



JSON (JavaScript Object Notation)

JSON, cuyas siglas significan en verdad JavaScript object notation que, en español se traducen como, notación de objetos de JavaScript, es un formato de intercambio de datos que resulta muy fácil de leer y escribir para los programadores y sencillo de interpretar y crear para las máquinas. Este objeto define seis tipos de valores: nulo, números, cadenas, booleanos, matrices y objetos.

Es un estándar basado en texto plano que sirve para intercambiar datos y se usa cuando se requiere que un sistema requiera mostrar o enviar información para que otros sistemas los lean e interpreten.

Básicamente JSON es un archivo que contiene datos estructurados, que se utiliza para transferir información entre sistemas.
Su funcionamiento se basa en la estructuración de una colección de pares con nombre y valor que contienen:

- Una clave: que corresponde al identificador del contenido.
- Un valor: que representa el contenido correspondiente.

Métodos JSON.parse() y JSON.stringify() para conversiones

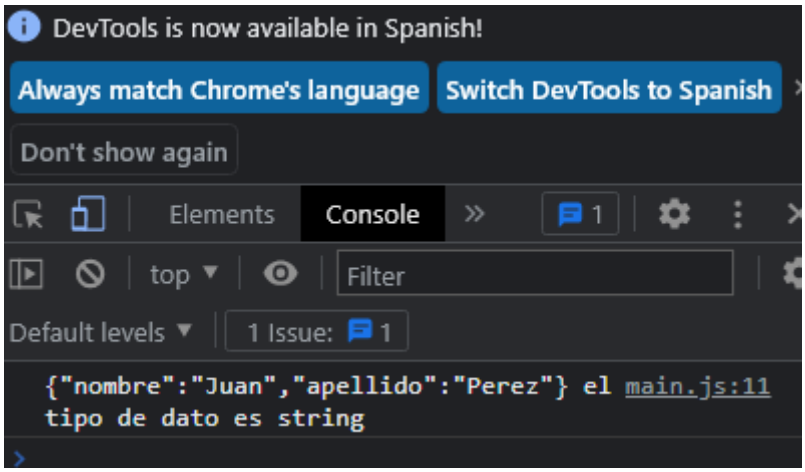
Como dijimos en un principio, en Storage,el dato es siempre una cadena de texto. Es por eso, que existen métodos de conversión para poder trabajar con el tipo de dato correcto.

Usamos el método JSON.stringify() para convertir un objeto en una cadena de texto JSON:

```
let objConvertido = JSON.stringify({nombre: "Juan", apellido:
"Perez"});

console.log(objConvertido, "el tipo de dato es " + typeof
objConvertido);
```

y por consola podemos ver el siguiente resultado:



De la misma manera, al llamar un dato guardado en Storage, este siempre va a ser tipo string. Podemos convertir ese valor en tipo JSON a través del método JSON.parse().

Utilizamos el ejemplo anterior, la variable objConvertido es de tipo string, pero ahora queremos que vuelva a ser un objeto:

```
let objJSON = JSON.parse(objConvertido);

console.log(objJSON , "el tipo de dato es " + typeof objJSON);
```

Verificamos por consola, que efectivamente cambió el tipo de dato: