LocalStorage

¿Qué es localStorage?

Es una parte del objeto Window que permite a los sitios web almacenar datos de manera persistente en el navegador del usuario. A diferencia de las cookies, los datos almacenados en localStorage no se envían al servidor con cada solicitud. Esto lo convierte en una solución eficiente y segura para almacenar datos que solo necesitan ser accesibles en el lado del cliente.
Los datos en localStorage permanecen almacenados incluso después de cerrar el navegador, a menos que sean explícitamente eliminados por el usuario o por el sitio web.
Algunas características son:

- **Persistencia**: Los datos guardados en localStorage permanecen almacenados incluso después de cerrar el navegador.
- Tipo de datos: localStorage solo admite guardar cadenas de texto .
- **Capacidad de almacenamiento**: Generalmente, puede almacenar hasta 5MB de datos, lo cual es mucho más de lo que almacenan las cookies.
- **Acceso sencillo**: Los datos se recuperan fácilmente usando JavaScript, lo que lo hace muy conveniente para aplicaciones web.
- **Seguridad**: Los datos están almacenados localmente en el navegador del usuario y son específicos del dominio, lo que significa que solo los scripts del mismo dominio pueden acceder a ellos.

¿Cuáles son los métodos implementados para gestionar localStorage?

A continuación te contamos cuáles son los métodos más importantes para guardar, leer y eliminar datos del navegador con localStorage:

- **setItem**(key, value): Guarda un par clave-valor en localStorage. Si la clave ya existe, el valor anterior se sobrescribe con el nuevo valor. Tanto la clave como el valor deben ser cadenas de texto.

```
Unset
localStorage.setItem('usuario', 'Juan');
```

- **getItem**(key): Recupera el valor asociado a una clave especificada. Si la clave no existe, devuelve null.

```
Unset
let usuario = localStorage.getItem('usuario'); // 'Juan'
```

- **removeItem**(key): Elimina el par clave-valor asociado a la clave especificada.

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

- **clear**(): Elimina todos los pares clave-valor almacenados en localStorage para el dominio del sitio web.

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

¿Qué tipo de datos se pueden guardar en localStorage?

localStorage está diseñado para almacenar datos en formato de texto plano. Esto significa que puede guardar cualquier tipo de dato que pueda ser representado como una cadena de texto (string). Aunque principalmente se utiliza para almacenar cadenas simples, como identificadores, tokens, configuraciones y preferencias del usuario, es posible guardar otros tipos de datos, siempre y cuando se conviertan a una cadena de texto antes de ser guardados.

Para tipos de datos más complejos, como objetos o arrays, se pueden utilizar funciones como **JSON.stringify()** para convertir estos tipos de datos a una cadena de texto en formato JSON antes de guardarlos en localStorage, y **JSON.parse()** para convertirlos de nuevo a su formato original al recuperarlos.

A modo de ejemplo:

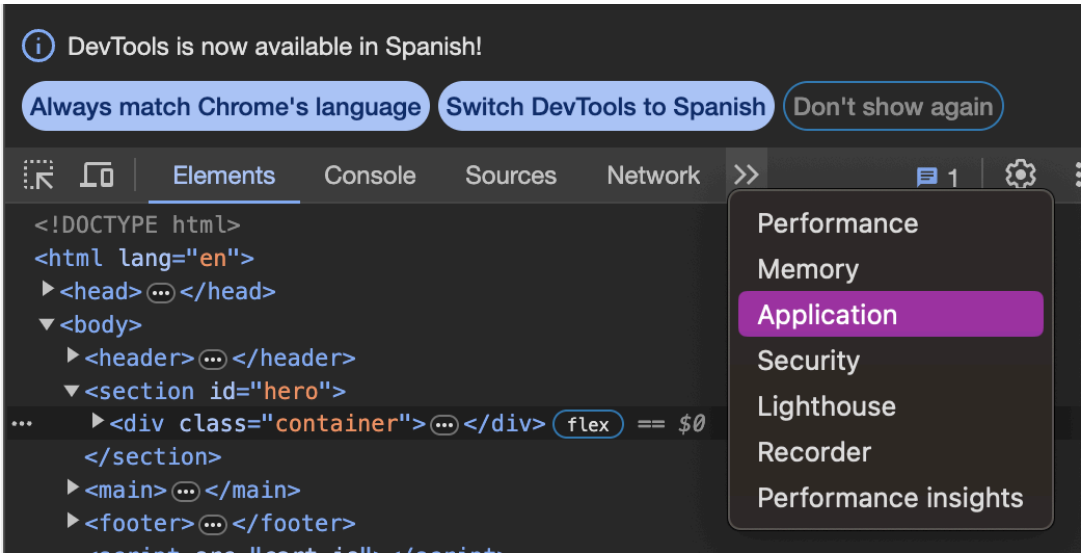
```
Unset
let usuario = { nombre: "Juan", edad: 30 };
localStorage.setItem('usuario', JSON.stringify(usuario));

let usuario = JSON.parse(localStorage.getItem('usuario'));
console.log(usuario.nombre); // "Juan"
```

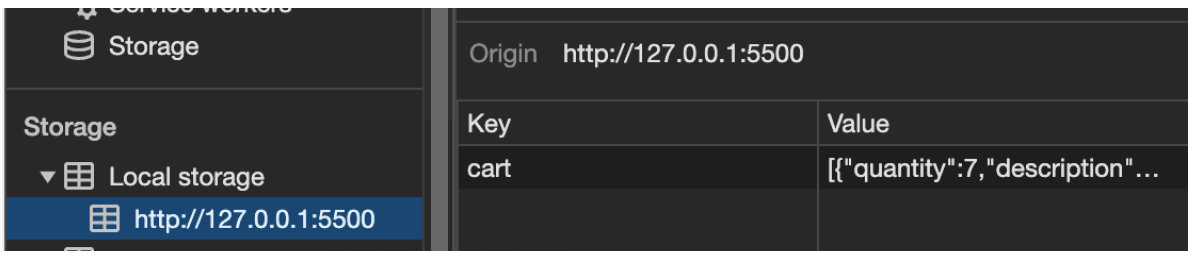
¿Cómo se accede visualmente al localStorage?

Se puede ver el localStorage en tiempo real, como así también modificar y eliminar registros del mismo.Para ello, tendremos que abrir la consola de desarrolladores con clic derecho y seleccionar la opción Inspeccionar.

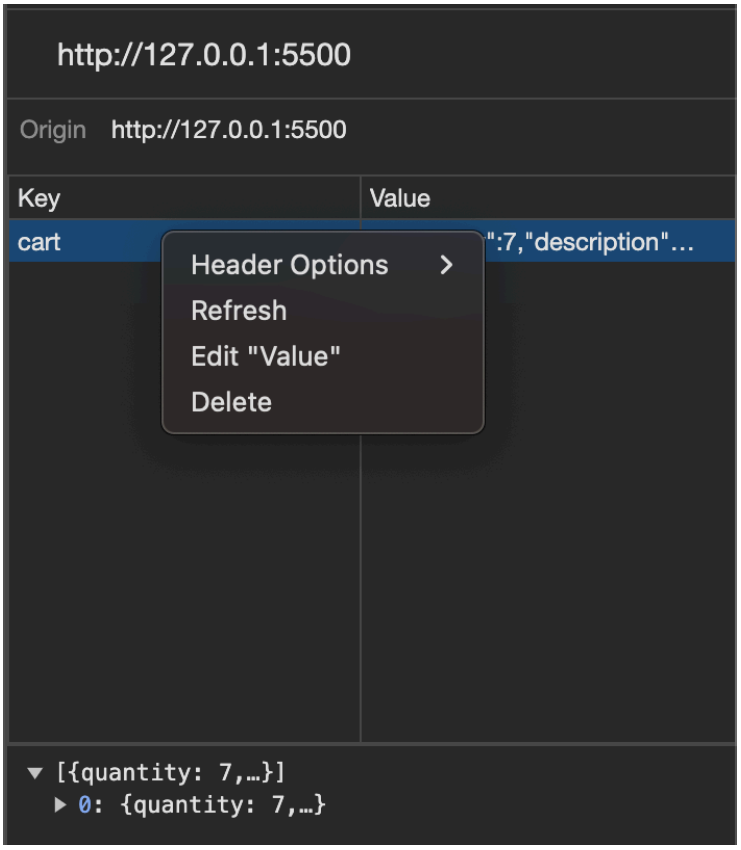
Se nos abrirá una solapa donde tendremos varias sub-pestañas, la que nos importa es Aplicación.



Para seleccionarla, tendremos que apretar el símbolo >> para desplegar todas las opciones. Una vez clickeada esa opción, nos dirigiremos al apartado Local storage



Ahí podrás ver todos los elementos guardados en el localStorage. Si haces clic derecho en cada ítem, podrás modificarlo o eliminarlo. Si haces doble clic en cada ítem, podrás modificarlo.



Abajo, podrás notar que el objeto cart se representa como un objeto JavaScript.

Repaso III - Funciones de Orden Superior

¿Qué es forEach()?

El método forEach se utiliza para ejecutar una función en cada elemento de un array. A diferencia de otros métodos como find o filter, forEach no devuelve un nuevo array ni un valor específico, sino que se utiliza principalmente para efectuar operaciones o efectos secundarios, como modificar elementos o imprimir valores en la consola.
A modo de ejemplo:

```
Unset
const fruits = ["apple", "banana", "cherry"];
fruits.forEach(fruit => console.log(fruit));
// "apple"
// "banana"
// "cherry"
```

¿Para qué se utiliza forEach()?

El método forEach se utiliza para realizar operaciones en todos los elementos de un array, sin necesidad de crear un nuevo array como resultado. Es útil para aplicar una función a cada elemento, por ejemplo, para mostrar en consola cada valor o actualizar una base de datos.

La sintaxis básica del método forEach es la siguiente:

```
Unset
array.forEach((each, index, arr) => { })
```

- **array**: Es el array a iterar y evaluar con el método find.
- **function**(currentValue, index, arr): Es la función que se ejecutará para cada elemento del array.
    - **each**: Es el valor del elemento actual que está siendo procesado en el array.
    - **index** (opcional): Es el índice del elemento actual en el array.
    - **arr** (opcional): Es el array al que pertenece el elemento actual.

Al ser un método que opera mediante efectos secundarios, forEach es ideal para situaciones en las que necesitas aplicar una operación a todos los elementos de un array pero no necesitas un resultado directo de esas operaciones.

A modo de ejemplo::

```
Unset
const numbers = [1, 2, 3];
numbers.forEach((number, index, arr) => {
        arr[index] = number + 1; // Incrementa el valor actual del elemento
});
console.log(numbers); // [2, 3, 4]
```