

M1 - 11 - JS Datos offline

Después de aprender a recibir los datos mediante fuentes externas por el protocolo REST, vamos a ver ahora que tenemos la posibilidad de guardar datos de las páginas web en el propio navegador. Esto nos permite cargar las páginas aún más rápidamente, ya que evitamos tener que volver a descargar elementos que ya tenemos en el ordenador, y permite también la posibilidad de utilizar una aplicación web como si fuera un programa descargado en el ordenador, sin siquiera la necesidad de conectarse a la red.

LocalStorage

Con esta instrucción podemos guardar datos permanentemente en el navegador y estos sobreviven a una actualización de la página e incluso a un reinicio completo del navegador.

Estos datos se van a almacenar en forma de “clave-valor” ambos siempre como cadenas de texto. Almacenamos y obtenemos los datos de localStorage a través de dos métodos:

- `.setItem(key, value)` - Almacenar por clave / valor.
- `.getItem(key)` - Obtener el valor por clave.

setItem()

Con este método accedemos al almacenamiento local (local storage) y añadimos un item. Al `setItem()` le pasamos dos parámetros, dos cadenas concretamente, primero el nombre de la clave y luego el valor que queremos guardar. La clave es el nombre con el que identificamos el valor.

Ejemplo:

Creemos una página con un input para que el usuario introduzca un dato.

```
<input type="text" id="datos" />
<button onclick="guardarDatos()" >Guardar datos</button>
<p id="resultado"></p>
```

s



Cuando este haga click sobre el botón llamamos a una función de JavaScript que va a recoger ese valor que ha introducido el usuario y a través de `localStorage.setItem()` vamos a guardar este valor en el almacenamiento local (local storage).

Para ello, le pasamos primero la clave con la que vamos a identificar dicho valor, le podéis llamar como queráis, yo lo voy a identificar como “datos”, y después, separado por una coma “,” el valor que está almacenada en la variable “datos”.

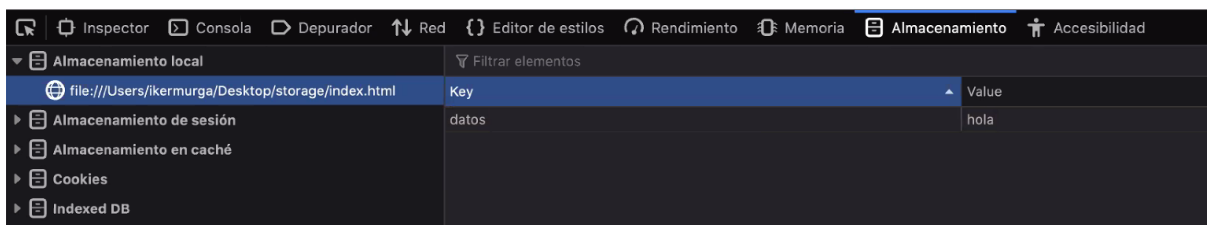
```
<script>
  let datos;
  function guardarDatos() {
    datos = document.getElementById('datos').value;
    document.getElementById('resultado').innerHTML = datos;
    localStorage.setItem('datos', datos); // nombre de la clave, el valor que queremos guardar
  }
</script>
```

Para comprobar que se ha almacenado correctamente, vamos a abrir la página en el navegador y vamos a acceder al inspector. En el inspector encontramos una pestaña de almacenamiento. Al acceder a esta pestaña vemos como a la izquierda nos aparecen varias opciones, nosotros haremos click sobre “almacenamiento local” que es dónde hemos guardado nuestro Item.

Cuando abrimos el almacenamiento local vemos cómo aparece un elemento con la clave “datos” y su valor que se corresponde con lo que haya escrito el usuario dentro del input.

[Página 2](#)

hola



getItem()

El valor ha quedado guardado en el almacenamiento local (local storage) y podemos acceder a él siempre que queramos a través del método `getItem()`.

Este método recibe un parámetro, una cadena de texto concretamente, comprobará en localStorage si hay alguna clave que coincida con esa cadena que le hemos pasado.

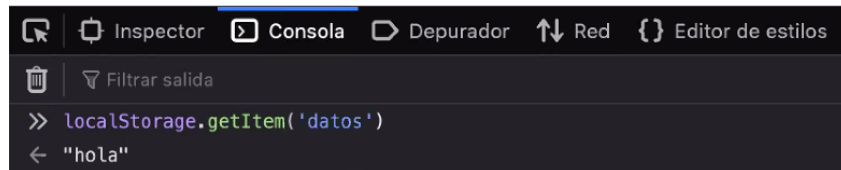
Si en localStorage existe una clave que coincida con ese string (cadena de texto) que le hemos pasado, entonces nos devuelve su valor, es decir, el valor que hemos identificado con dicha clave. Por el contrario, si no encuentra ninguna clave con ese nombre nos devuelve **null** (nulo).

Esto significa que yo puedo acceder a los datos almacenados en localStorage y la ventaja que tiene es que aunque yo cierre el navegador o recargue la página, esos datos siguen

estando en el almacenamiento local, por lo tanto, podemos acceder a ellos cuando queramos.

[Página 2](#) Guardar datos

hola



The screenshot shows a web application interface at the top with a link "Página 2", a text input field containing "hola", and a "Guardar datos" button. Below the interface is a dark-themed developer console. The console has tabs for "Inspector", "Consola", "Depurador", "Red", and "Editor de estilos", with "Consola" selected. A search bar labeled "Filtrar salida" is present. The console log shows a command `>> localStorage.getItem('datos')` followed by the output `<- "hola"`.

```
>> localStorage.getItem('datos')  
<- "hola"
```

Almacenar objetos y arrays en localStorage

El problema que nos vamos a encontrar es que estamos almacenando una clave y un valor, es decir, si volvemos a introducir cualquier cosa en el input, efectivamente ese valor se guardará en localStorage como “datos” pero sobrescribirá el valor que tenía esta clave anteriormente. Pensad que es similar a si tuviéramos una variable en javascript a la cual le decimos que su valor es ‘hola’:

```
let datos = 'hola';
```

En el momento que le damos otro valor estamos sobrescribiendo ese valor :

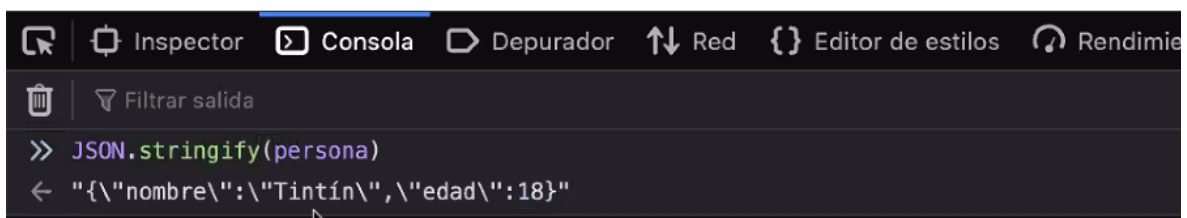
```
datos = 'adiós';
```

Algo similar pasa en localStorage, nosotros almacenamos un valor al cual identificamos con una clave, si después, almacenamos otro valor con esa misma clave lo que hacemos es sobrescribir el valor de dicha clave que ya estaba en localStorage con lo que perdemos los datos anteriores y sólo guardamos el último.

Pero por ejemplo, ¿Si quisiéramos guardar las películas favoritas del usuario? O creamos una clave para cada valor, algo poco práctico, o creamos una lista de películas que vamos almacenando en localStorage.

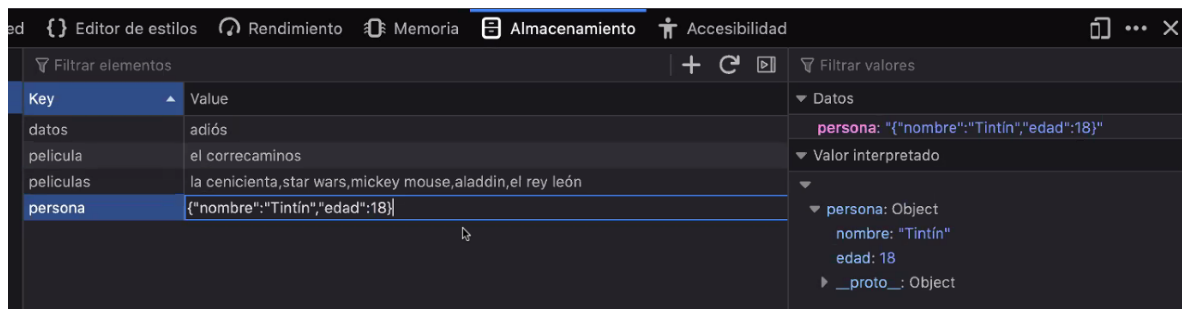
Para almacenar listas y objetos en localStorage la sintaxis que utilizamos es algo diferente ya que sólo podemos almacenar aquí bajo sus restricciones, estas son, el formato clave: valor con dos strings, uno para la clave y otro para el contenido. Tenemos que convertir, por tanto, el objeto o el array en formato texto (en formato JSON) para poder almacenarlo, y para eso utilizamos `JSON.stringify()`. Crearemos un objeto persona que tendrá dos propiedades, nombre y edad:

```
let persona = {  
  nombre: 'Tintín',  
  edad: 18  
}
```



```
>> JSON.stringify(persona)  
← "{\"nombre\": \"Tintín\", \"edad\": 18}"
```

```
>> localStorage.setItem('persona', JSON.stringify(persona))
```



Pero ojo, esto significa que cuando accedemos a este objeto desde javascript con el método `getItem()`, nos va a devolver el objeto en el mismo formato , en formato JSON (string) , y por eso tendremos que pasarlo a un objeto javascript, y para ello utilizaremos el método `JSON.parse()`.

```
>> localStorage.getItem('persona')
< '{"nombre\\":\\"Tintín\\",\\"edad\\":18}'

>> JSON.parse(localStorage.getItem('persona'))
< ▶ Object { nombre: "Tintín", edad: 18 }
```

Siguiendo con el ejemplo anterior en el que le pedíamos datos al usuario vamos a modificarlo un poco para ir pidiéndole películas e ir guardando en localstorage una lista con las películas que el usuario va introduciendo.

Para ello, inicializamos un array vacío dónde vamos a ir añadiendo las películas. Cuando el usuario introduce una película y le da al botón de añadir, llamamos a una función de JavaScript que va a recoger el valor introducido por el usuario en el input, que será el nombre de la película e introducimos ese valor en el array. Una vez introducido el nombre en el array, incluiremos este array en local storage con el método `setItem()` y utilizando `JSON.stringify()` para pasar este array a formato JSON.

```
<body>
  <input type="text" id="pelicula" />
  <button onclick="guardarPelicula()">
    Añadir película
  </button>

  <script>
    let listado = [];
    function guardarPelicula() {
      let titulo = document.getElementById('pelicula').value;
      window.alert(titulo);
      listado.push(titulo);
      localStorage.setItem('listado', JSON.stringify(listado));
    }
  </script>
</body>
```

Los distintos valores que vaya introduciendo el usuario se van a ir almacenando en el array, y por lo tanto al almacenar este array en localStorage tendremos acceso a todos los valores que vaya introduciendo el usuario.

The screenshot shows a web application interface at the top with a text input containing the word "adiós" and a button labeled "Añadir película". Below the input is a light blue button labeled "Añadir". The bottom half of the image shows a browser's developer console with the "Consola" tab selected. The console displays three log entries for a variable named "listado":

```
>> listado  
← ▶ Array []  
  
>> listado  
← ▶ Array [ "hola" ]  
  
>> listado  
← ▶ Array [ "hola", "adiós" ]
```