



# LocalStorage

clase n° 20

\*REC



recordá  
poner a grabar la clase

# LocalStorage

**LocalStorage** es una propiedad ofrecida por nuestro navegador para poder **almacenar datos de manera local** dentro de mismo, por lo que nos permitirá que este perdure más en el tiempo y sea reutilizable en distintos tipos de casos.

Esta propiedad la cual es un objeto con diversas funciones, nos ofrece de manera sencilla 2 las cuales son para poder almacenar un valor como también para poder consultar el mismo.

Esto lo realizaremos con las siguientes funciones:

■ LocalStorage.setItem()

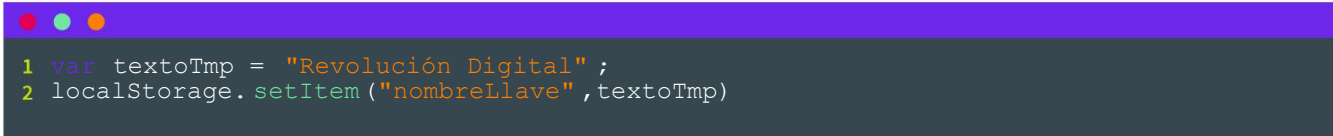
■ LocalStorage.getItem()

# ¿cómo funciona?

Para empezar vamos a explicar cómo funciona el LocalStorage, este almacena mediante una clave (popularmente conocida como key) el valor que nosotros deseemos. Pero tiene una excepción muy grande, los valores a almacenar preferentemente deben ser de un tipo sencillo, dígame de tipo numérico, texto, booleano o vacíos. Entonces se preguntaran ¿Cómo almacenamos variables de tipo un poco más complejos como arrays y objetos?

Para eso vamos a esperar al siguiente tema el cual aprenderemos una función nueva para hacer este paso

Mientras tanto aprendamos como almacenar variables o valores simples

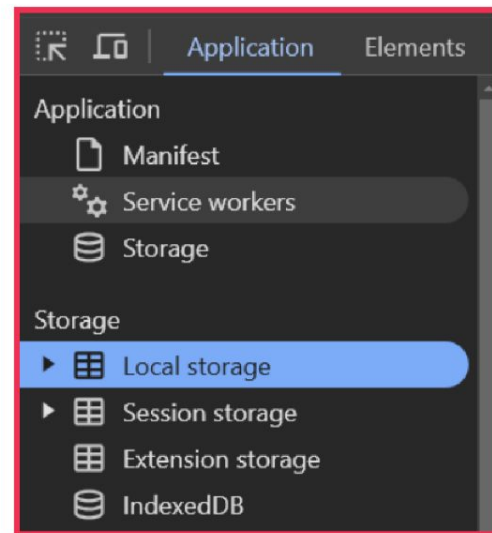


```
1 var textoTmp = "Revolución Digital";  
2 localStorage.setItem("nombreLlave", textoTmp)
```

Como pueden ver en el código anterior, declaramos la variable `textoTmp` y debajo damos la orden que al objeto `LocalStorage` le utilizamos `setItem()`, el cual tiene 2 parámetros como dijimos previamente, la key la cual es `nombreLlave` y después como 2do recibe `textoTmp`

Con esta línea logramos almacenar dentro de nuestro almacenamiento local del navegador el texto de “Revolución Digital”. Para checkearlo podemos ingresar al apartado de

Application (Aplicación) => Storage (Almacenamiento) => LocalStorage



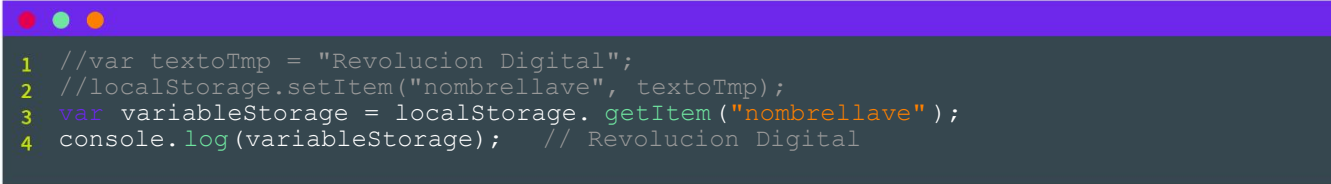
Desde consola podremos acceder a ver que estamos almacenando dentro de nuestro **LocalStorage** como también a otros espacios de memoria ofrecidos por el navegador

Una vez que aprendimos como poder almacenar un valor dentro del LocalStorage, debemos aprender como leer u obtener el mismo, ya sea para trabajar, mostrar, modificar, etc. con su valor. Para eso vamos a aprender a utilizar la siguiente función nombrada en un principio.



# ¿Cómo obtengo un valor de LocalStorage?

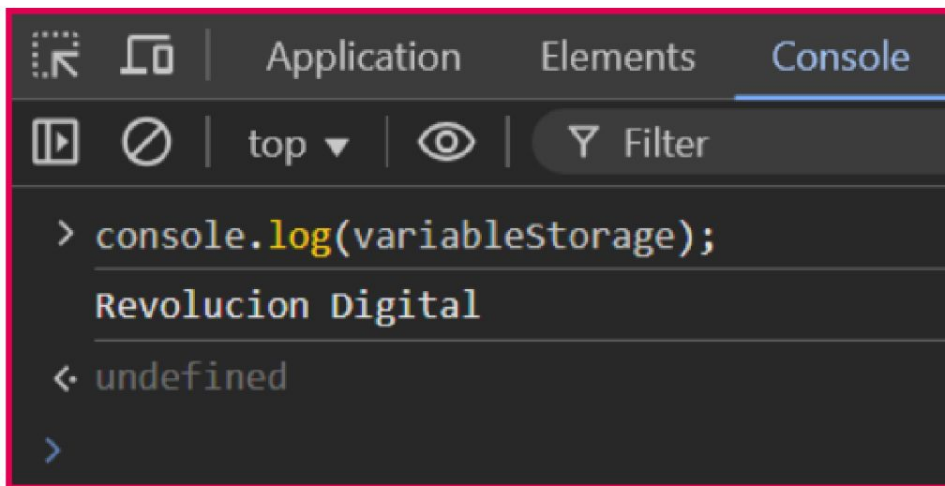
Para poder obtener un valor que está guardado dentro del localStorage vamos a utilizar la función del mismo llamada **getItem()**, esta solamente recibe un parámetro y este será la key con la identificamos el valor. Como en el ejemplo anterior el cual la nombramos como **nombreLlave**, quedaría de la siguiente manera.

A code editor window with a purple title bar and three colored window control buttons (red, green, yellow) on the left. The code is written in a dark-themed editor with syntax highlighting. The code consists of four lines: a comment, a localStorage.setItem call, a localStorage.getItem call, and a console.log call.

```
1 //var textoTmp = "Revolucion Digital";  
2 //localStorage.setItem("nombrellave", textoTmp);  
3 var variableStorage = localStorage.getItem("nombrellave");  
4 console.log(variableStorage); // Revolucion Digital
```

Como podemos ver, esta funcion al retornar el valor almacenado podremos guardarlo dentro de una variable, la cual después si le hacemos un **console.log()** podemos ver que se imprime en consola el valor de **“Revolucion Digital”**.





Con esto ya sabremos trabajar en el almacenamiento de valores y la consulta de los mismos dentro de LocalStorage



# parseo de JSON

Para empezar debemos entender a qué se le llama un “**Parse**” y cuál es su objetivo, primero que nada, “**parse**” hace referencia a la palabra del inglés la cual se entiende como un análisis, pero en la programación esta palabra tendrá como significado analizar y convertir un valor o variable en un formato interno que un entorno pueda interpretar. Para eso tendremos un formato “universal” el cual nos ayuda a interactuar el cual es **JSON** (Javascript Object Notation) ya que no depende del lenguaje que se esté utilizando ni de un formato predefinido

Ahora pasaremos a explicar cómo debemos trabajar con este formato y además relacionarlo con el almacenamiento de información dentro de **LocalStorage**.

## **JSON.stringify()**

Javascript nos ofrece muchas funcionalidades y unas de ellas es poder pasar un objeto o array a formato JSON, para eso podemos utilizar la función **Stringify** que se encuentra dentro del objeto JSON. Se realiza de la siguiente manera

```
1 // Declaramos un Objeto ó Array
2 var auto = {
3   puertas: 4,
4   color: "Rojo",
5   patente: "APJ204"
6 };
7 // Pasamos auto a JSON y lo almacenamos
8 var autoJSON = JSON.stringify(auto);
```

Si realizamos un **console.log()** de **autoJSON** se vera de la siguiente manera

```
> console.log(autoJSON)
console.log(auto)

{"puertas":4,"color":"Rojo","patente":"APJ204"}

▼ {puertas: 4, color: 'Rojo', patente: 'APJ204'} ⓘ
  color: "Rojo"
  patente: "APJ204"
  puertas: 4
  ► [[Prototype]]: Object
```

En la imagen vemos una comparativa entre **autoJSON** y auto, entre que uno es un objeto pasado a texto para decirlo de una manera más simple y otro es un objeto de Javascript

Ahora, que pasaría si quisiéramos hacer el camino inverso, ¿Cómo pasamos un JSON a un objeto para poder manipularlo como tal? Eso se hace con otra función que nos ofrece el objeto JSON.

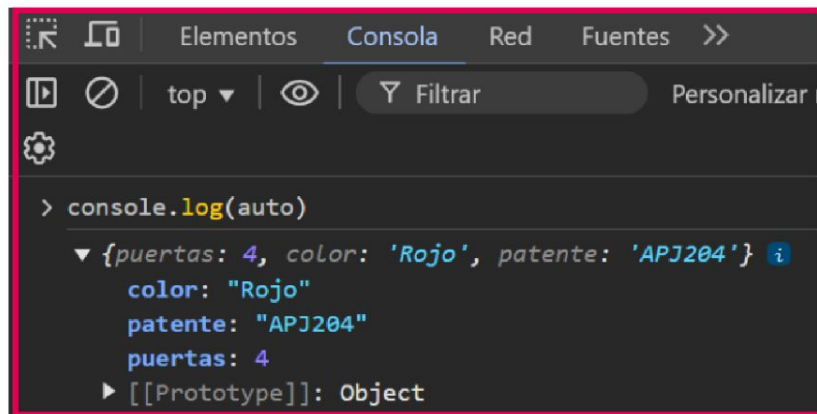
### **JSON.parse()**

Como vimos previamente, al pasar un objeto a JSON básicamente este pasa a ser un texto el cual contiene la información importante pero no se interpreta como tal. Para poder revertir este cambio o pasar un JSON a objeto en Javascript vamos a utilizar la función del objeto JSON llamada **parse()**, esta función recibe como único parámetro el JSON que queramos transformar y su retorno será el objeto con sus propiedades y valores.

Hagamos el ejemplo visto en **JSON.stringify()** pero a la inversa.

```
1 // Declaramos el JSON
2 var autoJSON = '{"puertas":4,"color":"Rojo","patente":"APJ204"}' ;
3 // Pasamos autoJSON a objeto y lo almacenamos
4 var auto = JSON.parse(autoJSON);
```

Como podemos ver, si realizamos un **console.log()** de auto, ahora podemos tener el objeto almacenado en esta variable, siendo que paso del texto declarado en autoJSON previamente.



Una vez obtenido el nuevo objeto podremos acceder nuevamente a todas las funcionalidades y propiedades ofrecidas por los objetos que nos da Javascript.



revolución\*  
digital\_