

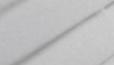
ADM

Dudas?



ADM

Dudas



LocalStorage

- Forma parte (al igual que sessionStorage) de las Apis HTML5.
- LocalStorage nos permite lograr la persistencia de datos en la aplicación.
- Espera una **clave** y un **valor** para la información.
- Es un objeto Javascript, por lo tanto tiene **métodos** y **una propiedad**.
- Estos **métodos** nos permiten hacer distintos tipos de cosas: setear elementos, obtener los ya guardados, borrar items o todo lo que posea localStorage.

LocalStorage

Métodos

- **localStorage.setItem(clave, valor);**

cuando reciba una clave y un valor, añadirá estos al almacenamiento, o actualizará el valor si la clave ya existe.

- **localStorage.getItem(key);**

devuelve el valor de la clave cuyo nombre se le pasa por parámetro.

Si la clave no existe, devuelve **null**.

- **localStorage.removeItem(key);**

elimina la clave cuyo nombre recibe por parámetro del almacenamiento.

LocalStorage

- **localStorage.clear();**

elimina todas las claves que haya en almacenamiento.

- **localStorage.key(0);**

nos permite saber las claves que tenemos guardadas. como argumento pondremos el **número** de elemento de la clave.

Y la propiedad ...

- **localStorage.length;**

nos devuelve el **número** de elementos guardados.

LocalStorage - JSON



Limitación → Sólo podemos guardar *cadenas de texto*, es decir, sólo podemos almacenar datos de tipo *string*.

- Si queremos guardar booleanos, números, etc ? Tendremos que *usar funciones de parseo de datos* para manipular esos tipos de datos.
- Para poder guardar esta información necesitamos usar el **formato JSON**
- **JSON** significa **JavaScript Object Notation**.
- Sirve para representar datos como **atributos con valores**, derivado originalmente de la sintaxis de **JavaScript** .
- **JSON** proporciona un medio ideal para encapsular los datos entre el cliente y el servidor.

LocalStorage - JSON

Serializar - Deserializar

- JSON se usa para **representar objetos** (o estructuras de datos) con **texto serializado** (proceso de *codificación de un objeto en un medio de almacenamiento*, por ejemplo, en un archivo, o guardar en localStorage)
- En complemento la **deserialización** consiste en *convertir una secuencia en un objeto.*
- *Ambos permiten almacenar y transferir fácilmente datos.*
- Los objetos **serializados** podrían guardarse en un archivo de texto plano.
- Si tenemos que consultar alguna representación debemos "**deserializar**" (reconstruir) *el archivo de texto plano a objetos,*
- Para estos procesos JSON posee 2 funciones: **JSON.stringify()** y **JSON.parse()**

LocalStorage - JSON

JSON.stringify() -> Recibe por parámetro un array y devuelve un string en formato JSON - (datos en string)

JSON.parse() -> Recibe por parámetro el string y devuelve un array.

Entonces ...

- Si queremos guardar en localStorage datos tenemos que aplicar ***JSON.stringify()*** (setter)
- Si queremos mostrar la información (requiere iterar), necesitamos usar ***JSON.parse()***. (getter)

LocalStorage - clave,valor

LocalStorage

Nombre:

Guardar

- Este formulario guardará el **valor** que ingrese el usuario.
- Nosotros le crearemos un **key** para guardar en localStorage.
- El botón **enviar** disparará la **funcion enviar_nombre()**

- Para manipular datos en localStorage tenemos 3 maneras :

1

- *Usando el método **setItem**, guardamos con la **clave nombre** y le pasamos el **valor** del input para que sea su valor.*
- *El método **getItem**, pide la clave llamada **nombre** desde localStorage y lo guardamos en **dato_recuperado_local**, para mostrarlo en el HTML*

LocalStorage - clave,valor

```
function enviar_nombre(){  
    const nombre = document.querySelector("#nombre").value;  
    localStorage.setItem ("nombre", nombre);  
    const mostrar = document.querySelector("#mostrar")  
    const dato_recuperado_local = localStorage.getItem ("nombre");  
  
    mostrar.innerHTML = dato_recuperado_local;  
}
```

2

- Las claves se almacenan en un array y son consideradas como propiedades del objeto localStorage. (Lo podemos guardar como array)

```
localStorage["nombre"] = nombre; // Guardamos en LocalStorage
```

```
var dato_recuperado_local = localStorage["nombre"]; // Pedimos lo guardado
```

LocalStorage - clave,valor

3

- También podemos referenciar las claves como propiedades del objeto localStorage.

```
localStorage.nombre = nombre;
```

// Guardamos en LocalStorage

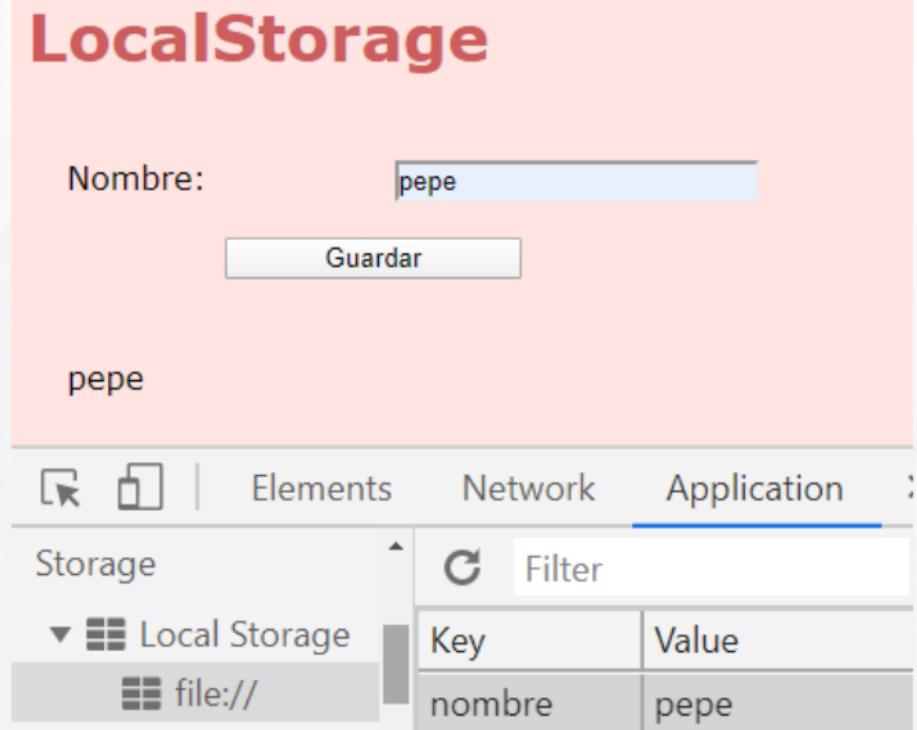
```
var dato_recuperado_local = localStorage.nombre; // Pedimos lo guardado
```

- Cualquiera de las 3 maneras que usemos obtendremos el mismo comportamiento.

LocalStorage

Nombre:

pepe



Key	Value
nombre	pepe

LocalStorage

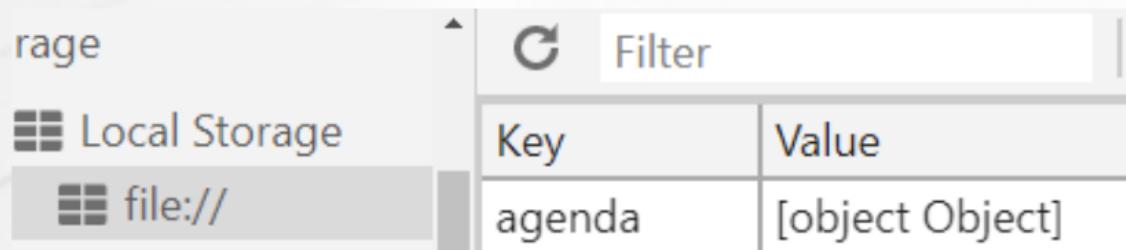
- Y si queremos guardar más datos, y otros tipos como number, boolean, object, etc?
- Podríamos guardar los datos en un objeto.
- Definir en el objeto las **propiedades** y los **valores** que nos llegan desde el html



```
js
1 const nombre = document.querySelector("#nombre").value;
2 const apellido = document.querySelector("#apellido").value;
3 const telefono = document.querySelector("#telefono").value;
4
5 const data = { nombre, apellido, telefono, };
6
7 localStorage.setItem("agenda", data);
```

LocalStorage

- Si queremos guardar esto la consola nos mostrará ...

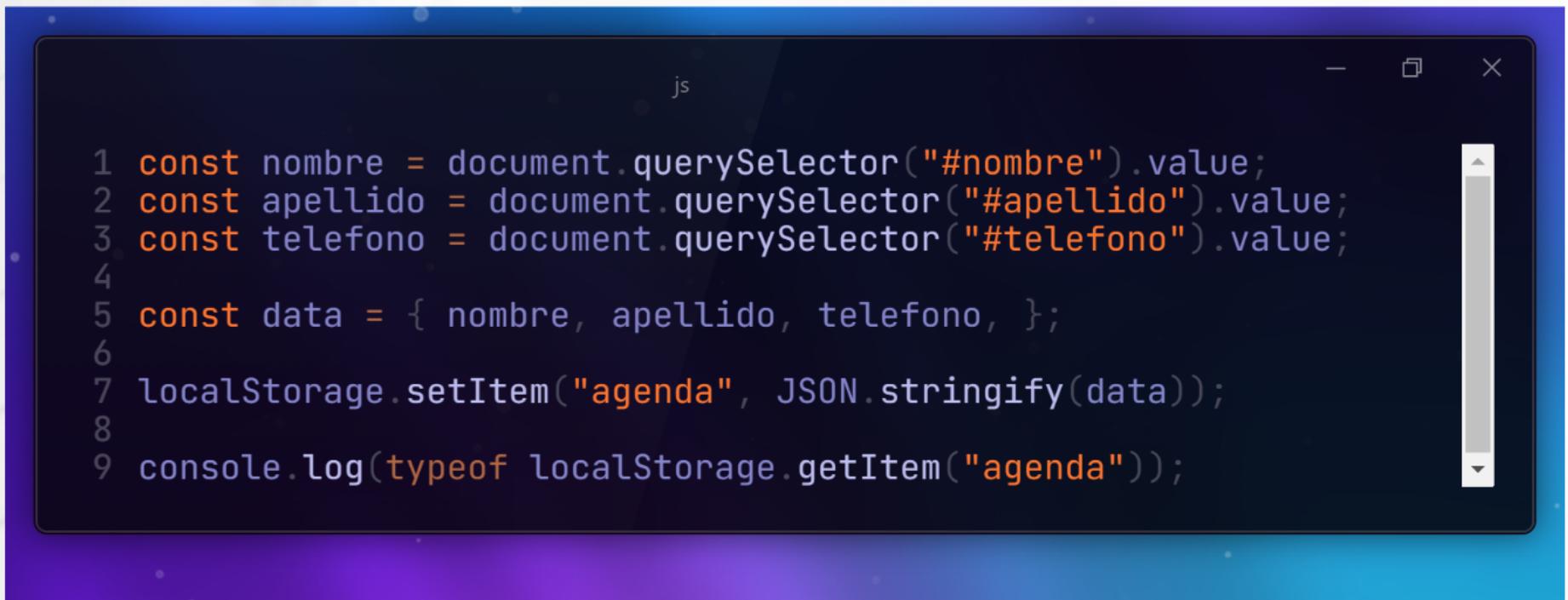


The screenshot shows the 'LocalStorage' tab in the Chrome DevTools Storage panel. The table has two columns: 'Key' and 'Value'. There is one item: 'agenda' with value '[object Object]'. A 'Filter' input field is at the top right.

Key	Value
agenda	[object Object]

- Cómo hacemos para guardar el objeto con las propiedades y valores que realmente ingresó el usuario?
- Las función de parseo **JSON.stringify** nos va a permitir convertir este objeto en un string para poder guardarlo en localStorage .

LocalStorage



```
js

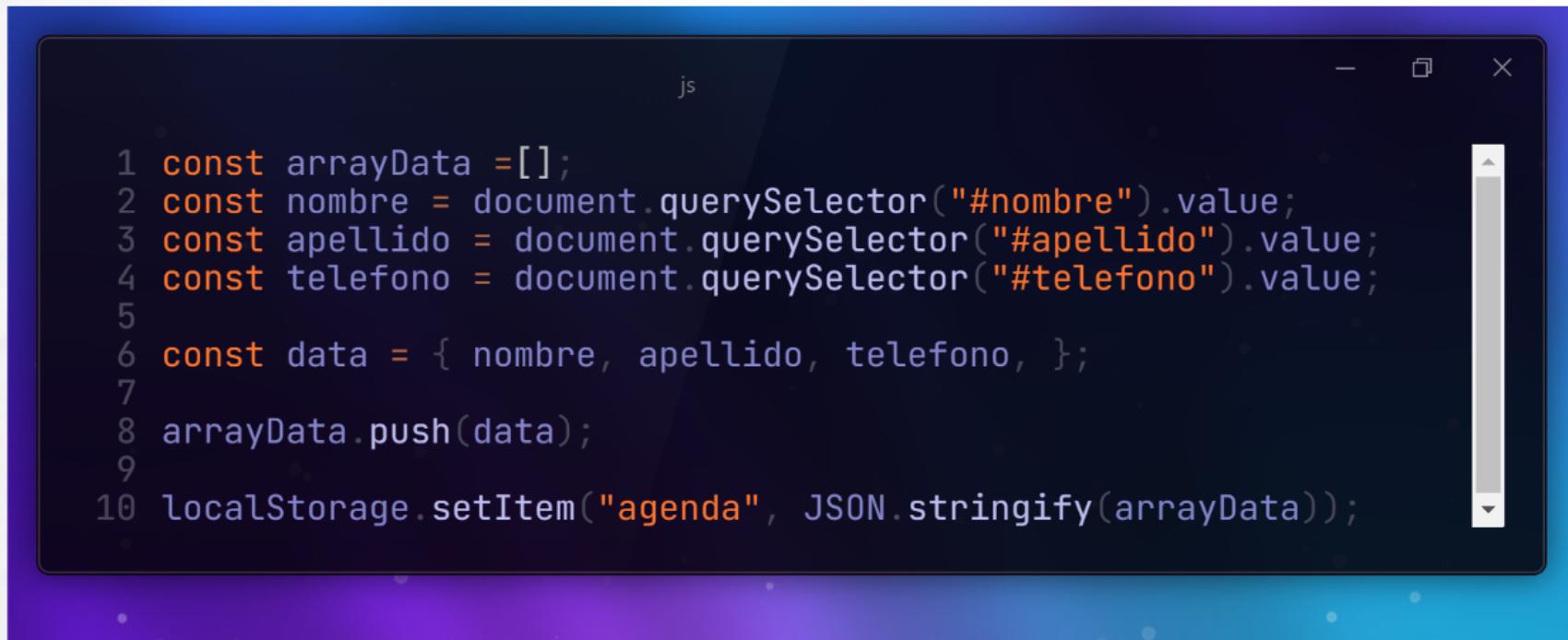
1 const nombre = document.querySelector("#nombre").value;
2 const apellido = document.querySelector("#apellido").value;
3 const telefono = document.querySelector("#telefono").value;
4
5 const data = { nombre, apellido, telefono, };
6
7 localStorage.setItem("agenda", JSON.stringify(data));
8
9 console.log(typeof localStorage.getItem("agenda"));
```

- La consola nos devolverá que el tipo de dato que tenemos con el key agenda, ahora sí, es un **string**.
- Y veremos que **los datos se almacenan correctamente de manera local**.

Local Storage	Key	Value
file://	agenda	{"nombre":"pepe","apellido":"perez","telefono":"555555...}

LocalStorage

- *Cada vez que se envie el formulario se pisarán los datos. Cómo hacemos para guardar más usuarios ??*
- Podríamos crear un array y cada vez que se envíe el fomulario generar un objeto con un nuevo usuario y meterlo al array.



```
js

1 const arrayData = [];
2 const nombre = document.querySelector("#nombre").value;
3 const apellido = document.querySelector("#apellido").value;
4 const telefono = document.querySelector("#telefono").value;
5
6 const data = { nombre, apellido, telefono, };
7
8 arrayData.push(data);
9
10 localStorage.setItem("agenda", JSON.stringify(arrayData));
```

- Con esta lógica, al ingresar un nuevo usuario, perdemos la data del usuario anterior. Por lo tanto, nos *falta un poco más de lógica*.

LocalStorage

- Necesitamos comprobar si existe el key **agenda** en **localStorage** y en base a eso tomar decisiones.

```
js

1 if (!localStorage.getItem("agenda")){
2     const arrayData = [];
3 }else{
4     arrayData = localStorage.agenda;
5 }
6
7 const nombre = document.querySelector("#nombre").value;
8 const apellido = document.querySelector("#apellido").value;
9 const telefono = document.querySelector("#telefono").value;
10
11 const data = { nombre, apellido, telefono, };
12
13 arrayData.push(data);
14
15 localStorage.setItem("agenda", JSON.stringify(arrayData));
```

LocalStorage

Traducción a nivel humano :

- Si **no existe** en localStorage el key agenda, **creame un array**.
- Si **tengo el key agenda**, en **arrayData**, **dame lo que tenga**.
 - **Metele** (por medio del push) al **array**, **cada uno de los objetos** que **van llegando**.
 - Por último, **convertime todo esto en string**, y lo guardás en **localStorage**.

LocalStorage

- Si lo probamos en el navegador **la consola nos tirará error en este código**, debido a un elemento no implementado en la lógica.

string

✖ ► Uncaught TypeError: arrayData.push is not a function
at enviar2 ([script.js:15:14](#))
at HTMLButtonElement.onclick ([index.html:36:45](#))

- Esto nos indica que seguimos trabajando con tipo de dato string, y **no podemos hacer un push a un string**.
- Nos faltó parsear esa información nuevamente para recuperar el tipo de dato real array, usando **JSON.parse**.

LocalStorage

- Vamos a aprovechar y mejorar la sintaxis :



```
js

1 const agenda = localStorage.getItem("agenda");
2 const arrayData = agenda ? JSON.parse(agenda) : [];
3
4 const nombre = document.querySelector("#nombre").value;
5 const apellido = document.querySelector("#apellido").value;
6 const telefono = document.querySelector("#telefono").value;
7
8 const data = { nombre, apellido, telefono, };
9
10 arrayData.push(data);
11 localStorage.setItem("agenda", JSON.stringify(arrayData));
12
13 document.querySelector("form").reset();
14
15 mostrar();
```

- De esta manera **sí**, podremos hacer el **push** de los objetos al array, y **no** arrojará error la consola.

LocalStorage

- Para poder mostrar la información almacenada en localStorage, nos falta Crear la función y recorrer la variable que recupera la info guardada de localStorage.

```
js

1 const agenda = localStorage.getItem("agenda");
2 const recuperar_localStorage = agenda ? JSON.parse(agenda) : [];
3
4 let guardar = '';
5 recuperar_localStorage.forEach(contacto => {
6     guardar += `<p class="datos">
7         <span> ${contacto.nombre}</span>
8         <span> ${contacto.apellido}</span>
9         <span> ${contacto.telefono}</span>
10    </p>`;
11 });
12
13 document.querySelector("#mostrar").innerHTML = guardar;
```

LocalStorage

Formulario

Nombre	<input type="text" value="Thor"/>
Apellido	<input type="text" value="García"/>
Teléfono	<input type="text" value="1567778990"/>
<input type="button" value="Enviar"/>	



Ingreso de datos

Datos desde LocalStorage

Randy Sluvis 1567778989



Muestra de datos

The screenshot shows the Chrome DevTools Application tab with the Local Storage panel selected. It displays a table with one key-value pair:

Key	Value
agenda	[{"nombre": "Randy", "apellido": "Sluvis", "telefono": "1567778989"}, {"nombre": "Thor", "apellido": "García", "telefono": "1567778990"}]

Almacenamiento
LocalStorage

Dudas?



ADM

Dudas

