Diplomatura en

GLOBAL ACADEMY

FullStack Developer

JavaScript
JSON, localStorage

JSON

- JSON (JavaScript Object Notation) es un formato de texto ligero para el intercambio de datos.
- Es un subconjunto de la notación de objetos de JavaScript, lo que lo hace fácil de entender y utilizar en aplicaciones web.
- Aunque JSON proviene de JavaScript, es un formato de texto que es independiente del lenguaje y se puede utilizar en muchos otros lenguajes de programación como Python, Java, C#, PHP, y más.

```json
{
    "nombre": "Juan Pérez",
    "edad": 30,
    "email": "juan.perez@example.com",
    "direccion": {
        "calle": "Calle Falsa 123",
        "ciudad": "Ciudad Ejemplo"
    },
    "telefonos": [
        "123-456-7890",
        "098-765-4321"
    ]
}
```

Sintaxis de JSON

Objetos:
Definidos por llaves { }
Contienen pares clave-valor

```
{
    "nombre": "Juan Pérez",
    "edad": 30
}
```

Arrays:

Definidos por corchetes [ ]Contienen una lista ordenada de valores

```
[
    "Rojo",
    "Verde",
    "Azul"
]
```

Comparación con objetos JavaScript:

JSON:

- Formato de texto
- Claves siempre en comillas dobles "
- Valores pueden ser strings, números, booleanos, null, arrays u objetos

```
{
    "nombre": "Ana",
    "edad": 25,
    "esEmpleado": true,
    "direccion": {
        "calle": "Av. Siempre Viva",
    },
    "hobbies": [
        "Leer",
        "Correr"
    ]
}
```

Objeto JavaScript:

- Definido como código
- Claves pueden estar sin comillas
- Admite funciones y valores indefinidos

```
const persona = {
    nombre: "Ana",
    edad: 25,
    esEmpleado: true,
    direccion: {
        calle: "Av. Siempre Viva",
    },
    hobbies: ["Leer", "Correr"]
};
```

Convertir datos a JSON

Uso de JSON.stringify():

- Método de JavaScript para convertir objetos y arrays en una cadena JSON.
- Muy útil para almacenar datos en un formato legible y transferible.

```
JSON.stringify(valor);
```

```
const persona = {
    nombre: "Juan Pérez",
    edad: 30,
    email: "juan.perez@example.com"
};

const jsonPersona = JSON.stringify(persona);
console.log(jsonPersona);
// Resultado: {"nombre":"Juan Pérez","edad":30,"email":"juan.perez@example.com"}
```

Convertir un array:

```
const colores = ["Rojo", "Verde", "Azul"];

const jsonColores = JSON.stringify(colores);
console.log(jsonColores);
// Resultado: ["Rojo","Verde","Azul"]
```

Aplicaciones comunes:

- Transferencia de datos entre cliente y servidor.
- Almacenamiento de datos en localStorage.
- Envío de datos a través de APIs.

Parsear JSON en JavaScript

Uso de JSON.parse():

- Método de JavaScript para convertir una cadena JSON en un objeto JavaScript.
- Permite trabajar con los datos de manera más estructurada y accesible.

Parsear una cadena JSON simple:

```
const jsonPersona = '{"nombre": "Juan Pérez", "edad": 30, "email": "juan.perez@example.com"}';

const persona = JSON.parse(jsonPersona);
console.log(persona);
// Resultado: {nombre: "Juan Pérez", edad: 30, email: "juan.perez@example.com"}
```

Parsear un array JSON:

```
const jsonColores = '["Rojo", "Verde", "Azul"]';

const colores = JSON.parse(jsonColores);
console.log(colores);
// Resultado: ["Rojo", "Verde", "Azul"]
```

Aplicaciones comunes:

- Recepción de datos desde un servidor.
- Procesamiento de datos almacenados en localStorage o sessionStorage.
- Trabajar con datos obtenidos de APIs.

Introducción a localStorage

- localStorage es una propiedad de la interfaz window que permite almacenar datos de manera persistente en el navegador del usuario.
- Los datos almacenados en localStorage no tienen fecha de expiración y permanecen incluso si el navegador se cierra y se vuelve a abrir.

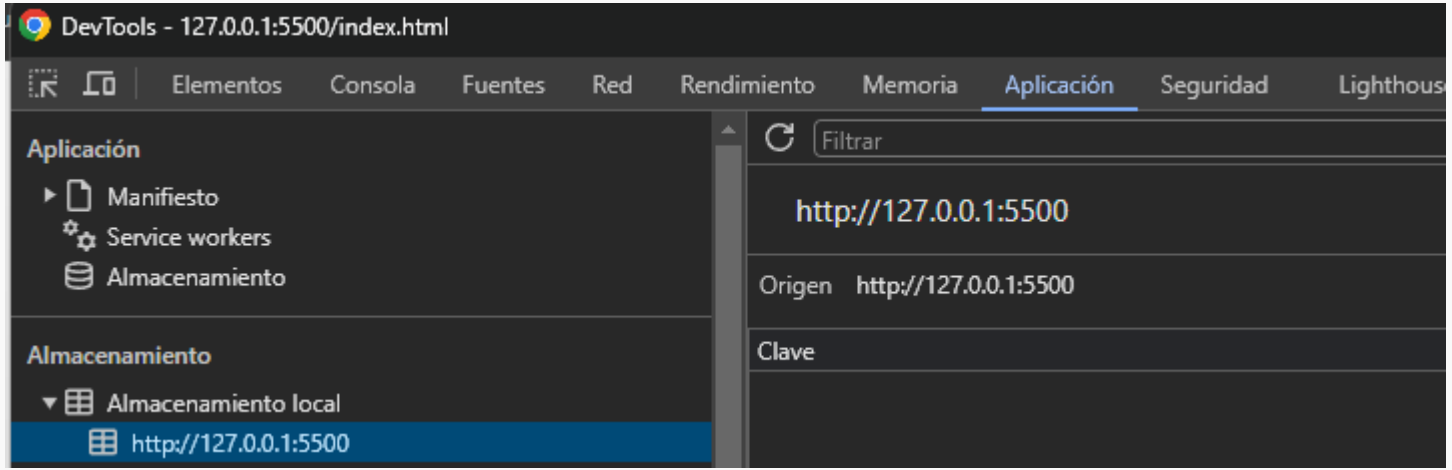
Ventajas de localStorage:

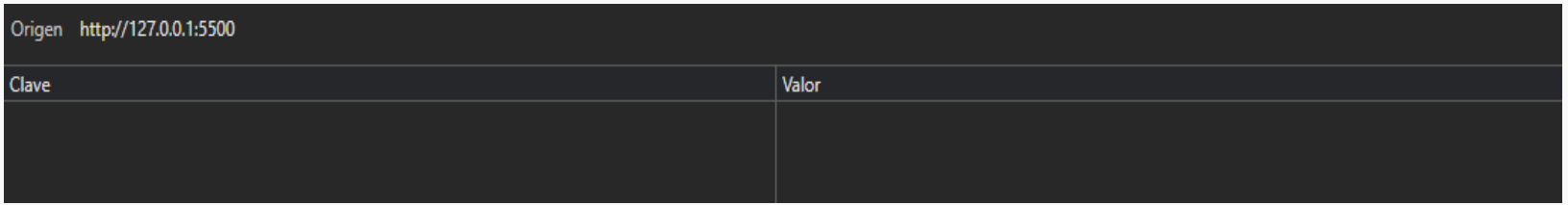
- Fácil de usar: API sencilla con métodos claros (setItem(), getItem(), removeItem(), clear()).
- Rendimiento: Acceso rápido a datos almacenados localmente.
- Sin necesidad de servidor: Los datos se almacenan en el navegador del usuario.

Limitaciones de localStorage:

- Tamaño limitado: 5MB puede no ser suficiente para aplicaciones que requieren almacenamiento intensivo de datos.
- Seguridad: Los datos no están encriptados, por lo que información sensible no debe ser almacenada en localStorage.

- Para poder visualizar los datos que vamos almacenando en el localStorage, usaremos las herramientas de desarrollo que incorporan los navegadores





API de localStorage:

setItem(key, value)

- Guarda un valor bajo una clave específica.
- Si la clave ya existe, el valor será sobrescrito.

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

getItem(key)

- Recupera el valor asociado a una clave.
- Si la clave no existe, retorna null.

```
const nombre = localStorage.getItem('nombre');
console.log(nombre); // "Juan Pérez"
```

removeItem(key)

- Elimina el valor asociado a una clave específica.

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

clear()

- Elimina todos los valores almacenados en localStorage.

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

Ejemplo completo

```
// Guardar datos
localStorage.setItem('nombre', 'Juan Pérez');
localStorage.setItem('edad', '30');

// Recuperar datos
const nombre = localStorage.getItem('nombre');
const edad = localStorage.getItem('edad');
console.log(`Nombre: ${nombre}, Edad: ${edad}`);

// Eliminar un dato
localStorage.removeItem('edad');

// Limpiar todos los datos
localStorage.clear();
```

Guardar datos en localStorage

```
localStorage.setItem('nombre', 'Juan Pérez');
console.log(localStorage.getItem('nombre')); // "Juan Pérez"
```

Método setItem(key, value)

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

localStorage.setItem('edad', 30);

const usuario = {
    nombre: 'Juan Pérez',
    edad: 30,
    email: 'juan.perez@example.com'
};

localStorage.setItem('usuario', JSON.stringify(usuario));


const colores = ['Rojo', 'Verde', 'Azul'];

localStorage.setItem('colores', JSON.stringify(colores));
```

Resultado

Origen http://127.0.0.1:5500

Clave	Valor
usuario	{"nombre":"Juan Pérez","edad":30,"email":"juan.perez@example.com"}
edad	30
colores	["Rojo","Verde","Azul"]
nombre	Juan Pérez

Recuperar datos de localStorage

```
const nombre = localStorage.getItem('nombre');
console.log(nombre);

const edad = localStorage.getItem('edad');
console.log(edad);

const usuarioRecuperado =
JSON.parse(localStorage.getItem('usuario'));
console.log(usuarioRecuperado);

const coloresRecuperados =
JSON.parse(localStorage.getItem('colores'));
console.log(coloresRecuperados);
```

Eliminar datos de localStorage

El método removeItem() elimina un item/clave que recibe por parámetro

```
localStorage.removeItem('nombre');
console.log(localStorage.getItem('nombre')); // null
```

El método clear() elimina todos los ítems/claves que existan en el localStorage

```
localStorage.clear();
console.log(localStorage.getItem('colores')); // null
console.log(localStorage.getItem('edad')); // null
```

Ejercicios

1. Vamos a guardar por medio de un formulario en un array de usuarios que se almacenará en el localStorage y mostraremos en una tabla HTML
2. Guardar la configuración de un usuario, es decir si eligió un tema oscuro para nuestra página, que esta opción se guarde en el localStorage.