JAVASCRIPT AVANZADO

Clase 08

EDUCACIÓN IT

¿Qué veremos hoy?

- **Almacenamiento local**
  - ❑SessionStorage
  - ❑LocalStorage
  - ❑Otros tipos de almacenamiento local
- **JSON**
- Prácticas

Almacenamiento local

Junto al nacimiento de HTML5, surgió la necesidad de poder almacenar datos del lado del usuario, escapando a la "*prehistórica*" e insegura solución de las Cookies.

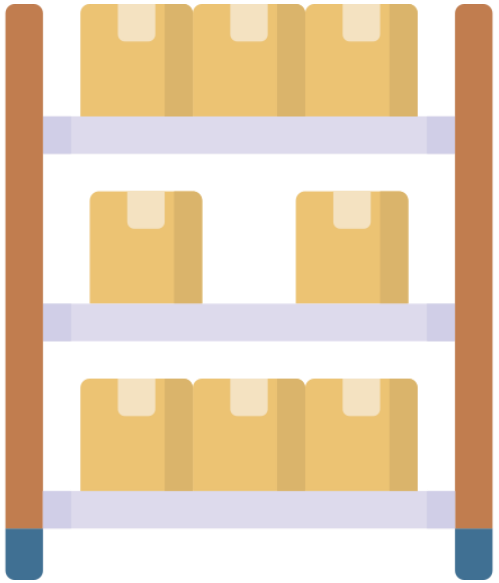
Así, la API de HTML5 comenzó a incluir diferentes propuestas para almacenar datos del lado del usuario con el mero fin de facilitar el acceso del navegador a determinada información, sin tener que hacer peticiones remotas y esperar a que los datos vuelvan.

Tipos de almacenamiento local

Así comenzaron a llegar diferentes propuestas. Y las tres que evolucionaron junto al navegador web, fueron:

- `localStorage/sessionStorage`
- `Web SQL`
- `Indexed DB`

Espacio de almacenamiento local



Cada uno de los tipos de almacenamiento mencionados, cuenta con diferentes características y capacidad de almacenamiento.

Veamos a continuación, un detalle de los mismos.

LocalStorage/SessionStorage

LocalStorage: almacena información del lado del usuario utilizando un mecanismo de pares `clave: valor`. Es extremadamente fácil de usar, y posee una serie de métodos que agregan, leen, modifican y eliminan la información almacenada.

SessionStorage: funciona exactamente igual que localStorage excepto que, al cerrar la pestaña del navegador, los datos se pierden. Como su nombre lo indica, quedan atados a la sesión de la pestaña.

LocalStorage/SessionStorage (almacenamiento)

Cada uno de ellos puede almacenar hasta 10 MB de información en el navegador web del usuario.

En este caso, la información almacenada tiene un límite fijo y, para poder almacenar nuevos datos, habrá que eliminar previamente, parte de los datos existentes.

Indexed DB

IndexedDB es una base de datos que funciona similar a las denominadas no-sql. Utiliza un mecanismo de almacenamiento, similar a la estructura JSON, y podemos crear todas las bases que consideremos necesarias para ser utilizadas con un mismo sitio web.

😕 como contra, tiene una curva de aprendizaje algo alta y es compleja de entender, al menos para quienes no dominan sistemas de almacenamiento similares a MongoDB.

😉 pero, existen opciones para implementarla que nos facilitan esta tarea. Una de ellas es la librería DexieJS, y es posible usarla para simplificar su curva de aprendizaje.

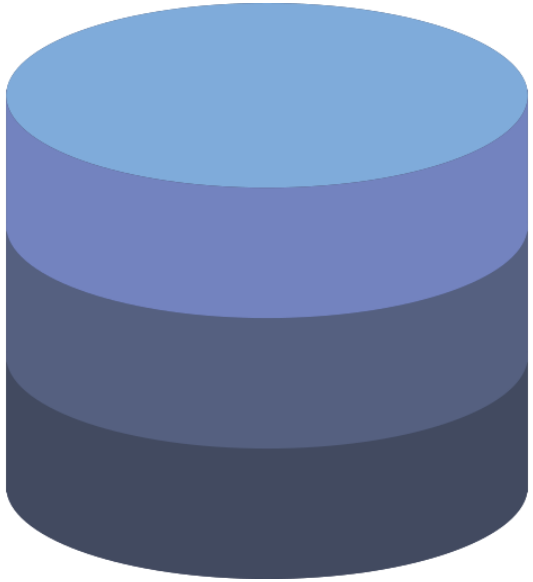
Indexed DB - almacenamiento

IndexedDB es el mecanismo de almacenamiento que más capacidad soporta almacenar, del lado del navegador web.

Este sistema puede guardar información por hasta el 80% de la capacidad de almacenamiento del S.O. (*sistema operativo*).

Es el más elegido por aplicaciones web profesionales, que tienen muy bien diseñada su arquitectura, para sacar el mejor provecho de este mecanismo: (*Youtube, Netflix, Spotify, Google Music, OneDrive, Google Drive*)

Web SQL - almacenamiento

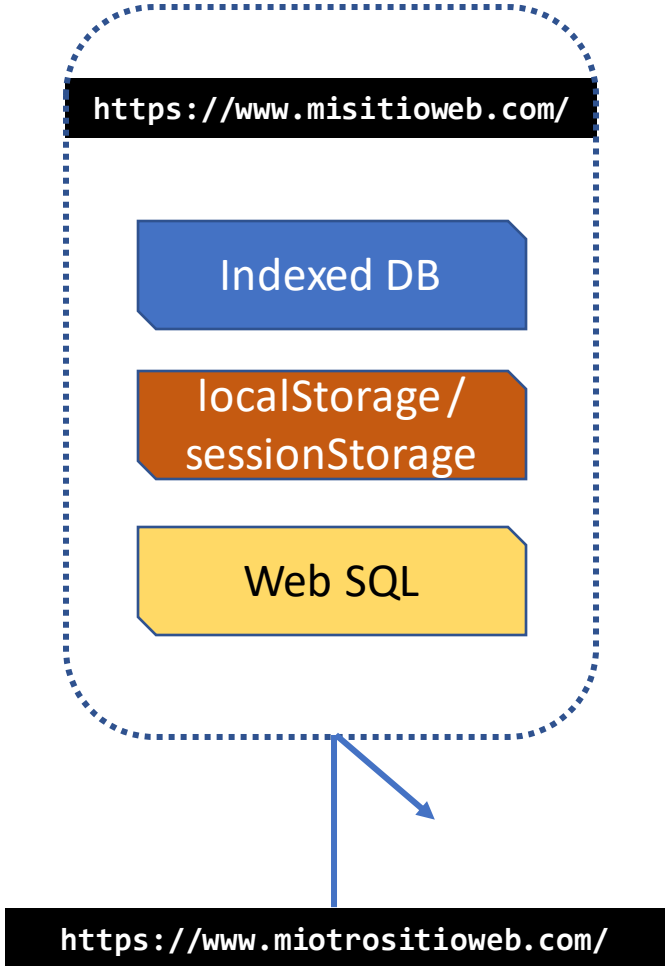


Web SQL, tal como su nombre lo indica, pone a disposición una base de datos con tablas, índices, y demás características, propias del mundo SQL.

💾 Soporta almacenamiento de hasta 20 MB con posibilidad de escalar/incrementar, según la necesidad, y con el permiso del usuario.

👎 Como desventaja, su API lleva + de 10 años sin actualizarse (*ni siquiera los bugs*), por lo tanto está desaconsejado su uso.
Safari ya desactivó esta característica de manera predeterminada.

Storage Scope



En cualquiera de los tres sistemas de almacenamiento local, el alcance o Scope de la información generada, se da 1 a 1 (*entre el sitio web que la generó y el sistema de almacenamiento*).

Si desde otro sitio web quiero leer datos almacenados por el primer sitio web, esto no será posible, obviamente que por cuestiones de seguridad.

Almacenamiento local

PRÁCTICAS

- Ejemplo Twitter (indexedDB)
- Pruebas LocalStorage en DevTools

LocalStorage/SessionStorage

Es un objeto contenido dentro del objeto global **`window`**.

Posee los siguientes métodos:

- `.setItem(item, valor)`
- `.getItem(item)`
- `.removeItem(item)`
- `.clear()`

LocalStorage/SessionStorage

Un escenario ideal para implementar local/sessionStorage, es cuando necesitamos resguardar la carga de un formulario complejo.

Para esto, básicamente tendríamos que hacer lo siguiente:

```
const nombre = document.querySelector('#nombre')

      //almacenar el valor del campo
      localStorage.setItem("nombre", nombre.value)
```

LocalStorage/SessionStorage

Si, luego, queremos recuperar el valor del input type almacenado en `localStorage`, para poder aplicarlo nuevamente a dicho `input type`, debemos realizar lo siguiente:

```
const nombre = document.querySelector('#nombre')
...
      //recuperar el valor del campo
      nombre.value = localStorage.getItem("nombre")
```

LocalStorage/SessionStorage

Cuando prescindimos de la información de dicho campo,

podemos eliminarla utilizando el siguiente método:

```
const nombre = document.querySelector('#nombre')
...
      //Eliminar una clave almacenada
      localStorage.removeItem("nombre")
```

LocalStorage/SessionStorage

Y si tenemos varios pares `clave:valor` almacenados y ya no los necesitamos más, utilizamos el siguiente método:

```
//Eliminar todas las claves almacenadas
localStorage.clear()
```

Almacenamiento local

PRÁCTICAS

- Almacenar / Recuperar datos de formulario

Storage y JSON

JSON


```
JSON.stringify(objetoJSON)
```

Convierte cualquier `objeto json` al formato *string*. De esta forma podrá ser almacenado, por ejemplo, en localStorage o en una bb.dd. remota.

```
JSON.parse(stringJSON)
```

Convierte cualquier formato *string* en un `objeto json`. De esta forma, podemos volver a interactuar con el objeto JSON dentro de JavaScript, aprovechando sus métodos.

JSON

PRÁCTICAS

- Almacenar / Recuperar datos de formulario en formato JSON
- Interactuar JSON con localStorage

EDUCACIÓN IT

¡GRACIAS!