Web APIs (localStorage, sessionStorage, IndexedDB)

Let's explore three important **Web APIs for client-side storage** in JavaScript: `localStorage`, `sessionStorage`, and `IndexedDB`. These APIs help store data in the browser without needing a backend.

🗃️ 1. localStorage

📌 Description:

- Stores data **with no expiration**.
- Data persists **even after the browser is closed and reopened**.
- Stored as **key-value pairs** (strings only).

✅ Use Case:

Remembering user settings, themes, login tokens, or preferences.

🧪 API Usage:

```
// Set data
localStorage.setItem("username", "Abhi");

// Get data
const name = localStorage.getItem("username"); // "Abhi"

// Remove item
localStorage.removeItem("username");

// Clear all
localStorage.clear();
```

> Note: Only strings can be stored. Use JSON.stringify() for objects.

```
const user = { name: "Abhi", age: 22 };
localStorage.setItem("user", JSON.stringify(user));

const retrieved = JSON.parse(localStorage.getItem("user"));
console.log(retrieved.name); // Abhi
```

🕒 2. sessionStorage

📌 Description:

- Stores data for the **duration of the page session**.
- Data is **cleared when the tab or window is closed**.
- Similar API to `localStorage`.

✅ Use Case:

Temporary data like a shopping cart, form input persistence, or navigation state.

🧪 API Usage:

```
// Set data
sessionStorage.setItem("step", "2");

// Get data
console.log(sessionStorage.getItem("step")); // "2"

// Remove data
sessionStorage.removeItem("step");

// Clear all
sessionStorage.clear();
```

> Same rules for string-only storage and JSON.stringify() apply here.

🧬 3. IndexedDB

📌 Description:

- A **low-level, asynchronous NoSQL database** for the browser.
- Stores **structured data**, including files/blobs.
- Much more powerful and flexible than `localStorage`.

✅ Use Case:

Offline apps, storing large datasets, caching API responses, or file storage.

🧪 Basic API Usage:

IndexedDB is **event-based** and more complex:

```
const request = indexedDB.open("MyDatabase", 1);

request.onupgradeneeded = function (e) {
  const db = e.target.result;
  db.createObjectStore("users", { keyPath: "id" });
};

request.onsuccess = function (e) {
  const db = e.target.result;

  const tx = db.transaction("users", "readwrite");
  const store = tx.objectStore("users");

  store.add({ id: 1, name: "Abhi" });

  tx.oncomplete = () => db.close();
};
```

🧩 Key Features:

- Asynchronous and non-blocking.
- Supports indexes and queries.
- Data can be objects, arrays, blobs, etc.
- Can store much more data than local/sessionStorage.

> Libraries like Dexie.js make IndexedDB easier to work with.

📊 Comparison Table

Feature	localStorage	sessionStorage	IndexedDB
Storage Limit	~5–10 MB	~5–10 MB	100MB+
Persistence	Forever	Until tab closes	Forever
Data Format	String only	String only	Objects, files, blobs, etc.
Access	Synchronous	Synchronous	Asynchronous (event-based)
Indexed Search	❌	❌	✅
Complex Data	❌ (needs JSON)	❌ (needs JSON)	✅

🛡️ Security Note:

- **Never store sensitive data** (like passwords, access tokens) in `localStorage` or `sessionStorage`.
- IndexedDB is sandboxed, but also avoid storing unencrypted sensitive data.

Would you like a sample project showing how to use all three in action (e.g., a notes app with persistent data)?