Web Storage Cheatsheet

Web Storage provides ways to store data in the browser: `localStorage`, `sessionStorage`, and `IndexedDB`.

- 1. localStorage
- 2. sessionStorage
- 3. IndexedDB

---

1. localStorage

Persistent key-value storage in the browser. Data survives page reloads and browser restarts.

API Overview

- `localStorage.setItem(key, value)` - Store a string value.
- `localStorage.getItem(key)` - Retrieve a value.
- `localStorage.removeItem(key)` - Remove a key.
- `localStorage.clear()` - Remove all keys.
- `localStorage.length` - Number of stored items.
- `localStorage.key(index)` - Get key by index.

Examples

```
// Store data
localStorage.setItem("username", "alice");

// Retrieve data
const user = localStorage.getItem("username"); // "alice"

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

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

```
12
13 // Iterate all keys
14 for (let i = 0; i < localStorage.length; i++) {
15   const key = localStorage.key(i);
16   console.log(key, localStorage.getItem(key));
17 }
```

2. sessionStorage

Similar to `localStorage`, but data is cleared when the page session ends (tab/window closed).

API Overview

- Same methods as `localStorage`, but scoped to the current tab/window.

Examples

```
// Store data
sessionStorage.setItem("token", "abc123");

// Retrieve data
const token = sessionStorage.getItem("token"); // "abc123"

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

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

3. IndexedDB

A low-level, asynchronous, NoSQL database for storing large amounts of structured data.

API Overview

- Use `indexedDB.open(name, version)` to open/create a database.
- Data is stored in object stores (like tables).
- All operations are asynchronous and use events.

Examples

```
// Open (or create) a database
const request = indexedDB.open("myDB", 1);

request.onupgradeneeded = function (event) {
  const db = event.target.result;
  // Create an object store
  db.createObjectStore("users", { keyPath: "id" });
};

request.onsuccess = function (event) {
  const db = event.target.result;
  // Start a transaction
  const tx = db.transaction("users", "readwrite");
  const store = tx.objectStore("users");
  // Add data
  store.add({ id: 1, name: "Alice" });
  // Get data
  const getReq = store.get(1);
  getReq.onsuccess = function () {
    console.log(getReq.result); // { id: 1, name: "Alice" }
  };
  tx.oncomplete = () ⇒ db.close();
};

```

```
request.onerror = function (event) {
  console.error("IndexedDB error:", event.target.error);
};
```

Notes

- IndexedDB is asynchronous and event-based.
- Use libraries like idb for a simpler promise-based API.