sessionStorage - Database and Caching

▼ What is `sessionStorage`?

Storage to store persistent data lasting a browser session.

MDN: The read-only `sessionStorage` property accesses a session `Storage` object for the current origin. `sessionStorage` is similar to `localStorage`; the difference is that while data in `localStorage` doesn't expire, data in `sessionStorage` is cleared when the *page session* ends.

▼ How to use `sessionStorage`?

methods available to use sessionStorage are:

```
// Setting data
sessionStorage.setItem('key', 'value');

// Getting data
const value = sessionStorage.getItem('key');

// Removing data
sessionStorage.removeItem('key');

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

▼ Size Limit

The storage limit for `sessionStorage` is approximately 5MB per origin (domain) in most browsers.

▼ Performance Issues

1. `sessionStorage` operations are synchronous. Hence, it can cause performance issues if these operations are used frequently or while working with large data

2. Speed of sessionStorage operations is fast for smaller data retrieval and storing but can soon degrade while working with large data or frequent use of these operations
3. Since data is stored as strings, any complex data using arrays or objects requires serialisation (eg: `JSON.stringify()`) and deserialisation methods (eg: `JSON.parse()`)

▼ Data Persistence

Data in `sessionStorage` persists only for the duration of the page session. This means it is available as long as the browser tab or window is open and will be cleared when the tab or window is closed or reloaded.

Important points to remember while using `sessionStorage` are as follows from **MDN**:

- Whenever a document is loaded in a particular tab in the browser, a unique page session gets created and assigned to that particular tab. That page session is valid only for that particular tab.
- A page session lasts as long as the tab or the browser is open, and survives over page reloads and restores.
- **Opening a page in a new tab or window creates a new session with the value of the top-level browsing context, which differs from how session cookies work.**
- Opening multiple tabs/windows with the same URL creates `sessionStorage` for each tab/window.
- Duplicating a tab copies the tab's `sessionStorage` into the new tab.
- Closing a tab/window ends the session and clears objects in `sessionStorage`

▼ Data Structure

`sessionStorage` stores data as key-value pairs where both the key and the value are strings.

If complex data structure has to be stored, one has to serialise the data and deserialise it when needed.

▼ Security

1. Since sessionStorage is accessible via JS code running on the same origin, it is vulnerable to XSS attacks
2. Since the data is stored in pain-text, no encryption is present to protect the information and can be viewed by anyone with access to the device.
3. We have to be careful about the storage limit and session expiry for usage of `sessionStorage`
4. We must sanitise the data added to `sessionStorage`
5. Avoid adding any sensitive data in `sessionStorage`

▼ **When to use?**

1. Storing temporary data specific to a single session (e.g., form inputs, temporary user state)
2. Data that does not need to persist beyond the page session.

▼ **When not to use?**

1. Storing data that needs to persist across browser sessions and tab closures (use `localStorage` instead).
2. Storing sensitive information (e.g., authentication tokens, personal information).
3. When data needs to be securely stored or encrypted.
4. Large amounts of data.