localStorage - Database and Caching

What is `localStorage`?

It is a storage to store data on a user's device persistently. The data will not be cleaned up when reloaded, tab closing etc.

MDN Definition: The `localStorage` read-only property of the `window` interface allows you to access a `Storage` object for the `Document`'s origin; the stored data is saved across browser sessions.

How to use `localStorage`?

The methods available to use localStorage are as follows:

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

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

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

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

Size Limit:

Approximately 5MB per domain. This lmit may differ based on the browser and the verisons

Performance Issues

1. **LocalStorage is synchronous:** Since localStorage is synchronous, it can block the main thread if used very frequently or with large amount of data
2. **Speed:** LocalStorage access is fast hopwever performance can be affected with frequency of usage and size of data
3. **Memory:** localStorage values are stored in the form of strings. Hence, complex data in the form of objects have to be serialised with the help of `JSON.stringify()` method and deserialised using `JSON.parse()` method which can affect the performance

▼ Data Persistence

Data stored in `localStorage` persists indefinitely until explicitly removed by the user or through JavaScript commands `localStorage` `.removeItem()` or `localStorage` `.clear()`

▼ Data Structure

Data is stored in key-value pairs and both key and value are strings. Hence for storing complex data in arrays or objects, we need to use serialisation methods such as `JSON.stringify()` and deserialisation methods like `JSON.parse()` to get the object form from the string value

▼ Security

Security concerns:

1. localStorage is accessible to JavaScript code running on the same origin and hence is vulnerable to XSS (Cross Site Scripting) Attacks.
2. Since this data is stored in plaintext, this data is easily visible making sensitive information stored in localStorage insecure

Precautions:

1. Do not store sensitive data, auth tokens etc in localStorage.
2. Encrypt data whenever necessary
3.

▼ When to use?

1. Uer preferences (e.g., theme settings, language choices).

2. Small amounts of data. Simple data.
3. Data that does not need to be kept secret or highly secure.
4. Data that needs to persist across browser sessions and tab closures.

▼ **When not to use?**

1. Sensitive information (e.g., authentication tokens, personal information).
2. When data needs to be securely stored or encrypted.
3. Large amounts of data which can impact performance.
4. Data requiring frequent changing and updating

The code example for localStorage can be checked here