**Local storage**

**Local storage** is the most recent mechanism. It allows for larger amounts of data to be stored, but the data is not deleted when the browser is closed. Local storage is useful for storing data that the user will need to access later, such as offline data.

Local Storage is a type of web storage that allows JavaScript to store and access data right in the browser. This is especially useful for storing data that you want to persist even if the user closes the browser, such as preferences or settings.

The data in Local Storage is stored in key/value pairs. The key is like the name of the data, and the value is like the actual data itself. You can think of it as a variable in JavaScript. To store data in Local Storage, you first need to create a key. Then you can store any data you want under that key.

Local storage in JavaScript allows you to store key-value pairs in a web browser. This storage persists even after the browser is closed, which makes it useful for storing small amounts of data locally.

Here's how you can use local storage in JavaScript:

1. Setting a value in local storage:

javascript

Copy code

localStorage.setItem('key', 'value');

2. Getting a value from local storage:

javascript

Copy code

const value = localStorage.getItem('key');

3. Removing a value from local storage:

javascript

Copy code

localStorage.removeItem('key');

4. Clearing all values from local storage:

javascript

Copy code

localStorage.clear();

Example:

javascript

Copy code

// Set a value

localStorage.setItem('username', 'john\_doe');

// Get a value

const username = localStorage.getItem('username');

console.log(username); // Output: john\_doe

// Remove a value

localStorage.removeItem('username');

// Clear all values

localStorage.clear();

Limitations:

Local storage is limited to string key-value pairs.

The data stored in local storage is accessible by any script from the same domain.

The total storage size is limited (usually around 5 MB per origin).

**Session Storage**

[**Session Storage**](https://www.tutorialspoint.com/get-data-from-sessionstorage-in-javascript) is a type of web storage that allows web applications to store data locally within the user's browser. Unlike cookies, data stored in session storage is specific to the site on which it was created and data is not shared with other

sites.

Session Storage is a new feature introduced in [**HTML5**](https://www.tutorialspoint.com/html5/index.htm) that allows you to store data locally in the user's browser. Unlike cookies, data stored in session storage is specific to the site on which it was created and data is not shared with other sites.

Session Storage is a way of storing data on the client side of an application. It's similar to local storage, but with a few key differences −

* Session Storage data is only available to the site that created it.
* Session Storage data is not shared with other sites.
* Session Storage data is not persistent, meaning it is only available for the duration of the user's session on a site.
* Session Storage data is specific to the browser tab in which it was created.

Session Storage is a great way to improve the performance of your web applications by reducing the amount of data that needs to be transferred between the client and server. It can also be used to store data in a more secure way since the data is not stored in cookies where it can be accessed by third-party sites.

Session storage in JavaScript is similar to local storage, but it has a different scope and lifetime. While local storage persists even after the browser is closed, session storage is available only for the duration of the page session. When the browser tab or window is closed, the session storage data is cleared.

Here's how you can use session storage in JavaScript:

1. Setting a value in session storage:

javascript

Copy code

sessionStorage.setItem('key', 'value');

2. Getting a value from session storage:

javascript

Copy code

const value = sessionStorage.getItem('key');

3. Removing a value from session storage:

javascript

Copy code

sessionStorage.removeItem('key');

4. Clearing all values from session storage:

javascript

Copy code

sessionStorage.clear();

Example:

javascript

Copy code

// Set a value

sessionStorage.setItem('username', 'john\_doe');

// Get a value

const username = sessionStorage.getItem('username');

console.log(username); // Output: john\_doe

// Remove a value

sessionStorage.removeItem('username');

// Clear all values

sessionStorage.clear();

Limitations:

Session storage is limited to string key-value pairs.

Data stored in session storage is specific to the current browser tab or window and is not shared across tabs or windows.

Data stored in session storage is cleared when the tab or window is closed.

Best Practices:

Use session storage for temporary data that needs to be available for the duration of the page session.

Avoid storing sensitive information in session storage as it's accessible within the same browser tab or window.

Handle cases where session storage is not available gracefully by providing fallback mechanisms.

Session storage provides a way to store temporary data within the context of a browser session, and it's particularly useful for scenarios where you need to maintain state or data across multiple page loads within the same tab or window.

