The HTTP protocol is one of the most important protocols for smooth communication between the server and the client. The main disadvantage of the HTTP protocol is that it is a stateless protocol, which means it does not track any response or request by the server or the client. So to resolve this problem, there are three ways to track useful information. In this, we will see the difference between local storage, and session storage.

**Local Storage:**

This read-only interface property provides access to the document’s local storage object; the stored data is stored across browser sessions. Similar to session Storage, except that session Storage data gets cleared when the page session ends—that is, when the page is closed. It is cleared when a browser's last “private” tab is closed (local Storage data for a document loaded in a private browsing or incognito session).

This behavior cannot be guaranteed because, as mentioned above, the file URL requirements remain unclear. As such, there’s a possibility that browsers may change how they handle files at any time. The way some browsers handle it has evolved.

Local Storage provides a simple API for storing and retrieving data in the client's browser. The data saved using local storage is saved across browser sessions, meaning it persists even when the browser is closed and reopened.

**Local Storage Methods:**

**1. setItem()**

This method is used to store data in local Storage. It takes two arguments: a key and a value. The key is a string that is used to reference the value, and the value is the data you want to store. Although only strings can be directly stored, you can store objects or arrays by converting them to strings using JSON.stringify().

localStorage.setItem('username', 'JohnDoe');

localStorage.setItem('userData', JSON.stringify({ age: 30, gender: 'male' }));

**2. getItem()**

This method retrieves data from local Storage by its key. If the key does not exist, null is returned. For data stored as JSON strings, you can convert them back into JavaScript objects using JSON.parse().

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

const userData = JSON.parse(localStorage.getItem('userData'));

console.log(username); // Output: JohnDoe

console.log(userData); // Output: { age: 30, gender: 'male' }

**3. removeItem()**

This method removes the specified item from local Storage. You just need to provide the key of the data you want to remove.

localStorage.removeItem('username');

**4. clear()**

This method clears all entries in local storage for the current domain. It is useful for clearing all data when it is no longer needed, like when logging out a user.

localStorage.clear(); // Removes all items in localStorage

**Session Storage:**

Session Storage objects can be accessed using the session Storage read-only property. The difference between session Storage and local Storage is that local storage data does not expire, whereas session storage data is cleared when the page session ends.

A unique page session gets created once a document is loaded in a browser tab. Page sessions are valid for only one tab at a time. Pages are only saved for the amount of time that the tab or the browser is open; they do not persist after the page reloads and restores. A new session is created each time a tab or window is opened; this is different from session cookies. Each tab/window that is opened with the same URL creates its session storage. When you duplicate a tab, the session Storage from the original tab is copied to the duplicated tab. Closing a window/tab ends the session and clears session Storage objects.

**Session Storage Methods:**

**1. setItem()**

This method allows you to store data in session Storage. It requires two parameters: key (the name you will use to reference the data) and value (the data you want to store). Session Storage can only directly store strings, but you can store objects by converting them to strings using JSON.stringify().

sessionStorage.setItem('sessionKey', '12345');

sessionStorage.setItem('userInfo', JSON.stringify({ username: 'JohnDoe', permissions: 'admin' }));

**2. getItem()**

This method retrieves data from session Storage using the specified key. If the key does not exist, it returns null. To use complex data types that were stored as strings, convert them back using JSON.parse().

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

const userInfo = JSON.parse(sessionStorage.getItem('userInfo'));

console.log(sessionKey); // Output: 12345

console.log(userInfo); // Output: { username: 'JohnDoe', permissions: 'admin' }

**3. removeItem()**

Use this method to delete a specific item from session Storage by specifying the key associated with the data you want to remove.

sessionStorage.removeItem('sessionKey');

**4. clear()**

This method is used to clear all data stored in session Storage for the current session. It’s useful for cleaning up all session-specific data at once, such as when a user logs out or completes a transaction.

sessionStorage.clear(); // Clears all items stored in sessionStorage

**Difference Between Local Storage and Session Storage**

| **Local Storage** | **Session Storage** |
| --- | --- |
| The storage capacity of local storage is 5MB/10MB | The storage capacity of session storage is 5MB |
| As it is not session-based, it must be deleted via javascript or manually | It’s session-based and works per window or tab. This means that data is stored only for the duration of a session, i.e., until the browser (or tab) is closed |
| The client  can only read local storage | The client can only read local storage |
| There is no transfer of data to the server | There is no transfer of data to the server |
| Supported by all browsers, including older ones. | Supported by all browsers, including older ones |