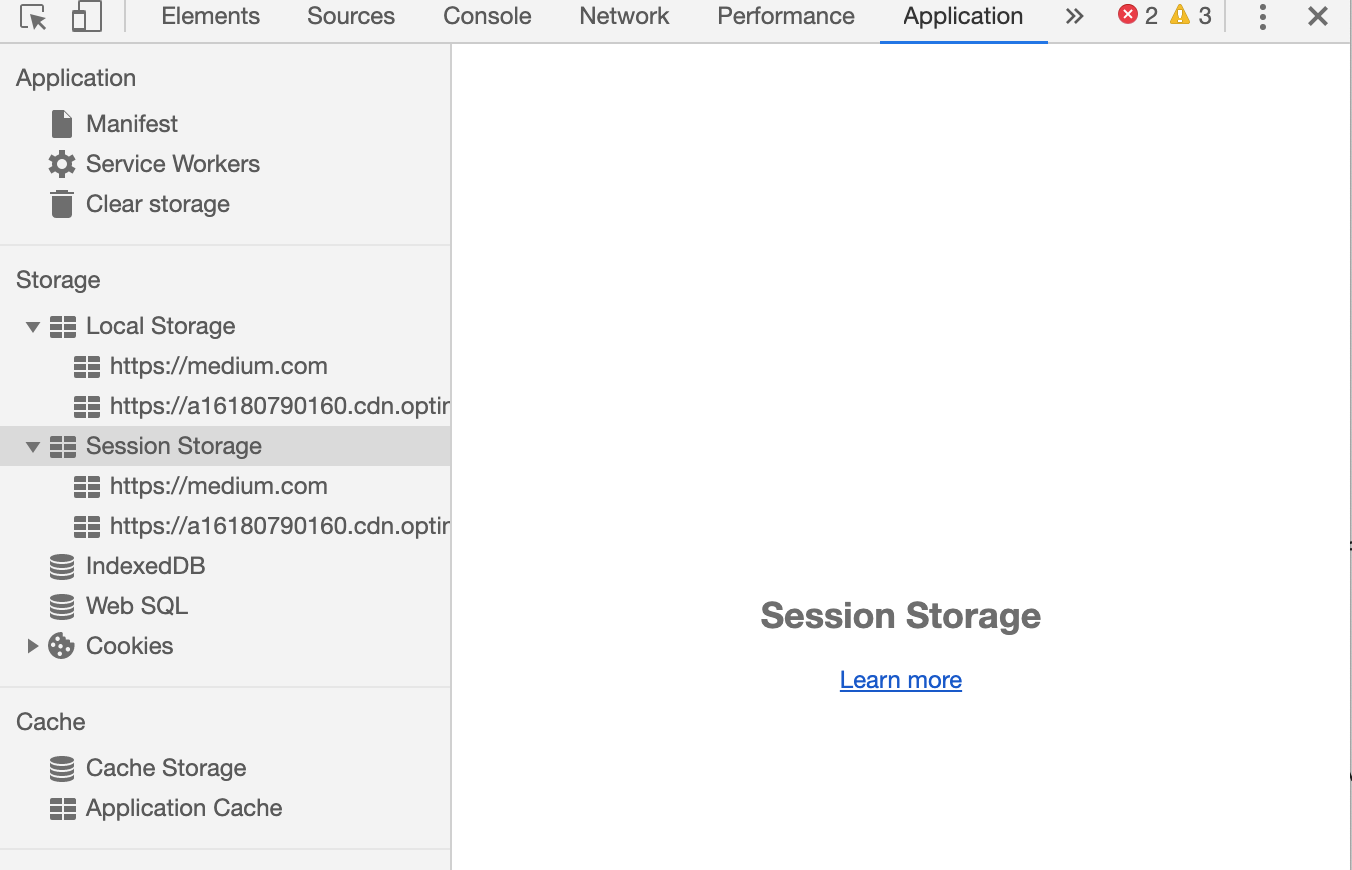
**Local Storage vs Session Storage vs Cookie**



Various Storage provided by browser now

With Advent of Html5 , we have got various option to cache or store info on client browser. Previously we were having only cookies , which were very restrictive and size of cookies was very small. but now we local storage and session storage as well. and cookies has been talk of past , though it is getting used for various purposes. let's talk about all these

**LocalStorage**

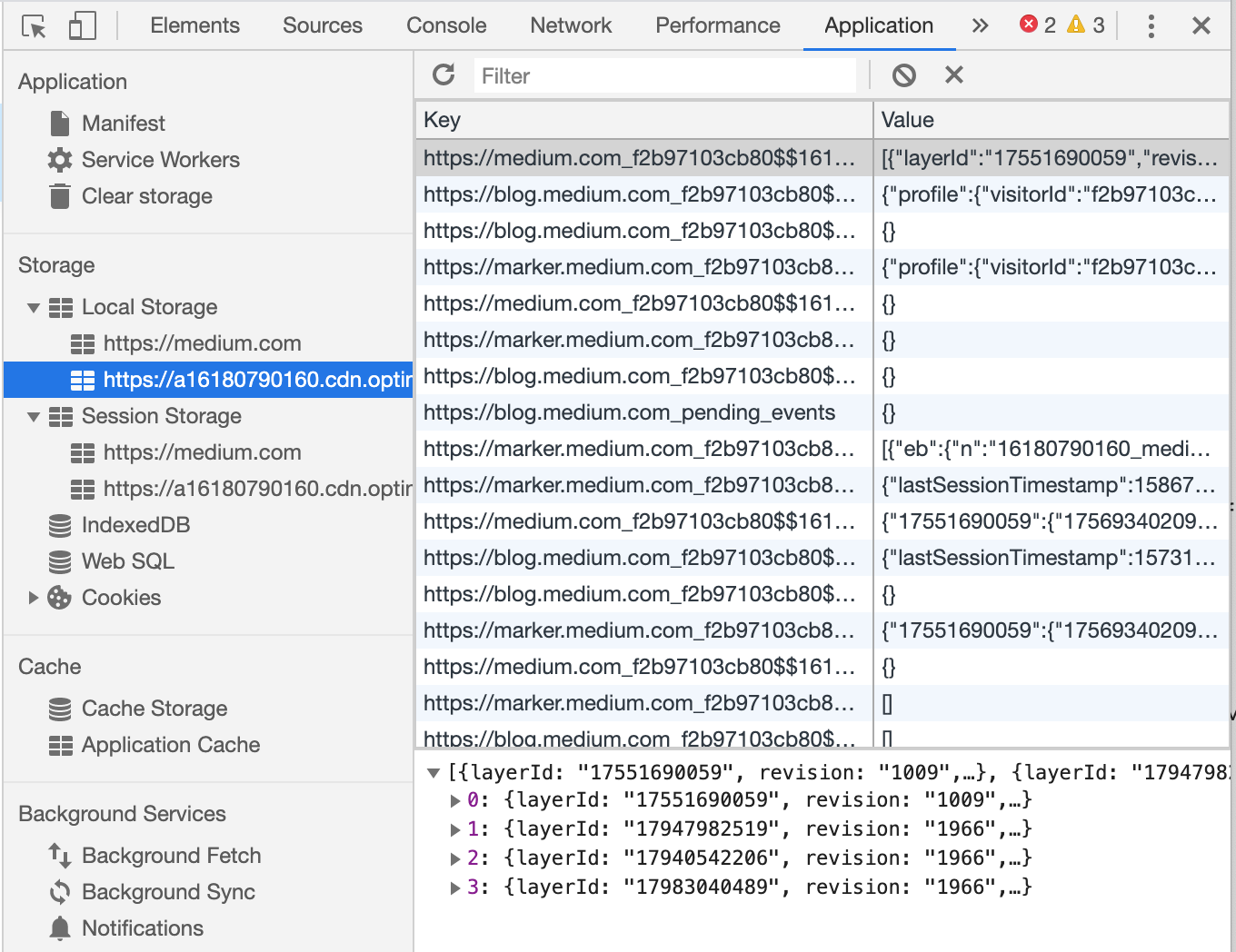
localStorage is a way to store data on the client’s computer. It allows the saving of key/value pairs in a web browser and it stores data with no expiration date. localStorage can only be accessed via JavaScript, and HTML5. However, the user has the ability to clear the browser data/cache to erase all localStorage data.

**Pros:**

* stores data with no expiration date
* storage limit is about 5MB
* data is never transferred to the server

**Cons:**

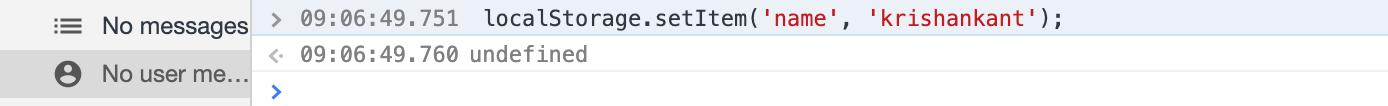
* plaintext, hence not secure by design
* limited to string data, hence need to be serialized
* can only be read on client-side



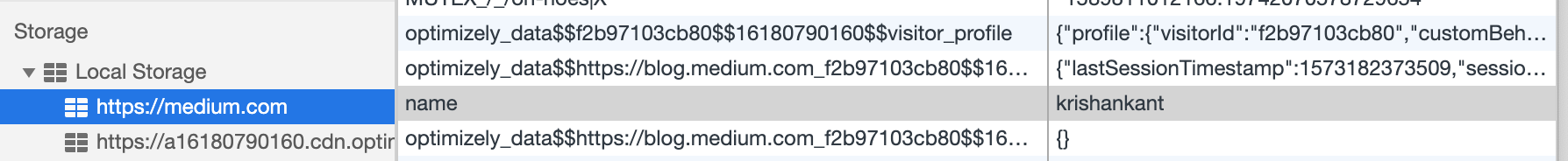
Local Storage

So we see , we have key value storage, where we have key and value as an serialized object.

We can try localstorage api from console

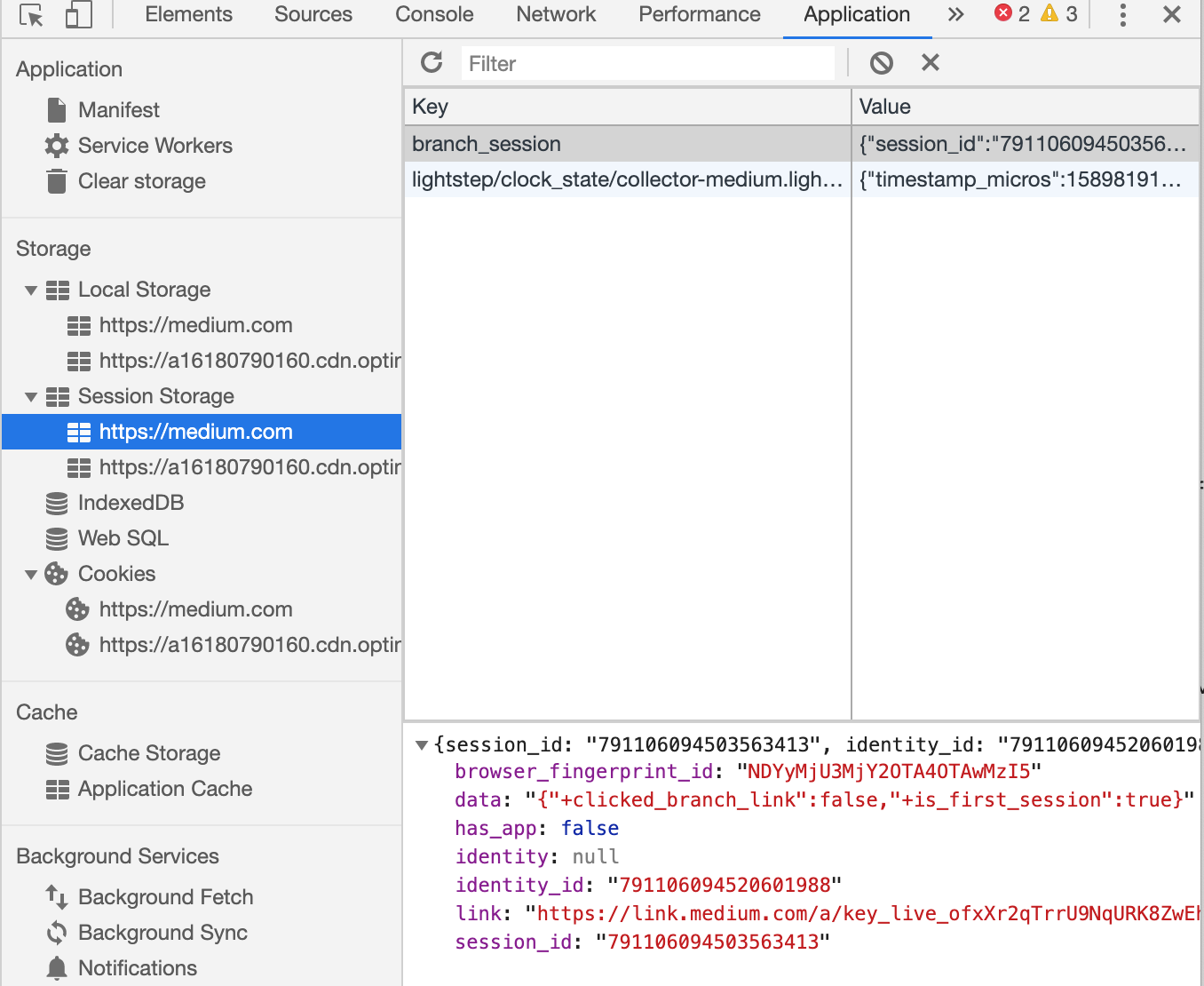


and it will create a key value pair in localstorage store.

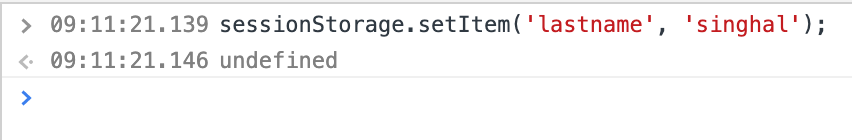


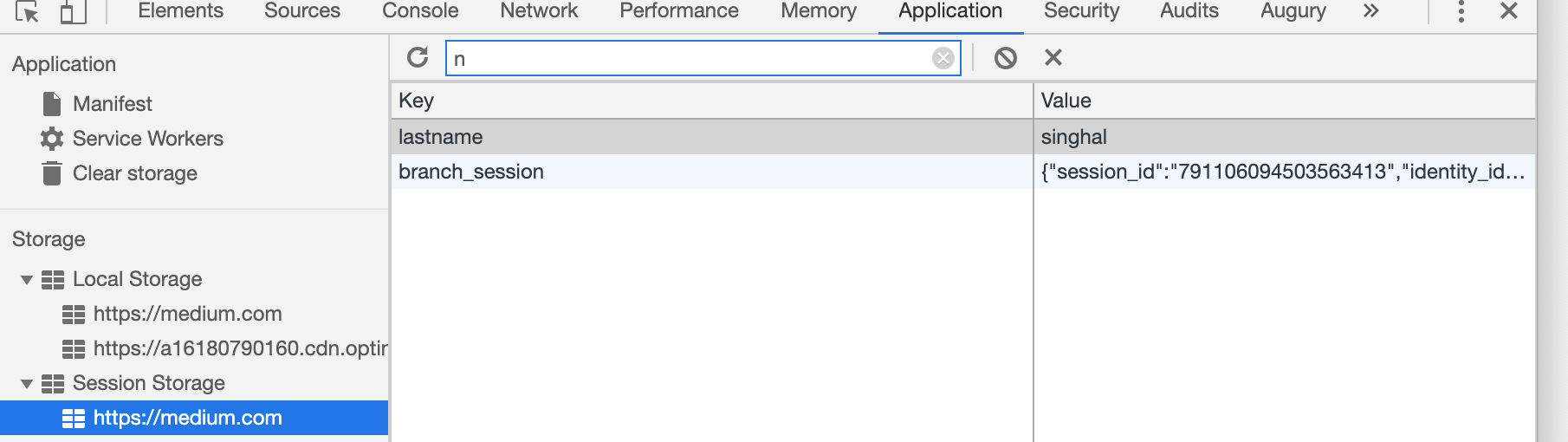
Session storage

* stores data only for a session, meaning that the data is stored until the browser (or tab) is closed
* data is never transferred to the server
* can only be read on client-side
* storage limit is about 5-10MB
* opening multiple tabs/windows with the same URL creates sessionStorage for each tab/window



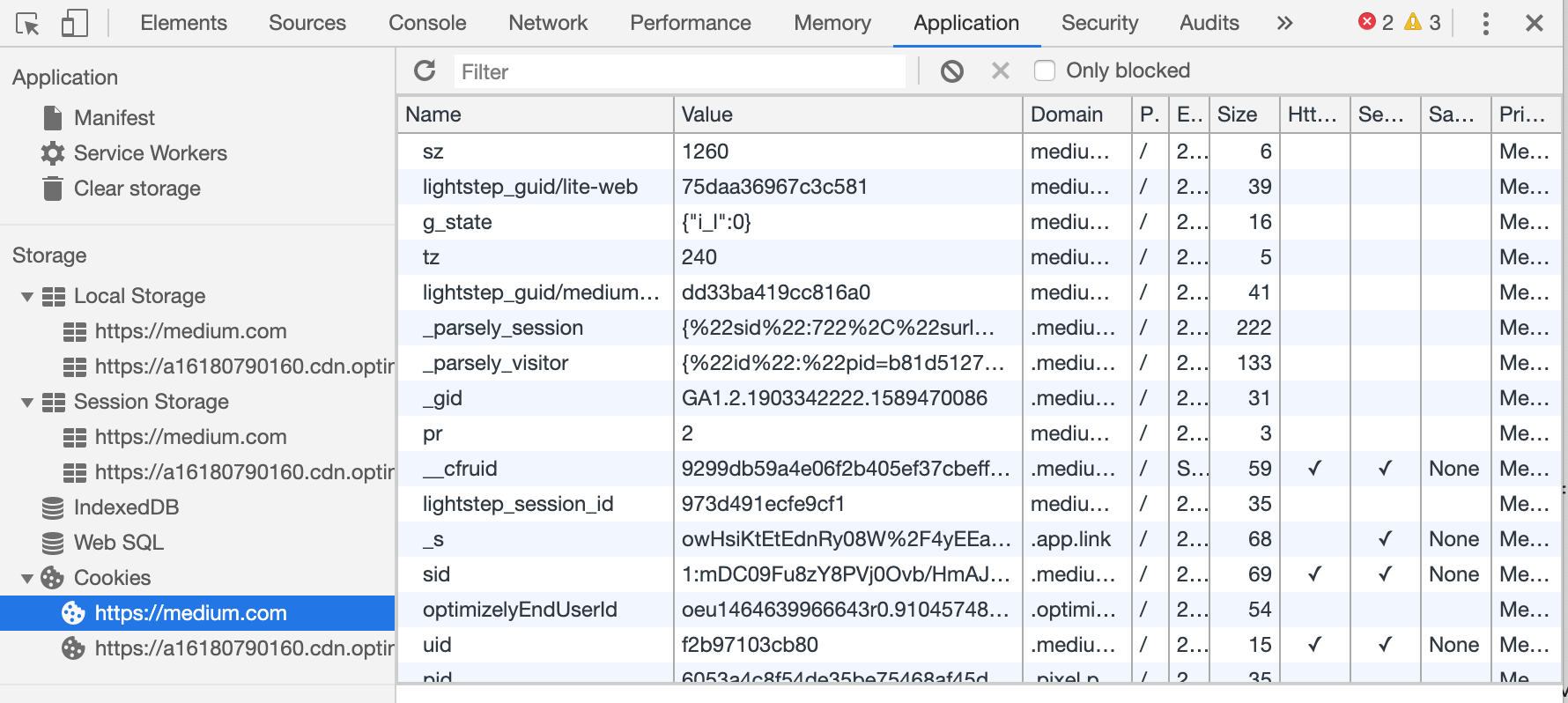
Same as localstorage , we can write key-value pair in session storage as well.



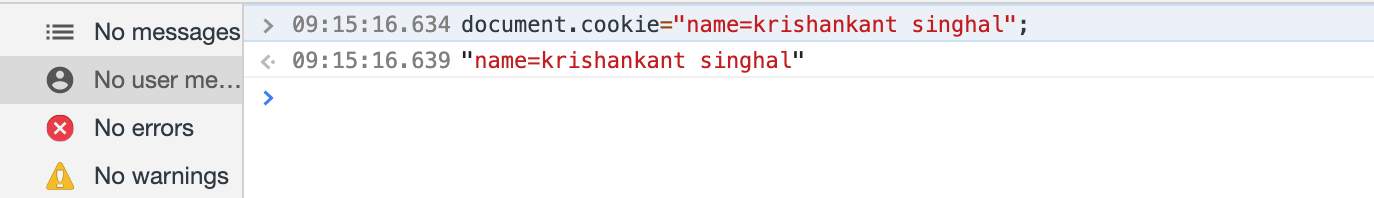


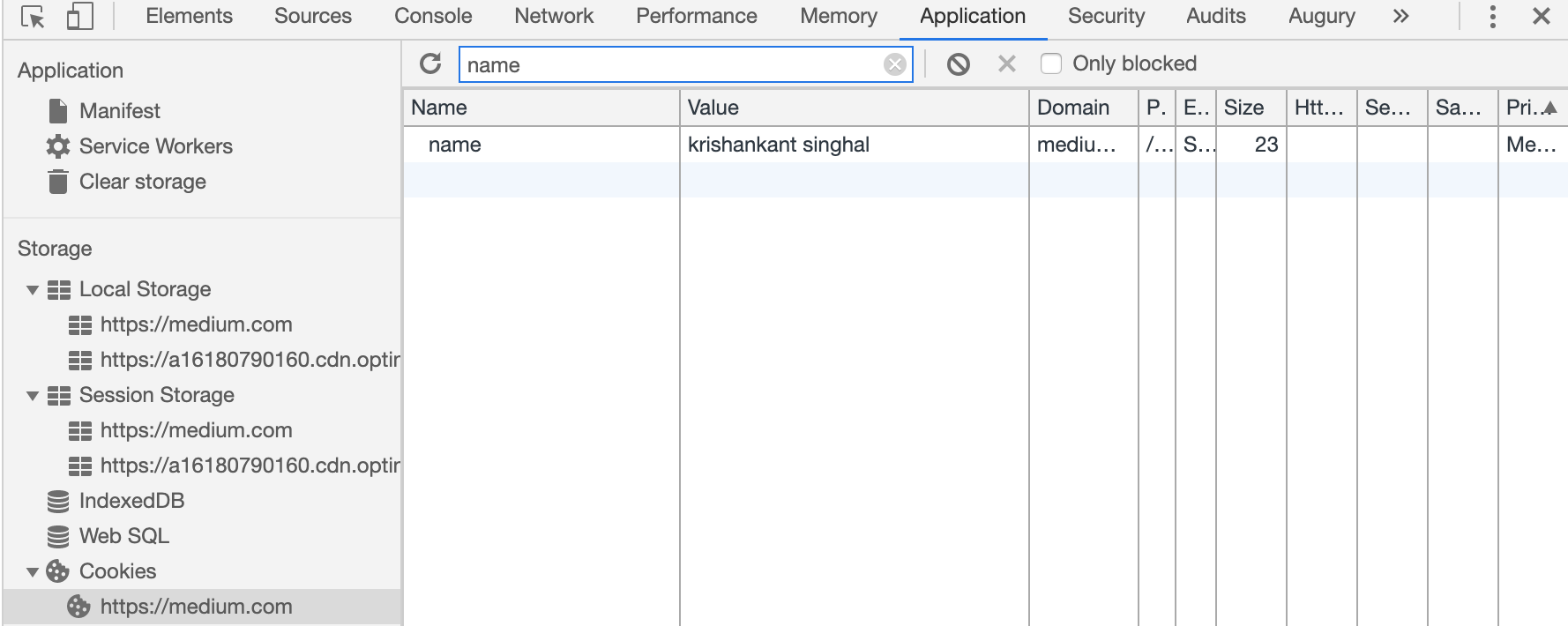
**Cookie**

* Stores data that has to be sent back to the server with subsequent XHR requests. Its expiration varies based on the type and the expiration duration can be set from either server-side or client-side .
* Cookies are primarily for server-side reading (can also be read on client-side), localStorage and sessionStorage can only be read on client-side.
* Size must be less than 4KB.
* Cookies can be made secure by setting the httpOnly flag as true for that cookie. This prevents client-side access to that cookie.



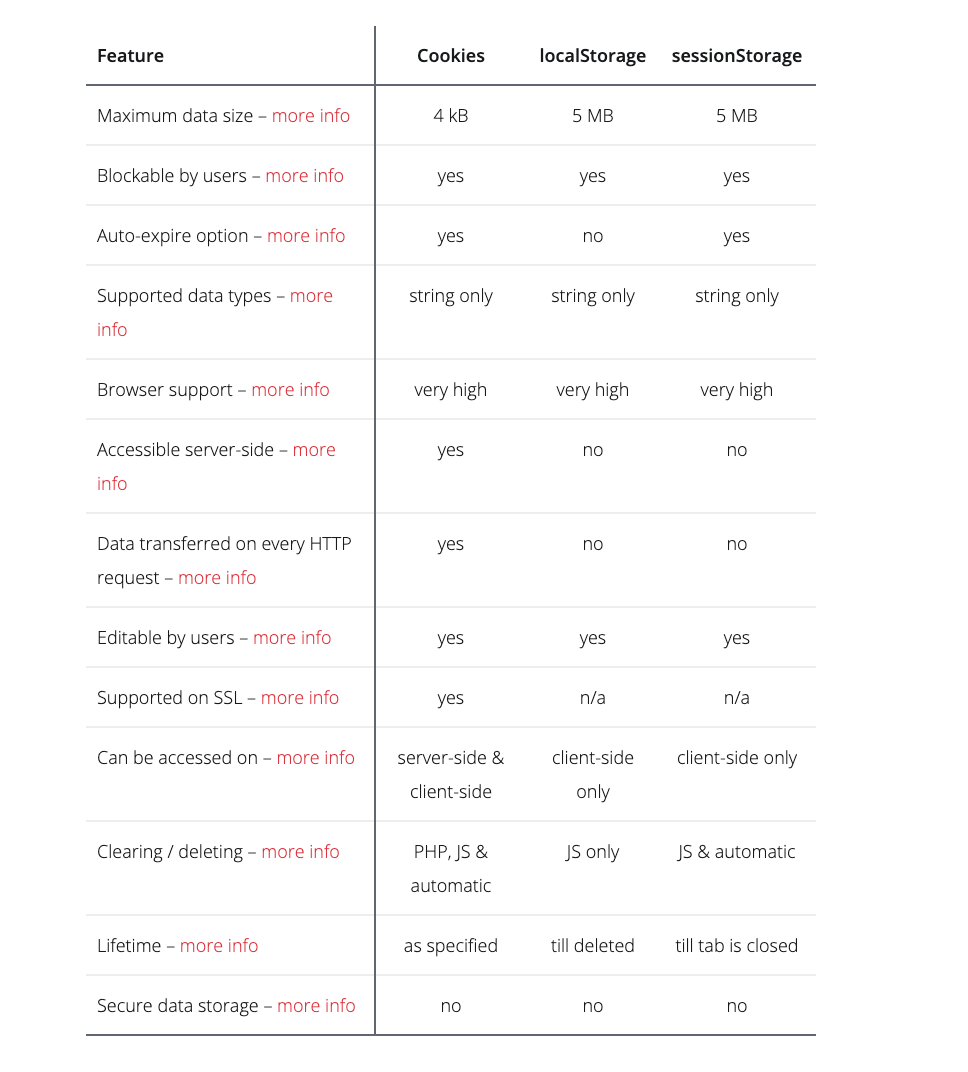
Cookies can be updated , set using document.cookie object from browser window object.





In chrome browser, by inspect element, we can go to application tab and see values which are in localStorage, Session Storage and Cookies.

So in nutshell



523

1

* [Localsto](https://medium.com/tag/localsto)
* [Cookies](https://medium.com/tag/cookies)
* [Sessionstorage](https://medium.com/tag/sessionstorage)
* [JavaScript](https://medium.com/tag/javascript)
* [Browser Storage](https://medium.com/tag/browser-storage)

[More from krishankant singhal](https://krishankantsinghal.medium.com/?source=follow_footer-------------------------------------)

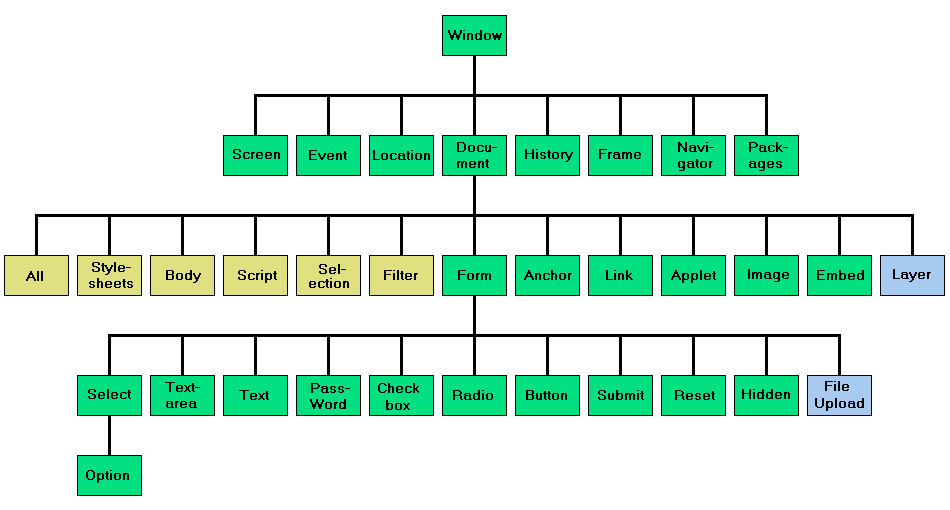
Follow

Angular,Vuejs,Android,Java,Git developer. i am nerd who want to learn new technologies, goes in depth.

[May 13, 2020](https://krishankantsinghal.medium.com/window-document-object-for-browser-ac15902be531?source=follow_footer---------0----------------------------)

[**window.document object for browser.**](https://krishankantsinghal.medium.com/window-document-object-for-browser-ac15902be531?source=follow_footer---------0----------------------------)

The window object is a global object which has various object to access various functionality of browser. below diagram gives an overview of window object.

[[](https://krishankantsinghal.medium.com/window-document-object-for-browser-ac15902be531?source=follow_footer---------0----------------------------)](https://krishankantsinghal.medium.com/window-document-object-for-browser-ac15902be531?source=follow_footer---------0----------------------------)

Here’s a bird’s-eye view of what we have when JavaScript runs in a web-browser:

[Read more · 7 min read](https://krishankantsinghal.medium.com/window-document-object-for-browser-ac15902be531?readmore=1&source=follow_footer---------0----------------------------)

9

[Apr 22, 2020](https://krishankantsinghal.medium.com/async-and-await-in-javascript-6253ccd6f41e?source=follow_footer---------1----------------------------)

[**Async and Await in javascript**](https://krishankantsinghal.medium.com/async-and-await-in-javascript-6253ccd6f41e?source=follow_footer---------1----------------------------)

Async and await are extension of promises in javascript. if you are not clear about promises, please go through my other tutorial.

**[Promises in javascript. Lets understand them](https://medium.com/@krishankantsinghal/promises-in-javascript-lets-understand-them-bdced7a7cf05" \t "_blank)**

[Promises in javascript is kind of promises in real life. you promise to someone that you will do something in future or…](https://medium.com/@krishankantsinghal/promises-in-javascript-lets-understand-them-bdced7a7cf05" \t "_blank)

[medium.com](https://medium.com/@krishankantsinghal/promises-in-javascript-lets-understand-them-bdced7a7cf05" \t "_blank)

async can be declared with function only and await can be used inside async function only.

From Mozilla website

**async function** declaration defines an **asynchronous function** — a function that is an [AsyncFunction](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/AsyncFunction) object. Asynchronous functions operate in a separate order than the rest of the code via the [event loop](https://developer.mozilla.org/en-US/docs/Web/JavaScript/EventLoop), returning an implicit [Promise](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise) as its result. *But the syntax and structure of your code using async functions is much more like using standard synchronous functions.*

So…

[Read more · 4 min read](https://krishankantsinghal.medium.com/async-and-await-in-javascript-6253ccd6f41e?readmore=1&source=follow_footer---------1----------------------------)

2

[Apr 18, 2020](https://krishankantsinghal.medium.com/promises-in-javascript-lets-understand-them-bdced7a7cf05?source=follow_footer---------2----------------------------)

[**Promises in javascript. Lets understand them**](https://krishankantsinghal.medium.com/promises-in-javascript-lets-understand-them-bdced7a7cf05?source=follow_footer---------2----------------------------)

Promises in javascript is kind of promises in real life. you promise to someone that you will do something in future or now. and there will be 2 outcome of that promise. Either promise will be fulfilled or promise will not be fulfilled.

Same behaviour is exhibited in javascript as well.

Let’s understand this using an conversation between mom and child .

1)Son Promises to his Mother , Mother i Promise i will clean my house.

So son is telling to do some work which will be done now or in future. but is not completed. …

[Read more · 2 min read](https://krishankantsinghal.medium.com/promises-in-javascript-lets-understand-them-bdced7a7cf05?readmore=1&source=follow_footer---------2----------------------------)

7

[Apr 13, 2020](https://krishankantsinghal.medium.com/serialization-and-deserialization-5046c958c317?source=follow_footer---------3----------------------------)

[**Serialization and Deserialization**](https://krishankantsinghal.medium.com/serialization-and-deserialization-5046c958c317?source=follow_footer---------3----------------------------)

As per Wikipedia, **serialization** (or **serialisation**) is the process of translating [data structures](https://en.wikipedia.org/wiki/Data_structure) or [object](https://en.wikipedia.org/wiki/Object_(computer_science)) state into a format that can be stored (for example, in a [file](https://en.wikipedia.org/wiki/Computer_file) or memory [buffer](https://en.wikipedia.org/wiki/Data_buffer)) or transmitted (for example, across a [network](https://en.wikipedia.org/wiki/Computer_network) connection link) and reconstructed later (possibly in a different computer environment).When the resulting series of bits is reread according to the serialization format, it can be used to create a semantically identical clone of the original object.

[[https://miro.medium.com/max/45/0*996yfGf7GcZHnYdH.png?q=20](https://krishankantsinghal.medium.com/serialization-and-deserialization-5046c958c317?source=follow_footer---------3----------------------------)](https://krishankantsinghal.medium.com/serialization-and-deserialization-5046c958c317?source=follow_footer---------3----------------------------)

so basically **Serialization** is a process of converting an object into a sequence of bytes which can be persisted to a disk or database or…

[Read more · 4 min read](https://krishankantsinghal.medium.com/serialization-and-deserialization-5046c958c317?readmore=1&source=follow_footer---------3----------------------------)

6

[Feb 17, 2020](https://krishankantsinghal.medium.com/javascript-understanding-destructuring-in-es6-c70ea842b8dc?source=follow_footer---------4----------------------------)

[**Javascript Understanding Destructuring in ES6**](https://krishankantsinghal.medium.com/javascript-understanding-destructuring-in-es6-c70ea842b8dc?source=follow_footer---------4----------------------------)

With the advent of ES6 in javascript we got lots of cool features. some of them are arrow function, map,filter, reducer and destructuring of object and Array etc.

When we talk About destructing in javascript. we have to talk about

1)Object Destructing

2) Array Destructuring

So Basically Destructuring is a new feature introduced in ES6 to unpack values from arrays or properties from an object. It helps in improving the readability and performance of our code.

Lets understand how it is to use properties or Array value before destructuring and after destructuring

**ES5 Implementation**

// Example 1 - Object Destructuring…

[Read more · 2 min read](https://krishankantsinghal.medium.com/javascript-understanding-destructuring-in-es6-c70ea842b8dc?readmore=1&source=follow_footer---------4----------------------------)