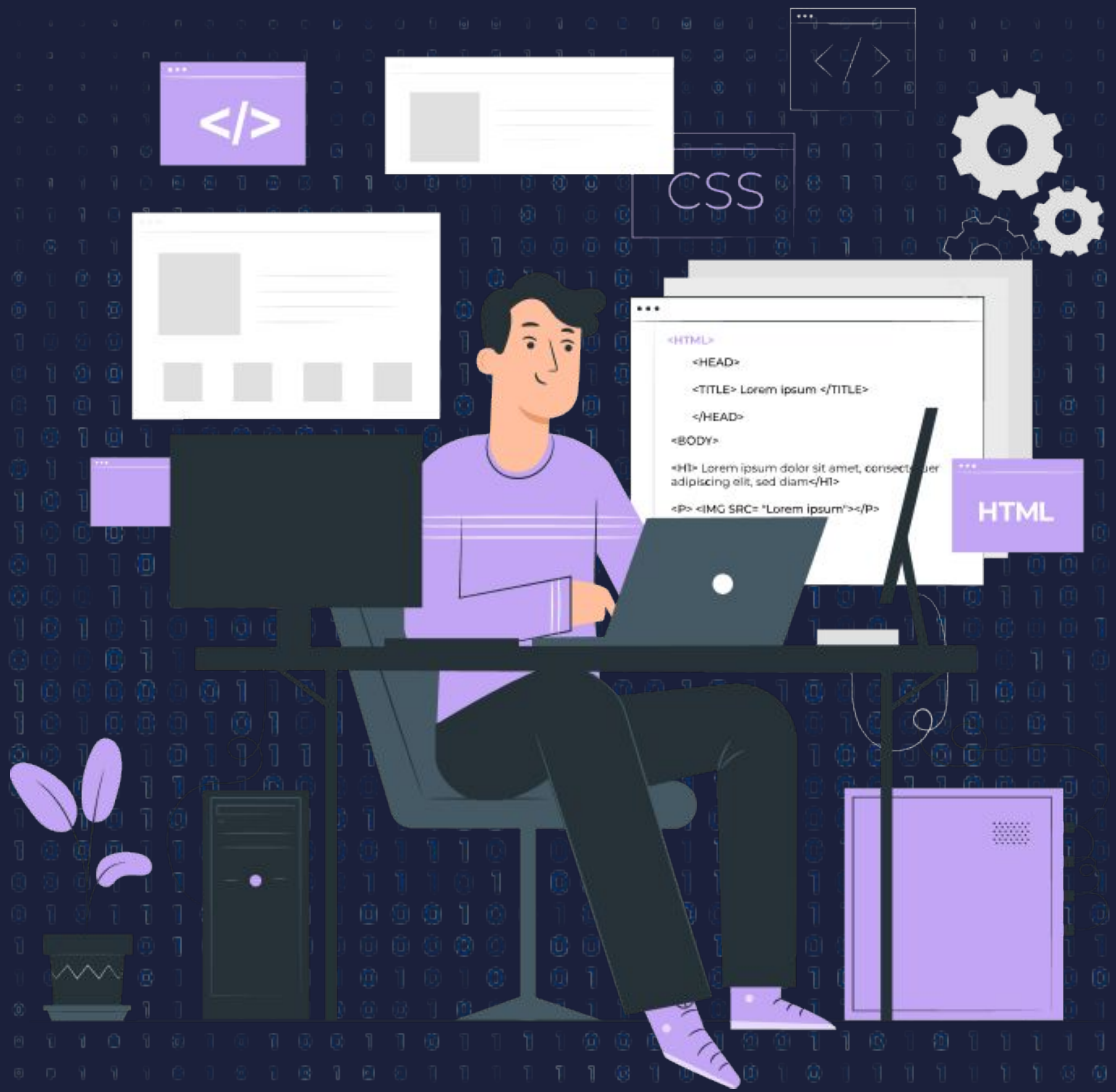
SKILLS

Introduction to window keyword



</>
CSS
HTML
<HTML>
<HEAD>
<TITLE> Lorem ipsum </TITLE>
</HEAD>
<BODY>
<H1> Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam</H1>
<P> <IMG SRC= "Lorem ipsum"></P>

Topics

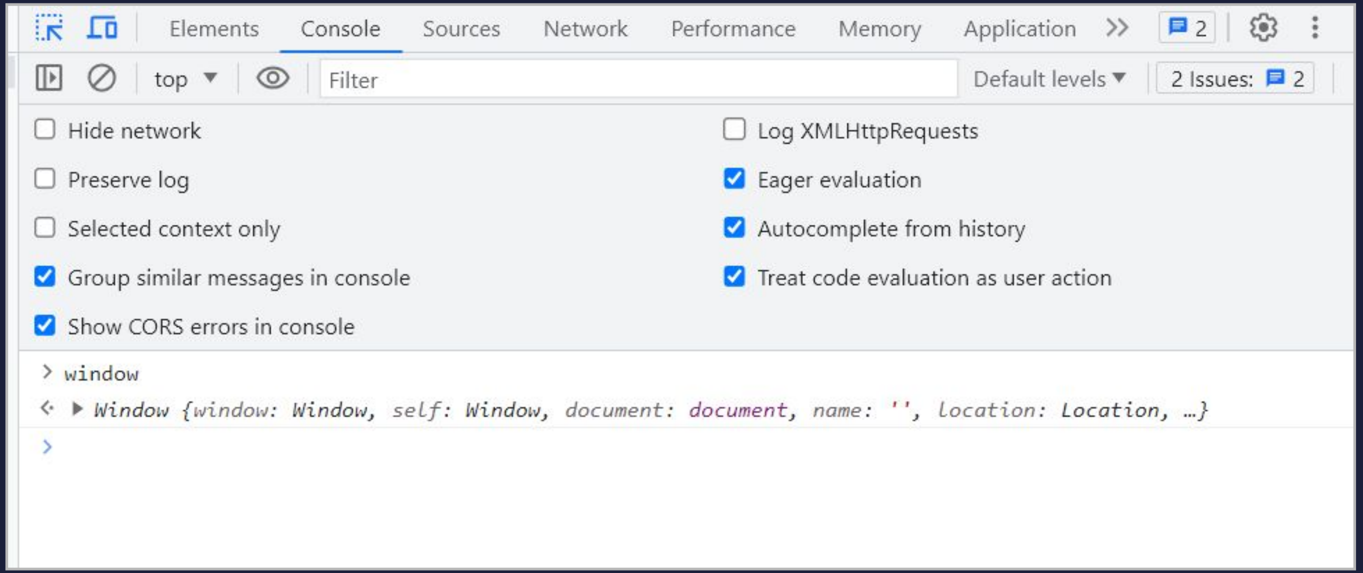
- Understanding Window keyword
- Window Properties
- Window method
- Window Events
- Cookies in JavaScript

DOM representation of HTML

In JavaScript, the window keyword represents the global object in a web browser environment. It is the top-level object that contains various properties and methods related to the browser window or tab.

Window keyword can be used to view all the global objects present in a web browser by going to developer mode of the browser and writing window in the console.

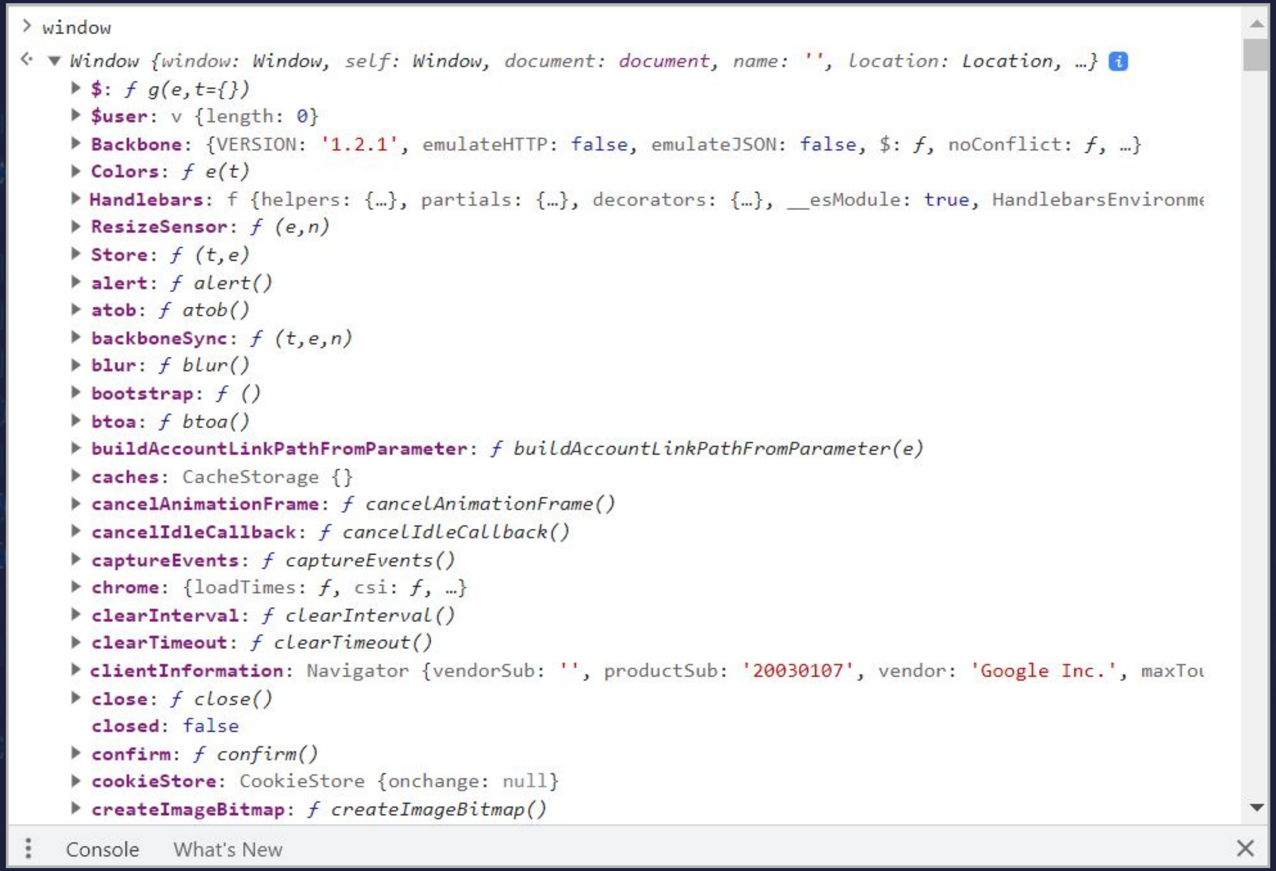
Browser developer mode



- [ ] Hide network
- [ ] Preserve log
- [ ] Selected context only
- [x] Group similar messages in console
- [x] Show CORS errors in console
- [ ] Log XMLHttpRequests
- [x] Eager evaluation
- [x] Autocomplete from history
- [x] Treat code evaluation as user action

```
> window
<· ▶ Window {window: Window, self: Window, document: document, name: '', location: Location, …}
>
```

Window object ->



```
> window
<· ▼ Window {window: Window, self: Window, document: document, name: '', location: Location, …}
    ▶ $: ƒ g(e,t={})
    ▶ $user: v {length: 0}
    ▶ Backbone: {VERSION: '1.2.1', emulateHTTP: false, emulateJSON: false, $: ƒ, noConflict: ƒ, …}
    ▶ Colors: ƒ e(t)
    ▶ Handlebars: f {helpers: {…}, partials: {…}, decorators: {…}, __esModule: true, HandlebarsEnvironme
    ▶ ResizeSensor: ƒ (e,n)
    ▶ Store: ƒ (t,e)
    ▶ alert: ƒ alert()
    ▶ atob: ƒ atob()
    ▶ backboneSync: ƒ (t,e,n)
    ▶ blur: ƒ blur()
    ▶ bootstrap: ƒ ()
    ▶ btoa: ƒ btoa()
    ▶ buildAccountLinkPathFromParameter: ƒ buildAccountLinkPathFromParameter(e)
    ▶ caches: CacheStorage {}
    ▶ cancelAnimationFrame: ƒ cancelAnimationFrame()
    ▶ cancelIdleCallback: ƒ cancelIdleCallback()
    ▶ captureEvents: ƒ captureEvents()
    ▶ chrome: {loadTimes: ƒ, csi: ƒ, …}
    ▶ clearInterval: ƒ clearInterval()
    ▶ clearTimeout: ƒ clearTimeout()
    ▶ clientInformation: Navigator {vendorSub: '', productSub: '20030107', vendor: 'Google Inc.', maxTou
    ▶ close: ƒ close()
      closed: false
    ▶ confirm: ƒ confirm()
    ▶ cookieStore: CookieStore {onchange: null}
    ▶ createImageBitmap: ƒ createImageBitmap()
```

Here are a few key points to understand about the window object:

- Global Scope
- Browser Environment Access
- Window Methods and Properties
- Global Object for Scripts

Window Properties

Some of the commonly used various properties related to the browser window keyword include -

- document - Represents the Document Object Model (DOM) of the current web page, allowing manipulation and interaction with the HTML structure and content.
- location - Provides information about the URL of the current web page, including properties like window.location.href (full URL), window.location.host (hostname and port), window.location.pathname (path of the URL), etc.
- navigator - Contains information about the browser and user's device, such as window.navigator.userAgent (user agent string), window.navigator.language (user's preferred language), etc.
- localStorage and sessionStorage - Provide access to the web storage APIs for storing data persistently (localStorage) or for the duration of a session (sessionStorage).
- history - Allows interaction with the browser's session history, enabling navigation back and forth through previously visited pages.
- console - Provides access to the browser console for logging messages and debugging.

Window method

Some of the commonly used various properties related to the browser window keyword include -

- alert - Displays an alert dialog box with the specified message.
- prompt - Displays a prompt dialog box with the given message, allowing the user to enter input. The optional default parameter sets a default value for the input
- confirm - Displays a confirmation dialog box with the specified message, presenting the user with the OK and Cancel buttons. Returns true if the user clicks OK, and false if the user clicks Cancel.
- open - Opens a new browser window or tab with the specified URL. The name, specs, and replace parameters are optional and allow
- close - Closes the current browser window or tab, if it was opened using JavaScript.
- scrollTo - Scrolls the document to the specified coordinates (x, y) within the window.
- scrollBy - Scrolls the document by the specified amount (x, y) relative to the current scroll position.

localStorage -

Data stored in localStorage persists even when the browser is closed and reopened. It remains available until explicitly cleared or removed by the user or the application.

```
// Storing data in localStorage
localStorage.setItem('name', 'John');
localStorage.setItem('email', 'john@gmail.com');

// Retrieving data from localStorage
const name = localStorage.getItem('name');
const email = localStorage.getItem('email');

console.log(name); // Output: John
console.log(email); // Output: john@gmail.com

// Removing a key-value pair from localStorage
localStorage.removeItem('email');

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

console.log(localStorage.length); // Output: 0
```

sessionStorage -

Data stored in sessionStorage is isolated within a single browsing session. It is available as long as the browser window or tab is open and gets cleared when the session ends

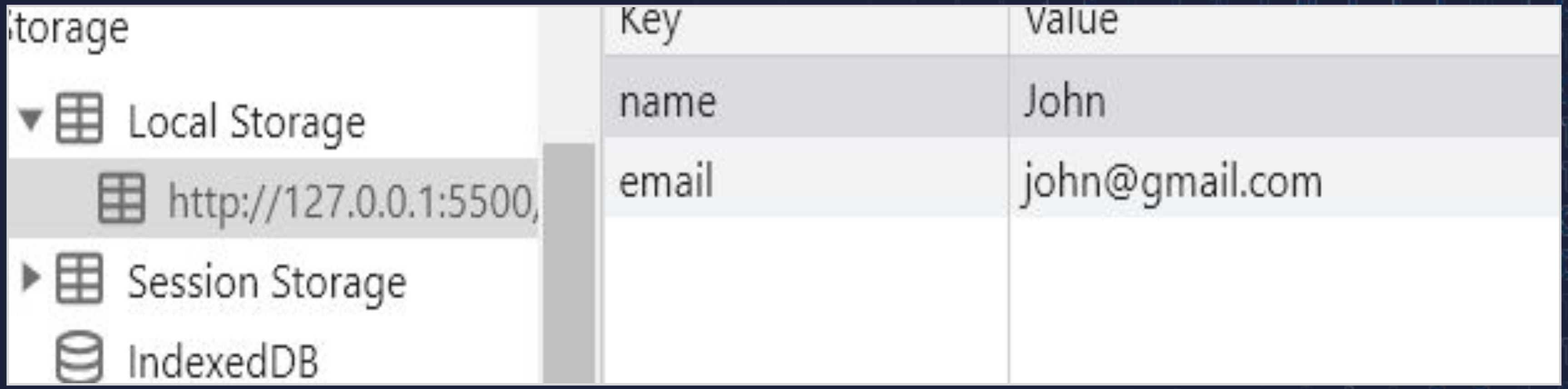
```
// Storing data in sessionStorage
sessionStorage.setItem('name', 'John');
sessionStorage.setItem('email', 'john@gmail.com');

// Retrieving data from sessionStorage
const name = sessionStorage.getItem('name');
const mail= sessionStorage.getItem('john@gmail.com');

console.log(name); // Output: John
console.log(mail); // Output: john@gmail.com

// Removing a key-value pair from sessionStorage
sessionStorage.removeItem('mail');

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

console.log(sessionStorage.length); // Output: 0
```

LocalStorage demonstration

```
index.html
<html lang="en">
 <body>
  <h1>LocalStorage demo</h1>
  <script src="./script.js"></script>
 </body>
</html>

script.js
// Storing data in localStorage
localStorage.setItem('name', 'John');
localStorage.setItem('email', 'john@gmail.com');
```

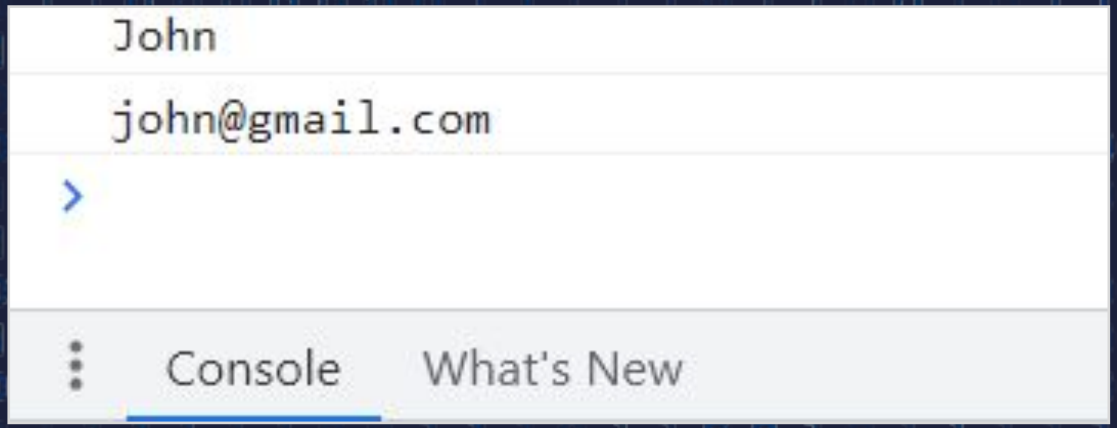
```
// Retrieving data from localStorage
const name = localStorage.getItem('name');
const email = localStorage.getItem('email');

console.log(name); // Output: John
console.log(email); // Output: john@gmail.com
```

Browser output –



Key	Value
name	John
email	john@gmail.com

Browser output –

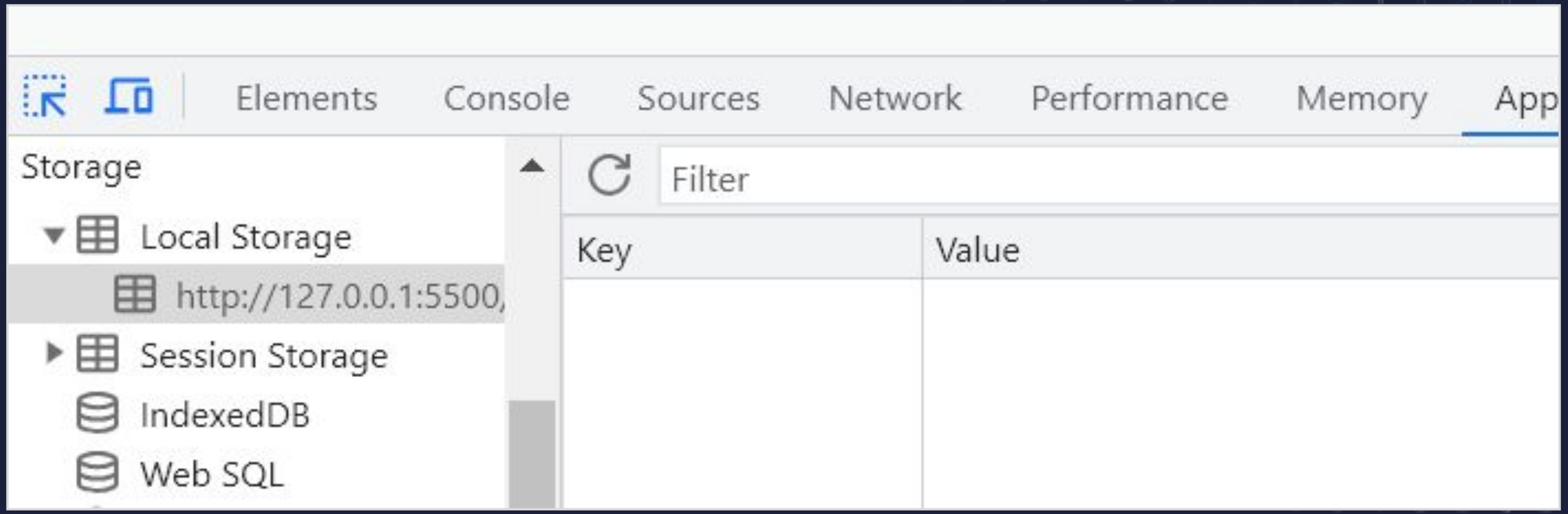


```
John
john@gmail.com
>
```

continue...

```
// Removing a key-value pair from localStorage
localStorage.removeItem('email');
// Clearing all data from localStorage
localStorage.clear();
console.log(localStorage.length); // Output: 0
```

Browser output -

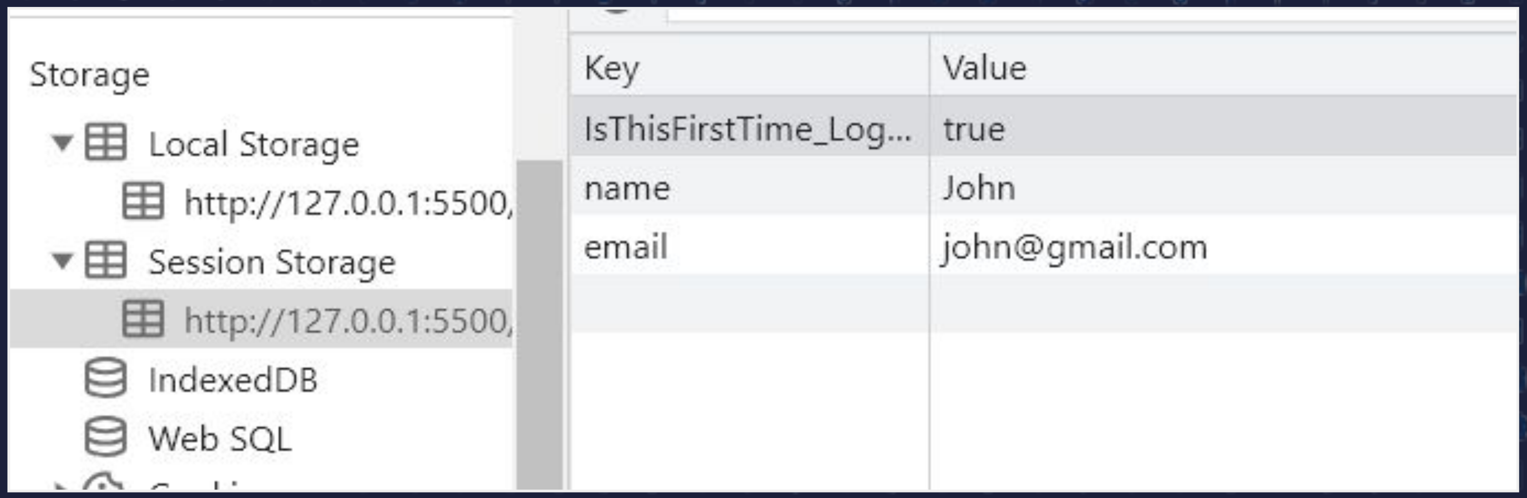


sessionStorage demonstration

```
index.html
<html lang="en">
    <body>
    <h1>sessionStorage demo</h1>
  </body>
  <script src="./script.js"></script>
</html>
```

```
 script.js
// Storing data in sessionStorage
sessionStorage.setItem('name', 'John');
sessionStorage.setItem('email', 'john@gmail.com');

// Retrieving data from sessionStorage
const name = sessionStorage.getItem('name');
const mail= sessionStorage.getItem('mail');

console.log(name); // Output: John
console.log(mail); // Output: john@gmail.com
```

Browser output -

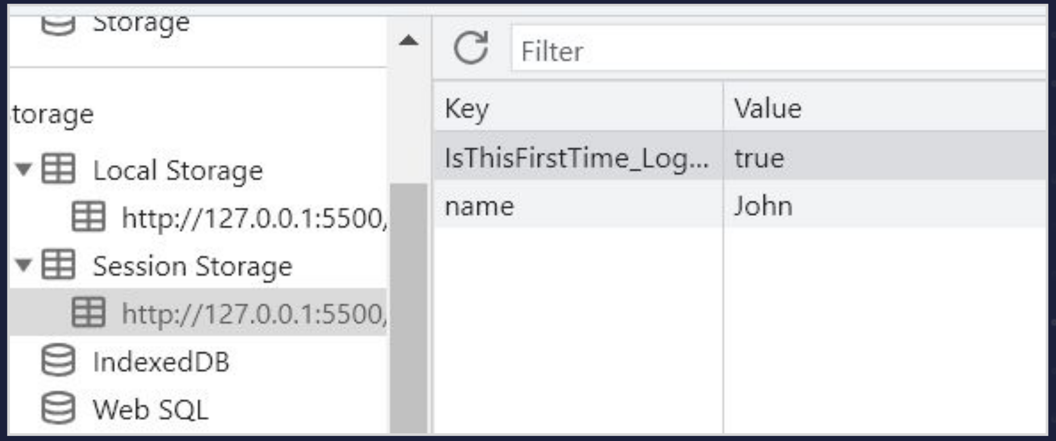


Key	Value
IsThisFirstTime_Log...	true
name	John
email	john@gmail.com

continue...

```
// Removing a key-value pair from sessionStorage
sessionStorage.removeItem(email);
```

Browser output -

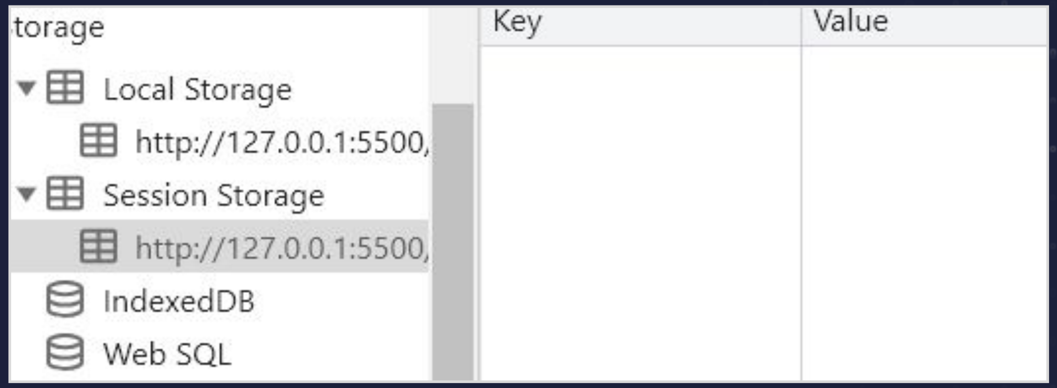

Storage
torage
Local Storage
http://127.0.0.1:5500,
Session Storage
http://127.0.0.1:5500,
IndexedDB
Web SQL

Filter

Key	Value
IsThisFirstTime_Log...	true
name	John

```
// Clearing all data from sessionStorage
sessionStorage.clear();
console.log(sessionStorage.length); // Output: 0
```

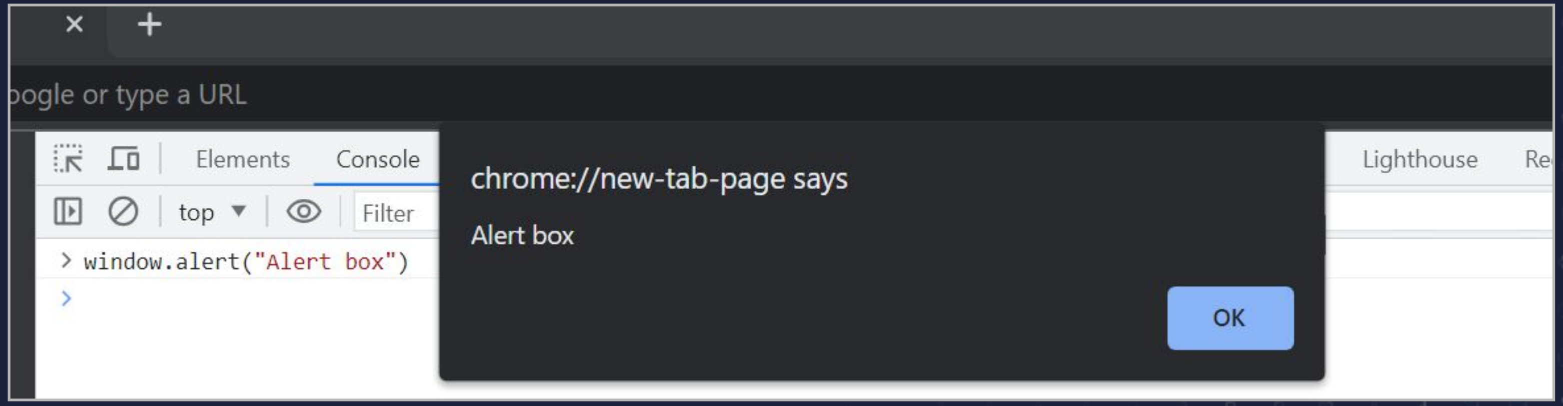
Browser output -


torage
Local Storage
http://127.0.0.1:5500,
Session Storage
http://127.0.0.1:5500,
IndexedDB
Web SQL

Key	Value

continue...

Alert - window.alert("Alert box")

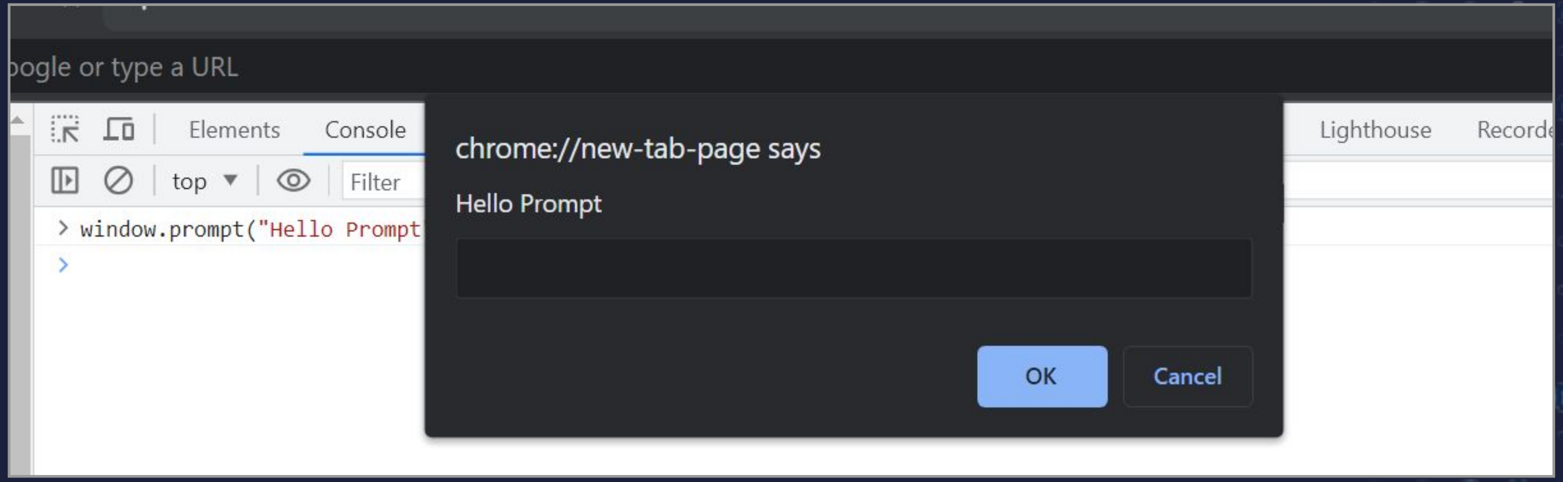


```
> window.alert("Alert box")
```

chrome://new-tab-page says

Alert box

prompt- window.prompt("Hello Prompt")

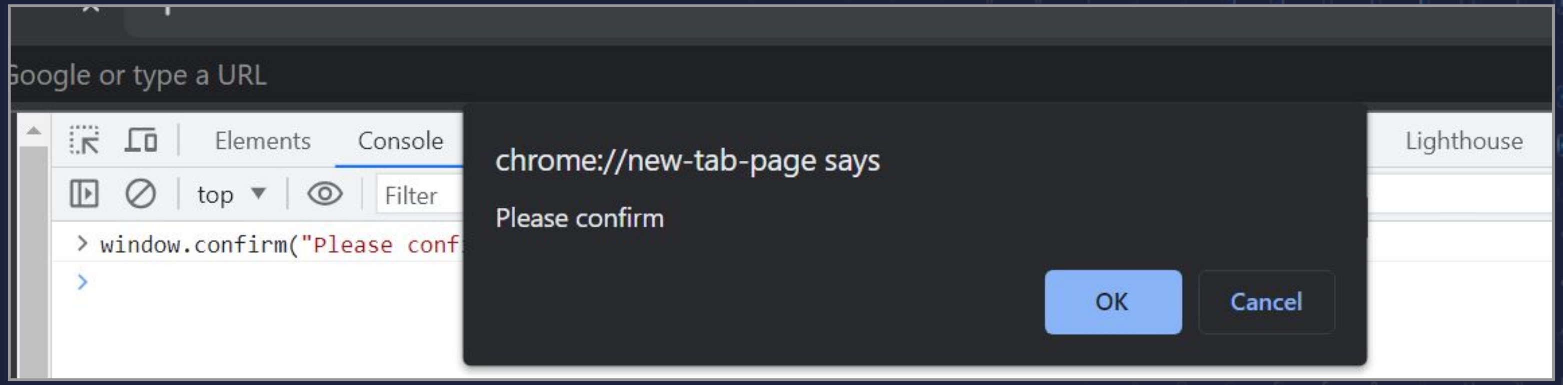


```
> window.prompt("Hello Prompt
```

chrome://new-tab-page says

Hello Prompt

prompt- window.confirm("Please confirm")



```
> window.confirm("Please conf
```

chrome://new-tab-page says

Please confirm

continue...

```
open -
 window.open('https://www.google.com', '_blank');
// output - open google page in new tab


close -
window.close();
// output - it will close the current open browser tab.


scrollTo -
window.scrollTo(0, 500);
/** output - it will scroll vertically starting from top i.e 0 pixel
to 500 pixels
**/
```

```
scrollBy -
window.scrollBy(0, 200);
/*
// Scroll the window down by 200 pixels vertically and 0 pixels
horizontally
*/
```

Window Event

Here are some commonly used window events

- load -
- resize - hen the window finishes loading all its resources, including images, stylesheets, and scripts.
- unload - Fired just before the window is about to be unloaded, such as when the user navigates away from the page or closes the window.
- beforeunload - Similar to unload, but fires just before the window is about to be unloaded, allowing you to prompt the user with a confirmation dialog to prevent accidental navigation away from the page.
- hashchange - Fired when the URL hash (fragment identifier) of the page changes.
- focus & blur - Fired when the window gains or loses focus, respectively.
- DOMContentLoaded - Fired when the initial HTML document has been completely loaded and parsed, without waiting for stylesheets, images, and subframes to finish loading.
- error - Fired when an error occurs during the loading of a resource, such as a script or an image.
- online - Fired when the browser establishes a connection to the internet.
- offline - Fired when the browser loses its internet connection

continue...

load -
```
window.addEventListener('load', function() {
  // code to execute after window has finish // laoding
});
```

resize -
```
window.addEventListener('resize', function() {
  // Code to execute when the window is resized
});
```

unload -
```
window.addEventListener('unload', function() {
  // Code to execute before the windown is unloaded
});
```

beforeunload -
```
window.addEventListener('beforeunload', function() {
  // Code to execute before the window is unloaded
  });
```

hashchange -
```
window.addEventListener('hashchange', function() {
  // Code ...
});
```

focus & blur -
```
window.addEventListener('focus ', function() {
  // Code ...});
window.addEventListener('blur', function() {
  // Code ...});
```

continue...

DOMContentLoaded -
```
window.addEventListener('DOMContentLoaded', function() {
  // Code ...});
```

error -
```
window.addEventListener('error', function() {
  // Code ... });
```

online
```
window.addEventListener('online', function() {
  // Code ... });
```

offline
```
window.addEventListener('offline', function() {
  // Code ... });
```

Cookies in Javascript

Cookies are small pieces of data that a website can store on your computer or mobile device. They are used to remember things about users, such as user preferences, login status, and shopping cart contents etc. Cookies can also be used to track

your browsing activity across multiple websites.

The **document.cookie** property also has the following properties:

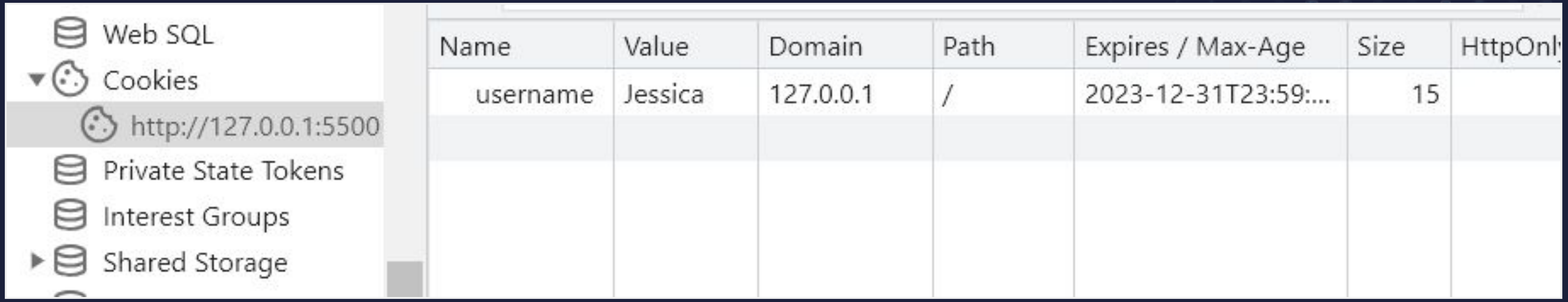
- length: The number of cookies that are associated with the current document.
- expires: The date and time at which the cookie will expire.
- path: The path on the server that the cookie is valid for.
- domain: The domain that the cookie is valid for.
- secure: Whether or not the cookie can only be sent over a secure connection.
- httpOnly: Whether or not the cookie can be accessed by JavaScript.

The **document.cookie** property has the following methods:

- Gets the value of a cookie by name.
- Sets the value of a cookie with some of the properties.
- Removes a cookie by setting expires date to the past date.

Example of a document.cookie methods-

```
index.html
<html lang="en">
  <body>
    <h1>Cookie demo</h1>
  </body>
  <script src="./script.js"></script>
</html>
script.js
/** create method cookie **/
// syntax - document.cookie = "key=value; //expires=expiration_date; path=path_value";
document.cookie =
  "username=Jessica; expires=Fri, 31 Dec 2023 23:59:59 GMT; path=/";
// the path refers to the root path, it can be modified to //desire path of the website.
// getting cookie value
let cookieValue = document.cookie;
console.log(cookieValue); // output - username=Jessica
```

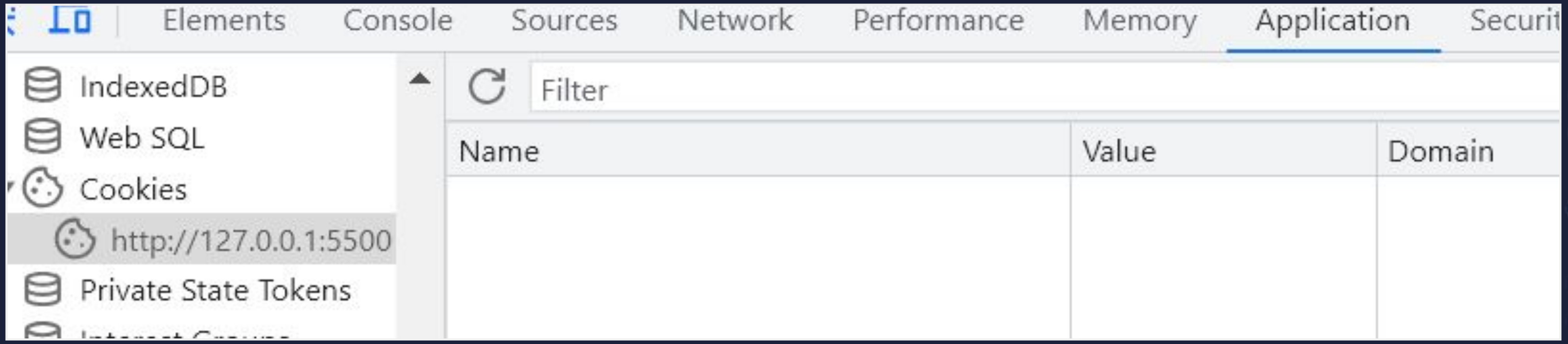


Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnl
username	Jessica	127.0.0.1	/	2023-12-31T23:59:...	15	

script.js

```
// deleting cookie
// to delete - expiration date is set to a past date.
document.cookie = "username=; expires=Thu, 01 Jan 1970 00:00:00 GMT; path=/";
```

output -



Name	Value	Domain

It's important to keep in mind that cookies have some limitations, such as their size limit (usually 4KB) and the fact that they can only store string values. For more complex data storage, you may want to consider using other mechanisms, such as Local Storage or IndexedDB.

SKILLS

THANK YOU