JavaScript localStorage

LocalStorage is a data storage type of web storage. This allows the JavaScript sites and apps to store and access the data without any expiration date. This means that the data will always be persisted and will not expire. So, data stored in the browser will be available even after closing the browser window.

In short, all we can say is that the localStorage holds the data with no expiry date, which is available to the user even after closing the browser window. It is useful in various ways, such as remembering the shopping cart data or user login on any website.

In the past days, cookies were the only option to remember this type of temporary and local information, but now we have localStorage as well. Local storage comes with a higher storage limit than cookies (5MB vs 4MB). It also does not get sent with every HTTP request. So, it is a better choice now for client-side storage. Some essential points of localStorage need to be noted:

- localStorage is not secure to store sensitive data and can be accessed using any code. So, it is quite insecure.
- It is an advantage of localStorage over cookies that it can store more data than cookies. You can store 5MB data on the browser using localStorage.
- localStorage stores the information only on browser instead in database. Thereby the localStorage is not a substitute for a server-based database.
- localStorage is synchronous, which means that each operation executes one after another.

localStorage Methods

The localStorage offers some methods to use it. We will discuss all these localStorage methods with examples. Before that, a basic overview of these methods are as follows:

⇧ SCROLL TO TOP

Methods	Description
setItem()	This method is used to add the data through key and value to localStorage.
getItem()	It is used to fetch or retrieve the value from the storage using the key.
removeItem()	It removes an item from storage by using the key.
clear()	It is used to gets clear all the storage.

Each of these methods is used with localStorage keyword connecting with dot(.) character. **For Example:** localStorage.setItem().

> Remember that localStorage property is read-only.

⇧ SCROLL TO TOP

Following some codes given, which are used to add, retrieve, remove, and clear the data in localStorage. Use them in your code accordingly whenever needed. You need a key-value pair to add some data in localStorage. So, let key is city and its value is Noida, **i.e.,** key: value = city: Noida.

Add data

To add the data in localStorage, both key and value are required to pass in setItem() function.

```
localStorage.setItem("city", "Noida");
```

Retrieve data

It requires only the key to retrieve the data from storage and a JavaScript variable to store the returned data.

```
const res = localStorage.getItem("city");
```

Remove data

It also requires only the key to remove the value attached with it.

⇧ SCROLL TO TOP

```
eItem("city");
```

Clear localStorage

It is a simple clear() function of localStorage, which is used to remove all the localStorage data:

```
localStorage.clear()
```

Limitation of localStorage

As the localStorage allows to store temporary, local data, which remains even after closing the browser window, but it also has few limitations. Below are some limitations of localStorage are given:

- Do not store sensitive information like username and password in localStorage.
- localStorage has no data protection and can be accessed using any code. So, it is quite insecure.
- You can store only maximum 5MB data on the browser using localStorage.
- localStorage stores the information only on browser not in server-based database.
- localStorage is synchronous, which means that each operation executes one after another.

Advantage of localStorage

The localStorage has come with several advantages. First and essential advantage of localStorage is that it can store temporary but useful data in the browser, which remains even after the browser window closed. Below is a list of some advantages:

- The data collected by localStorage is stored in the browser. You can store 5 MB data in the browser.
- There is no expiry date of data stored by localStorage.
- You can remove all the localStorage item by a single line code, i.e., **clear()**.

⇧ SCROLL TO TOP

,e data persist even after closing the browser window, like items in shopping cart.

- It also has advantages over cookies because it can store more data than cookies.

Browser compatibility

The localStorage has specified in HTML 5, which is supported by several browsers, like Chrome. Below is a list of different browsers and their versions that supports JavaScript localStorage.

Browser	Chrome	Internet Explorer	Firefox	Opera	Safari	Edge
Version support	4.0	8.0	3.5	11.5	4	12

JavaScript code to check browser compatibility

With the help of below code example, you can check the browser compatibility. Use this code in your every localStorage program to check the browser compatibility before adding or deleting something from localStorage.

⇧ SCROLL TO TOP

```
// Code to check browser support
if (typeof(Storage) !== "undefined") {
   //browser support localStorage
} else {
  //browser does not support localStorage
}
</script>
```

Output

```
Undefined
```

Example

It is a basic example of adding a key and value to localStorage and retrieving back by the key. See the code below how localStorage methods work:

```
<html>
<body>

<div id="result"></div>
```

⇧ SCROLL TO TOP

```
<script>
// Check browser support
if (typeof(Storage) !== "undefined") {
  // Store an item to localStorage
  localStorage.setItem("firstname", "Alen");

  // Retrieve the added item
  document.getElementById("result").innerHTML = localStorage.getItem("firstname");

} else {
  //display this message if browser does not support localStorage
  document.getElementById("result").innerHTML = "Sorry, your browser does not support Web Storage.";
}
</script>
</body>
</html>
```

Test it Now

Output

```
Alen
```

More Examples

⇧ SCROLL TO TOP

It is an example to count the button clicks means that it will count how many times a user clicks the button. In this example, we will create two buttons, one for counting the user clicks and another for clear the that clicks data.

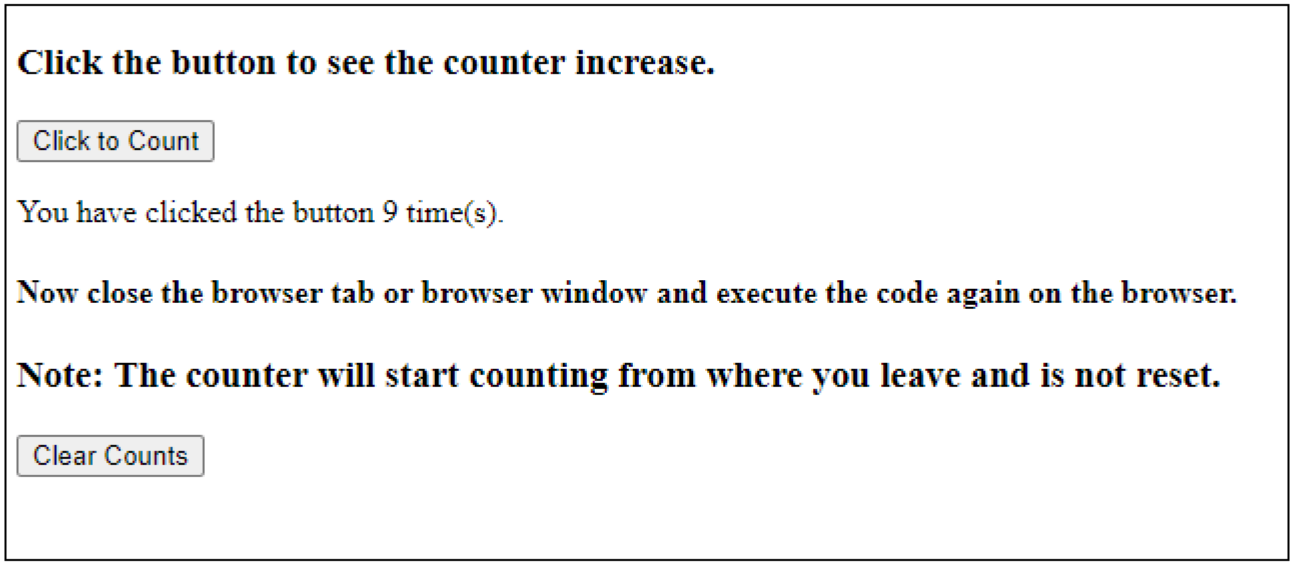
```
<html>
<head>
<script>
//function to count the button clicks
function clickCounting() {
  if(typeof(Storage) !== "undefined") {
    if (localStorage.clickcount) {
      localStorage.clickcount = Number(localStorage.clickcount)+1;
    } else {
      localStorage.clickcount = 1;
    }
    document.getElementById("result").innerHTML = "You have clicked the button " + localStorage.clickcount + " time(s).";
  }
    //when the browser does not support
  else {
    document.getElementById("result").innerHTML = "Your browser does not support web storage.";
  }
}

//function to clear the data stored by browser
function clearCounting() {
⇧ SCROLL TO TOP   age.clear();
}
</script>
</head>
<body>

<h3> Click the button to see the counter increase.</h3>
<p> <button onclick="clickCounting()" type="button">Click to Count</button></p>
<div id="result"> </div>
<h4> Now close the browser tab or browser window and execute the code again on the browser.  <h4>
<h3>Note: The counter will start counting from where you leave and is not reset.</h3>
<p> <button onclick="clearCounting()" type="button">Clear Count</button></p>

</body>
</html>
```

Test it Now

Output 1

In the below output, you can see that we have clicked the **Click to Count** button 9 times.

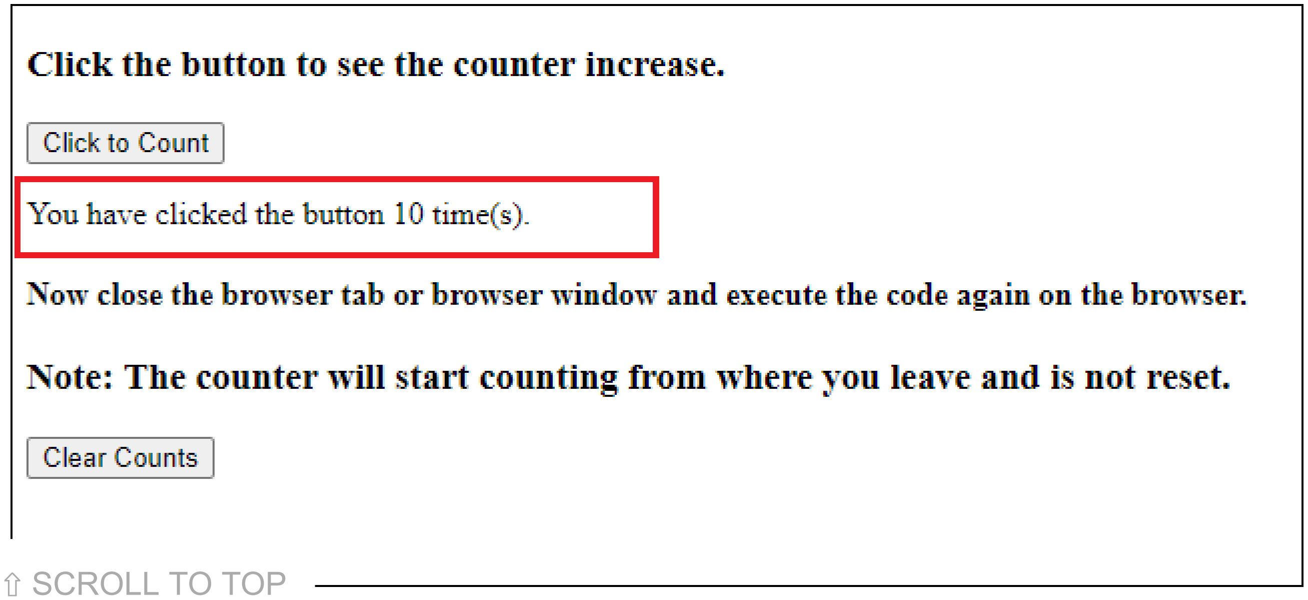
⇧ SCROLL TO TOP



```
Click the button to see the counter increase.

[Click to Count]

You have clicked the button 9 time(s).

Now close the browser tab or browser window and execute the code again on the browser.

Note: The counter will start counting from where you leave and is not reset.

[Clear Counts]
```

Output 2

Now close the window tab and reopen to run the code again. Again, click on the **Click to Count** button, it will start counting from 10 where you left.

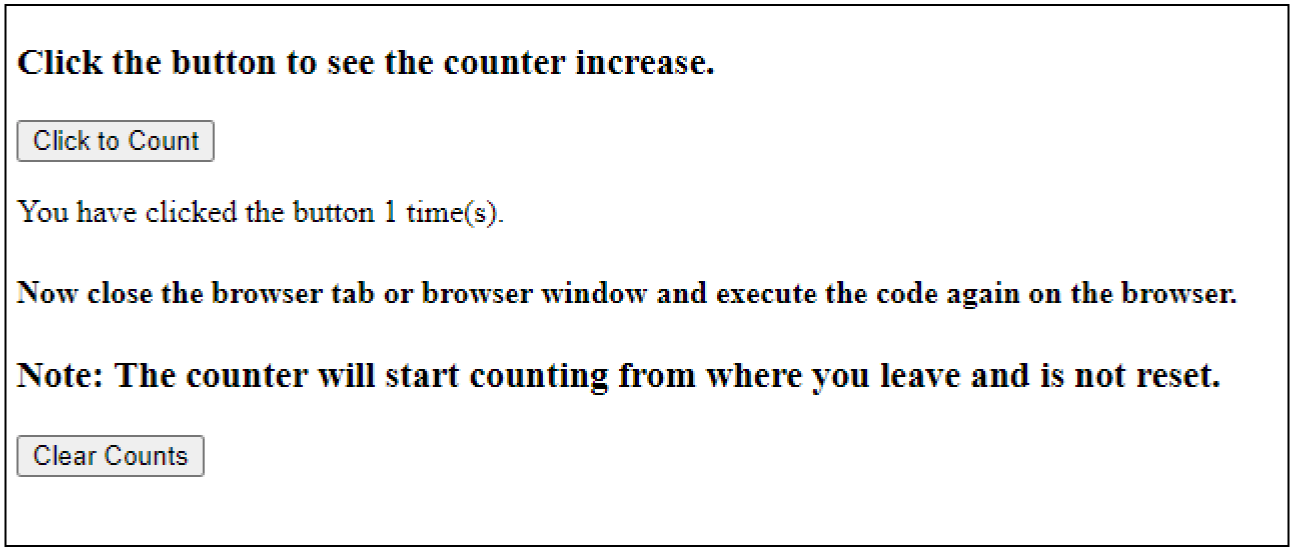


```
Click the button to see the counter increase.

[Click to Count]

You have clicked the button 10 time(s).

Now close the browser tab or browser window and execute the code again on the browser.

Note: The counter will start counting from where you leave and is not reset.

[Clear Counts]
```

⇧ SCROLL TO TOP

Output 3

Now, click the **Clear Count** button to clear the stored data. When you again click on the **Click to Count** button, it will again start with 1.



Click the button to see the counter increase.

You have clicked the button 1 time(s).

Now close the browser tab or browser window and execute the code again on the browser.

Note: The counter will start counting from where you leave and is not reset.

Clear all records

Clear() method of localStorage is used to clear the entire storage data. When this method invokes, it clears all the records for that domain from the storage. It does not contain any parameters. See the syntax to clear the localStorage:

```
window.localStorage.clear();
```

Or

```
localStorage.clear();
```

We will use this clear code in below example.

Check localStorage

On the JavaScript console, you can check what is in local storage by typing **localStorage** command on it. Even if there nothing in localStorage, it has length = 0 inside it.

Command