JavaScript TUTORIAL

# JavaScript sessionStorage

**Summary**: in this tutorial, you'll learn how to use the JavaScript `sessionStorage` to store data only for a session.

## Introduction to JavaScript sessionStorage

The `sessionStorage` object stores data only for a session. It means that the data stored in the `sessionStorage` will be deleted when the browser is closed.

A page session lasts as long as the web browser is open and survives over the page refresh.

When you open a page in a new tab or window, the web browser creates a new session.

If you open multiple tabs or windows with the same URL, the web browser creates a separate `sessionStorage` for each tab or window. So data stored in one web browser tab cannot be accessible in another tab.

When you close a tab or window, the web browser ends the session and clears data in the `sessionStorage` .

Data stored in the `sessionStorage` is specific to the protocol of the page. For example, the same site `javascripttutorial.net` has different `sessionStorage` when accessing with the `http` and `https` .

Since the `sessionStorage` data is tied to a server session, it's only available when a page is requested from a server. The `sessionStorage` isn't available when the page runs locally without a server.

Because the `sessionStorage` is an instance of the `Storage` type, you can manage data using the Storage's methods:

- `setItem(name, value)` – set the value for a name
- `removeItem(name)` – remove the name-value pair identified by name.

- `getItem(name)` – get the value for a given name.
- `key(index)` – get the name of the value in the given numeric position.
- `clear()` – remove all values in the `sessionStorage` .

## Managing data in the JavaScript sessionStorage

### 1) Accessing the sessionStorage

To access the `sessionStorage` , you use the `sessionStorage` property of the `window` object:

```
window.sessionStorage
```

Since the `window` is the global object, you can simply access the `sessionStorage` like this:

```
sessionStorage
```

### 2) Storing data in the sessionStorage

The following stores a name-value pair in the `sessionStorage` :

```
sessionStorage.setItem('mode','dark');
```

If the `sessionStorage` has an item with the name of `mode` , the `setItem()` method will update the value for the existing item to `dark` . Otherwise, it'll insert a new item.

### 3) Getting data from the sessionStorage

To get the value of an item by name, you use the `getItem()` method. The following example gets the value of the item ' `mode` ':

```
const mode = sessionStorage.getItem('mode');
console.log(mode); // 'dark'
```

If there is no item with the name `mode` , the `getItem()` method will return `null` .

### 4) Removing an item by a name

To remove an item by the name, you use the `removeItem()` method. The following removes the item with the name of `'mode'` :

```
sessionStorage.removeItem('mode');
```

### 5) Iterating over all items

To iterate over all items stored in the `sessionStorage` , you follow these steps:

- Use `Object.keys()` to get all keys of the `sessionStorage` object.
- Use `for...of` to iterate over the keys and get the items by keys.

The following code illustrates the steps:

```
let keys = Object.keys(sessionStorage);
for(let key of keys) {
  console.log(`${key}: ${sessionStorage.getItem(key)}`);
}
```

### 6) Deleting all items in the sessionStorage

The data stored in the `sessionStorage` are automatically deleted when the web browser tab/window is closed.

In addition, you can use the `clear()` method to programmatically delete all data stored in the `sessionStorage` .

```
sessionStorage.clear();
```

## Why JavaScript sessionStorage

The `sessionStorage` has many practical applications. And the following are the notable ones:j

- The `sessionStorage` can be used to store the state of the user interface of the web application. Later, when the user comes back to the page, you can restore the user interface stored in the `sessionStorage` .

- The `sessionStorage` can also be used to pass data between pages instead of using the hidden input fields or URL parameters.

## JavaScript sessionStorage application

You'll build a simple web application that allows users to select the mode, either dark or light mode. By default, it has a light mode. And you'll use the `sessionStorage` to remember the mode when the page refreshes.

If you refresh the page, the mode that you selected will restore since it's stored in the `sessionStorage` .

However, if you close the tab or window, the page will reset to the `dark` mode, which is the default mode.

### 1) Creating the project folder structure

First, create a new folder called `session-storage` . In the `session-storage` folder, create two subfolders: `js` and `css` that will store the JavaScript and CSS files.

Second, create a new `index.html` in the `sessionStorage` folder, the `app.js` file in the `js` folder, and `style.css` file in the `css` folder.

### 2) Building the HTML page

The following shows the `index.html` page:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>JavaScript sessionStorage Demo</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
    <div class="container">
```

```
        <h1>JavaScript sessionStorage Demo</h1>
        <p>Click the button to switch to the dark/light mode.</p>
        <p>Refresh the page to check if the mode is saved.</p>
        <a id="theme-switcher" class="btn"></a>
    </div>
    <script src="js/app.js"></script>
</body>

</html>
```

In this `index.html` file, we place the `style.css` in the head section and `app.js` in the body section.

The page has some elements. The most important one is the button with the id `theme-switcher` .

## 3) Creating app.js file

First, declare two constants that will be used as the butotn’s label:

```
const MOON = '🌙';
const SUN = '🌞';
```

You’ll use the `SUN` as the label of the `theme-switcher` button in the dark mode and `MOON` in the light mode.

Second, declare three constants for the dark, light, and default modes:

```
const DARK_MODE = 'dark';
const LIGHT_MODE = 'light';
const DEFAULT_MODE = DARK_MODE;
```

Third, select the button `theme-switcher` by using the `querySelector()` :

```
const btn = document.querySelector('#theme-switcher');
```

Fourth, define a new function `setMode()` to change the mode:

```
function setMode(mode = DEFAULT_MODE) {
    if (mode === DARK_MODE) {
        btn.textContent = SUN;
        document.body.classList.add(DARK_MODE);

    } else if (mode === LIGHT_MODE) {
        btn.textContent = MOON;
        document.body.classList.remove(DARK_MODE);
    }
}
```

In the dark mode, the `setMode()` changes the button to `SUN` and adds the `DARK_MODE` class to the body element

And in the light mode, the `setMode()` changes the button label to `MOON` and removes the `DARK_MODE` class from the body element.

The following shows the CSS of the light mode. The background color is white and the text color is black:

```
body {
    font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', Roboto, Oxygen, Ubuntu, Canta
    font-size: 16px;
    background-color: #fff;
    color: #333;
    line-height: 1.7;
    transition: 0.2s ease-in-out;
    padding: 20px;
}
```

In the dark mode, the background color is black and the text color is white:

```
.dark {
    background-color: black;
    color: #fff;
}
```

To switch from the light to dark mode, you add the .dark class to the body element and vice versa.

Fifth, define the `init()` function that will run when the page loads:

```
function init() {
    let storedMode = sessionStorage.getItem('mode');
    if (!storedMode) {
        storedMode = DEFAULT_MODE;
        sessionStorage.setItem('mode', DEFAULT_MODE);
    }
    setMode(storedMode);
}
```

In this function, we use the `getItem()` method to retrieve the mode stored in the `sessionStorage` .

If the `sessionStorage` doesn’t have the `mode` item, the `init()` function will switch the page to the default mode, which is the dark mode. Otherwise, it sets to the mode stored in the `sessionStorage` .

Sixth, attach a click event handler to the `theme-switcher` button:

```
btn.addEventListener('click', function () {
    let mode = sessionStorage.getItem('mode');
    if (mode) {
        let newMode = mode == DARK_MODE ? LIGHT_MODE : DARK_MODE;
        setMode(newMode);
        sessionStorage.setItem('mode', newMode);
    }
});
```

The click event handler gets the mode stored in the sessionStorage.

If the mode item exists, it toggles the mode. In other words, the light mode becomes the dark mode and vice versa.

It then uses the `setItem()` method to update the mode item in the `sessionStorage` to the new one.

The following shows a complete app.js file:

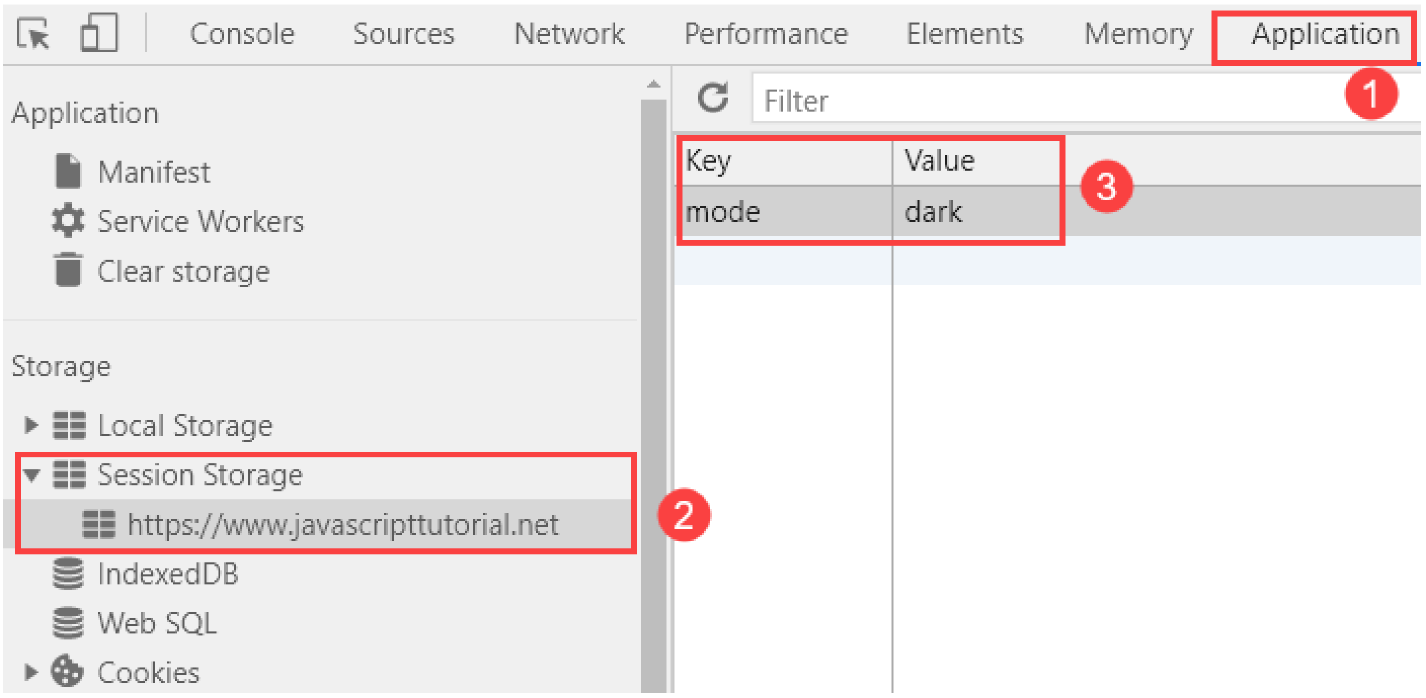
```
const MOON = '🌙';
const SUN = '🌞';
const DARK_MODE = 'dark';
const LIGHT_MODE = 'light';
const DEFAULT_MODE = DARK_MODE;

const btn = document.querySelector('#theme-switcher');

init();

function init() {
    let storedMode = sessionStorage.getItem('mode');
    if (!storedMode) {
        storedMode = DEFAULT_MODE;
        sessionStorage.setItem('mode', DEFAULT_MODE);
    }
    setMode(storedMode);
}

function setMode(mode = DEFAULT_MODE) {
    if (mode === DARK_MODE) {
        btn.textContent = SUN;
        document.body.classList.add(DARK_MODE);

    } else if (mode === LIGHT_MODE) {
        btn.textContent = MOON;
        document.body.classList.remove(DARK_MODE);
    }
}

btn.addEventListener('click', function () {
    let mode = sessionStorage.getItem('mode');
    if (mode) {
        let newMode = mode == DARK_MODE ? LIGHT_MODE : DARK_MODE;
        setMode(newMode);
        sessionStorage.setItem('mode', newMode);
    }
});
```

Here is the final application.

First, you select a mode e.g., light mode, the `sessionStorage` will save it.

Then, you refresh the page. It'll show the previously selected mode.

To view the data stored in the session storage in the web browser, you click the Application tab and select the Session Storage:



| Key | Value |
|---|---|
| mode | dark |

## Summary

- The `sessionStorage` allows you to store the data for session only. The browser will delete the `sessionStorage` data when you close the browser tab or window.
- The `sessionStorage` is an instance of the `Storage` type, therefore, you can use the methods of the Storage type to manage data in the `sessionStorage` .