Window: sessionStorage property

- Value
  - Exceptions
- Examples
  - Basic usage
  - Saving text between refreshes
- Specifications
- Browser compatibility
- See also

The read-only **`sessionStorage`** property accesses a session Storage object for the current origin. `sessionStorage` is similar to localStorage; the difference is that while `localStorage` is partitioned by origin only, `sessionStorage` is partitioned by both origin and browser tabs (top-level browsing contexts). The data in `sessionStorage` is only kept for the duration of the page session.

- Whenever a document is loaded in a particular tab in the browser, a unique page session gets created and assigned to that particular tab. That page session is accessible only in that particular tab. The main document, and all embedded browsing contexts (iframes), are grouped by their origin and each origin has access to its own separate storage area.
- If the page has an opener, the `sessionStorage` is initially a copy of the opener's `sessionStorage` object. However, they are still separate and changes to one do not affect the other. To prevent the `sessionStorage` from being copied, use one of the techniques that remove `opener` (see Window.opener).
- A page session lasts as long as the tab or the browser is open, and survives over page reloads and restores.
- Opening a page in a new tab or window creates a new session with the value of the top-level browsing context, which differs from how session cookies work.
- Closing the tab/window ends the session and clears the data in `sessionStorage`.

Value

A Storage object which can be used to access the current origin's session storage space.

Exceptions

- `SecurityError` DOMException
  - : Thrown in one of the following cases:
    - The origin is not a valid scheme/host/port tuple. This can happen if the origin uses the `file:` or `data:` schemes, for example.
    - The request violates a policy decision. For example, the user has configured the browsers to prevent the page from persisting data.

    Note that if the user blocks cookies, browsers will probably interpret this as an instruction to prevent the page from persisting data.

Examples

Basic usage

```
// Save data to sessionStorage
sessionStorage.setItem("key", "value");

// Get saved data from sessionStorage
let data = sessionStorage.getItem("key");

// Remove saved data from sessionStorage
sessionStorage.removeItem("key");

// Remove all saved data from sessionStorage
sessionStorage.clear();
```

Saving text between refreshes

The following example autosaves the contents of a text field, and if the browser is refreshed, restores the text field content so that no writing is lost.

```
// Get the text field that we're going to track
let field = document.getElementById("field");

// See if we have an autosave value
// (this will only happen if the page is accidentally refreshed)
if (sessionStorage.getItem("autosave")) {
  // Restore the contents of the text field
  field.value = sessionStorage.getItem("autosave");
}

// Listen for changes in the text field
field.addEventListener("change", () => {
  // And save the results into the session storage object
  sessionStorage.setItem("autosave", field.value);
});
```

> **NOTE:** Please refer to the Using the Web Storage API article for a full example.

Specifications

Browser compatibility

See also

- Using the Web Storage API
- Window.localStorage

□□□ #3
□ □□□ □□□ 2026-10-31 03:27:53 UTC
□ cocobook-sync □□□ 2026-11-04 17:35:21 UTC