The read-only **localStorage** property allows you to access a `Storage` object for the `Document`'s origin; the stored data is saved across browser sessions. `localStorage` is similar to `sessionStorage`, except that while data stored in `localStorage` has no expiration time, data stored in `sessionStorage` gets cleared when the page session ends — that is, when the page is closed. (Data in a `localStorage` object created in a "private browsing" or "incognito" session is cleared when the last "private" tab is closed.)

Data stored in either `localStorage` **is specific to the protocol of the page**. In particular, data stored by a script on a site accessed with HTTP (e.g., http://example.com) is put in a different `localStorage` object from the same site accessed with HTTPS (e.g., https://example.com).

The keys and the values are *always* in the UTF-16 `DOMString` format, which uses two bytes per character. As with objects, integer keys are automatically converted to strings.

Syntax

```
myStorage = window.localStorage;
```

Value

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

Exceptions

`SecurityError`
The request violates a policy decision, or the origin is not a valid scheme/host/port tuple (this can happen if the origin uses the `file:` or `data:` scheme, for example). For example, the user may have their browser configured to deny permission to persist data for the specified origin.

Example

The following snippet accesses the current domain's local `Storage` object and adds a data item to it using `Storage.setItem()`.

```
localStorage.setItem('myCat', 'Tom');
```

The syntax for reading the `localStorage` item is as follows:

```
const cat = localStorage.getItem('myCat');
```

The syntax for removing the `localStorage` item is as follows:

```
localStorage.removeItem('myCat');
```

The syntax for removing all the `localStorage` items is as follows:

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

Note: Please refer to the Using the Web Storage API article for a full example.