Chapter 33: Web Storage

Parameter	Description
name	The key/name of the item
value	The value of the item

Section 33.1: Using localStorage

The localStorage object provides persistent (but not permanent - see limits below) key-value storage of strings. Any changes are immediately visible in all other windows/frames from the same origin. The stored values persistent indefinitely unless the user clears saved data or configures an expiration limit. localStorage uses a map-like interface for getting and setting values.

```
localStorage.setItem('name', "John Smith");
console.log(localStorage.getItem('name')); // "John Smith"

localStorage.removeItem('name');
console.log(localStorage.getItem('name')); // null
```

If you want to store simple structured data, you can use JSON to serialize it to and from strings for storage.

```
var players = [{name: "Tyler", score: 22}, {name: "Ryan", score: 41}];
localStorage.setItem('players', JSON.stringify(players));

console.log(JSON.parse(localStorage.getItem('players')));
// [ Object { name: "Tyler", score: 22 }, Object { name: "Ryan", score: 41 } ]
```

localStorage limits in browsers

Mobile browsers:

Browser	Google Chrome Android Browser	Firefox	iOS Safari
Version	40	4.3	34	6-8
Space available	10MB	2MB	10MB	5MB

Desktop browsers:

Browser	Google Chrome	Opera	Firefox	Safari	Internet Explorer
Version	40	27	34	6-8	9-11
Space available	10MB	10MB	10MB	5MB	10MB

Section 33.2: Simpler way of handling Storage

`localStorage`, `sessionStorage` are JavaScript **Objects** and you can treat them as such.
Instead of using Storage Methods like `.getItem()`, `.setItem()`, etc… here's a simpler alternative:

```
// Set
localStorage.greet = "Hi!"; // Same as: window.localStorage.setItem("greet", "Hi!");

// Get
localStorage.greet;         // Same as: window.localStorage.getItem("greet");

// Remove item
delete localStorage.greet;  // Same as: window.localStorage.removeItem("greet");
```

```
// Clear storage
localStorage.clear();
```

Example:

```
// Store values (Strings, Numbers)
localStorage.hello = "Hello";
localStorage.year  = 2017;

// Store complex data (Objects, Arrays)
var user = {name:"John", surname:"Doe", books:["A","B"]};
localStorage.user = JSON.stringify( user );

// Important: Numbers are stored as String
console.log( typeof localStorage.year ); // String

// Retrieve values
var someYear = localStorage.year; // "2017"

// Retrieve complex data
var userData = JSON.parse( localStorage.user );
var userName = userData.name; // "John"

// Remove specific data
delete localStorage.year;

// Clear (delete) all stored data
localStorage.clear();
```

Section 33.3: Storage events

Whenever a value in set in localStorage, a `storage` event will be dispatched on all other `windows` from the same origin. This can be used to synchronize state between different pages without reloading or communicating with a server. For example, we can reflect the value of an input element as paragraph text in another window:

First Window

```
var input = document.createElement('input');
document.body.appendChild(input);

input.value = localStorage.getItem('user-value');

input.oninput = function(event) {
  localStorage.setItem('user-value', input.value);
};
```

Second Window

```
var output = document.createElement('p');
document.body.appendChild(output);

output.textContent = localStorage.getItem('user-value');

window.addEventListener('storage', function(event) {
  if (event.key === 'user-value') {
    output.textContent = event.newValue;
  }
});
```

Notes

Event is not fired or catchable under Chrome, Edge and Safari if domain was modified through script.

First window

```
// page url: http://sub.a.com/1.html
document.domain = 'a.com';

var input = document.createElement('input');
document.body.appendChild(input);

input.value = localStorage.getItem('user-value');

input.oninput = function(event) {
  localStorage.setItem('user-value', input.value);
};
```

Second Window

```
// page url: http://sub.a.com/2.html
document.domain = 'a.com';

var output = document.createElement('p');
document.body.appendChild(output);

// Listener will never called under Chrome(53), Edge and Safari(10.0).
window.addEventListener('storage', function(event) {
  if (event.key === 'user-value') {
    output.textContent = event.newValue;
  }
});
```

Section 33.4: sessionStorage

The sessionStorage object implements the same Storage interface as localStorage. However, instead of being shared with all pages from the same origin, sessionStorage data is stored separately for every window/tab. Stored data persists between pages *in that window/tab* for as long as it's open, but is visible nowhere else.

```
var audio = document.querySelector('audio');

// Maintain the volume if the user clicks a link then navigates back here.
audio.volume = Number(sessionStorage.getItem('volume') || 1.0);
audio.onvolumechange = function(event) {
  sessionStorage.setItem('volume', audio.volume);
};
```

Save data to sessionStorage

```
sessionStorage.setItem('key', 'value');
```

Get saved data from sessionStorage

```
var data = sessionStorage.getItem('key');
```

Remove saved data from sessionStorage

```
sessionStorage.removeItem('key')
```

Section 33.5: localStorage length

`localStorage.length` property returns an integer number indicating the number of elements in the `localStorage`

Example:

Set Items

```
localStorage.setItem('StackOverflow', 'Documentation');
localStorage.setItem('font', 'Helvetica');
localStorage.setItem('image', 'sprite.svg');
```

Get length

```
localStorage.length; // 3
```

Section 33.6: Error conditions

Most browsers, when configured to block cookies, will also block `localStorage`. Attempts to use it will result in an exception. Do not forget to manage these cases.

```
var video = document.querySelector('video')
try {
    video.volume = localStorage.getItem('volume')
} catch (error) {
    alert('If you\'d like your volume saved, turn on cookies')
}
video.play()
```

If error were not handled, program would stop functioning properly.

Section 33.7: Clearing storage

To clear the storage, simply run

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

Section 33.8: Remove Storage Item

To remove a specific item from the browser Storage (the opposite of `setItem`) use `removeItem`

```
localStorage.removeItem("greet");
```

Example:

```
localStorage.setItem("greet", "hi");
localStorage.removeItem("greet");

console.log( localStorage.getItem("greet") ); // null
```

(Same applies for `sessionStorage`)