WEB STORAGE

by Maksym Khudoliy

softserve

WEB STORAGE

Web Storage introduces storage in the browser and includes two objects **sessionStorage** and **localStorage**:

- in sessionStorage the data is stored **temporally** and will be **automatically deleted** after **closing** the browser **tab**
- in localStorage, the data is **not automatically deleted** and will exist even after restarting the browser

In spite of Web Storage data can be stored in a **cookie**, but there are a few differences that you need to consider when choosing a storage location:

- data from Web Storage is **not sent to the server** every time a request is made
- the size of the Web Storage is **much larger** than a cookie
- server **cannot directly manipulate** data in Web Storage

WEB STORAGE

The sessionStorage and localStorage objects represent **data** as a set of **key: value pairs**.
The **same set** of properties and methods are used to work with objects:

- **setItem(key, value)** – saves a key: value pair, if the key already existed, the value will be update
- **getItem(key)** – return value by key
- **removeItem(key)** – remove pair with key
- **clear()** – clear all data
- **key(index)** – return key with the specified index
- **length** – number of pair in storage

WEB STORAGE

```
localStorage.setItem("user", "Tom");
localStorage.setItem("role", "guest");
console.log(localStorage.getItem("user")); // "Tom"
console.log(localStorage.getItem("role")); // "guest"
console.log(localStorage.length); // 2
console.log(localStorage.key(0)); // "user"
localStorage.setItem("user", "Bob");
console.log(localStorage.getItem("user")); // "Bob"
localStorage.removeItem("role");
console.log(localStorage.length); // 1
localStorage.clear();
console.log(localStorage.length); // 0
```

softserve

WEB STORAGE

Please note, that both key and value **must be strings only**, this lay on some particular qualities when working with non-string data:

```
localStorage.setItem("num", 10);

const num = localStorage.getItem("num");

console.log(typeof num); // "string"
```

WEB STORAGE

To store **complex data**, such as objects or arrays, you need to use **serializations to JSON** format:

```
localStorage.setItem("data1", [1, 2, 3]);

localStorage.setItem("data2", JSON.stringify([1, 2, 3]));

const data1 = localStorage.getItem("data1");

console.log(data1 instanceof Array); // false

console.log(data1); // "1,2,3"

const data2 = JSON.parse(localStorage.getItem("data2"));

console.log(data2 instanceof Array); // true

console.log(data2); // [1, 2, 3]
```

softserve

WEB STORAGE

When the data in localStorage or sessionStorage is **updated**, a "**storage**" event is fired with the following properties:

- **key** – the key, which updated (null, if called clear()):
- **oldValue** – the old value of the changed storage pair (null, if the pair added firstly)
- **newValue** – the new value of the changed storage pair (null, if the pair was deleted)
- **url** – url of the document where the update took place
- **storageArea** – the localStorage or sessionStorage object where the update occurred

Please note, that the event is triggered **on all other browser tabs** where storage is available, **except for the tab where it happened**. This mechanism allows synchronization of tabs and exchange of messages

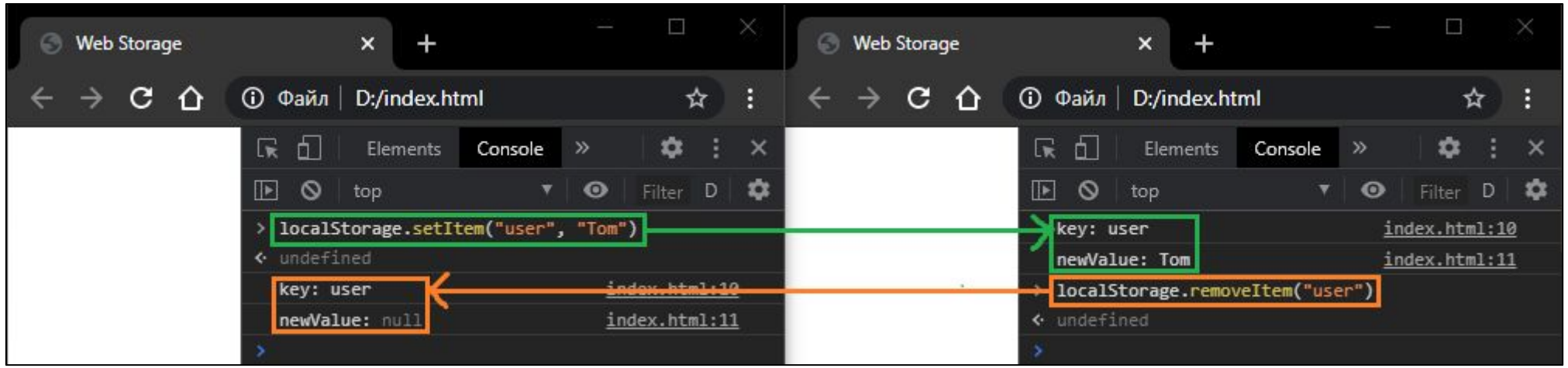
WEB STORAGE

To demonstrate the work of the "storage" event, create an **index.html** file the following content:

```
<body>
    <script>
        window.addEventListener("storage", (e) => {
            console.log("key:", e.key);
            console.log("newValue:", e.newValue);
        });
    </script>
</body>
```

WEB STORAGE

Open index.html in two tabs of one browser and call several methods from the localStorage object:



softserve


TO
THE
FUTURE
softserve