JavaScript localStorage

Summary: in this tutorial, you'll learn about the `Storage` type and how to use the JavaScript `localStorage` to store persistent data.

Introduction to the Storage type

The `Storage` type is designed to store name-value pairs. The `Storage` type is an `Object` with the following additional 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.

To get the number of name-value pairs in a `Storage` object, you can use the `length` property.

The `Storage` object can store only strings. It'll automatically convert non-string data into a string before storing it.

When you retrieve data from a `Storage` object, you'll always get the string data.

The JavaScript localStorage object

HTML5 specification introduces the `localStorage` as a way to store data with no expiration date in web browsers.

In other words, the data stored in the browsers will persist even after you close the browser windows.

The data stored in the `localStorage` is bound to an origin. It means that the `localStorage` is unique per `protocol://host:port` .

localStorage vs. cookies

First, the data stored in the `localStorage` isn't sent to the server in every request like cookies. For this reason, you can store more data in the `localStorage` .

Most modern web browsers allow you to store up to 5MB of data in the `localStorage` . Note that you can store up to 4KB in cookies.

Second, the data stored in the `localStorage` can be managed by the client, specifically JavaScript in the web browser. It cannot be accessible by the servers.

However, cookies can be managed by both JavaScript in web browsers and servers.

Accessing the localStorage

You can access the `localStorage` via the property of the `window` object:

```
window.localStorage
```

Since the `localStorage` is an instance of the `Storage` type, you can invoke the methods of the `Storage` type to manage data.

When you type the following code in the Console:

```
window.localStorage
```

... you'll see the following object:

```
Storage {length: 0}
```

1) The setItem() method

The following uses the `setItem()` method to store a name-value pair in the `localStorage` :

```
window.localStorage.setItem('theme','dark');
```

2) The length property

To get the number of name-value pairs, you use the `length` property like this:

```
console.log(window.localStorage.length); // 1
```

Since the `window` object is global, you don't need to explicitly specify it. For example:

```
console.log(localStorage.length); // 1
```

3) The getItem() method

To get the value by a key, you use the `getItem()` method. The following example uses the `getItem()` method to get the value of `theme` key:

```
localStorage.getItem('theme'); // 'dark'
```

4) The removeItem() method

To remove a name-value pair by a key, you use the `removeItem()` method. For example:

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

5) Loop over keys of the localStorage object

The following stores three name-value pairs to the `localStorage` :

```
localStorage.setItem('theme','light');
localStorage.setItem('backgroundColor','white');
localStorage.setItem('color','#111');
```

To iterate over name-value pairs stored in the `localStorage` , you use the `Object.keys()` method with `for...of` loop:

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

Output:

```
color: #111
theme: light
backgroundColor: white
```

Storing objects

The `Storage` type stores only string data. To store objects, you need to convert them into strings using the `JSON.stringify()` method. For example:

```
const settings = {
    backgroundColor: '#fff',
    color: '#111',
    theme: 'light'
};

localStorage.setItem('settings', JSON.stringify(settings));

console.log(localStorage.getItem('settings'));
```

Output: (a string)

```
'{"backgroundColor":"#fff","color":"#111","theme":"light"}'
```

The following retrieves the value from the `localStorage` and converts it back to the object using the `JSON.parse()` method.

```
let storedSettings = JSON.parse(localStorage.getItem('settings'));
```

```
console.log(storedSettings);
```

The storage event

When you make a change to the `Storage` object, the `storage` event is fired on the document.

The `storage` event occurs in the following scenarios:

- Store a name-value pair by calling the `setItem()` method.
- Remove a name-value pair by calling the `removeItem()` method.
- And remove all values by calling the `clear()` method.

The `storage` event has the following properties:

- `domain` – the domain which the storage changes for.
- `key` – the key that was set or removed.
- `newValue` – the value that the key was set to or `null` if the key was removed.
- `oldValue` – the value before the key was set or removed.

To listen for the `storage` event, you use the `addEventListener()` method of the `window` object like this:

```
addEventListener('storage', function(e){
   console.log(`The value of the ${e.key} changed for the ${e.domain}.`);
});
```

Summary

- The `Storage` type provides you with the methods for storing and managing data in web browsers.
- The `localStorage` is an instance of the `Storage` type that allows you to store persistent data in the web browsers.
- The `localStorage` can store only strings. To store objects, you convert them to strings using the `JSON.stringify()` method. And you convert the strings into objects when you

retrieve them from the `localStorage` using the `JSON.parse()` method.