Using Web Storage To Store Data In The Browser

Like cookies, the Web Storage APIallows browsers to store data in the form of key-value pairs. Web Storage has a much larger storage limit than cookies, making it a useful place to store data on the client side.

In the cookies reading, we reviewed the two main mechanisms of Web Storage:`sessionStorage`and `localStorage`. While `sessionStorage`persists for the duration of the session and ends when a user closes the browser, `localStorage`persists past the current session and has no expiration date.

One typical use case for local storage is caching data fetched from a server on the client side. Instead of making multiple network requests to the server to retrieve data, which takes time and might slow page load, we can fetch the data once and store that data in local storage. Then, our website could read the persisting data stored in localStorage, meaning our website wouldn't have to depend on our server's response - even if the user closes their browser!

In this reading, we'll go over how to store and read a key-value pair in local storage.

Storing data in local storage

Web Storage exists in the window as an object, and we can access it by usingWindow.localStorage. As we previously reviewed, with window properties we can omit the *"window"*part and simply use the property name, `localStorage`.

We can set a key-value pair in local storage with a single line of code. Here are a few examples:

```
localStorage.setItem('eatz', 'I <3 falafel');
localStorage.setItem('coffee', 'black');
localStorage.setItem('doughnuts', '["glazed", "chocolate", "blueberry",
"cream-filled"]');
```

The code above calls the `setItem()`method on the Storage object and sets a key-value pair. Examples: `eatz`(key) and `I <3 falafel`(value), `coffee`(key) and `black`(value), and `doughnut`(key) and `["glazed", "chocolate", "blueberry", "cream-filled"]`(value). Both the key and the value must be strings.

Reading data in local storage

If we wanted to retrieve a key-value pair from local storage, we could use `getItem()`with a key to find the corresponding value. See the example below:

```
localStorage.setItem('eatz', 'I <3 falafel');
localStorage.setItem('coffee', 'black');
localStorage.setItem('doughnuts', '["glazed", "chocolate", "blueberry",
"cream-filled"]');

const eatz = localStorage.getItem('eatz');
const coffee = localStorage.getItem('coffee');
const doughnuts = localStorage.getItem('doughnuts');

console.log(eatz); // 'I <3 falafel'
console.log(coffee); // 'black'
console.log(doughnuts); // '["glazed", "chocolate", "blueberry", "cream-filled"]
```

The above code reads the item with a key of `eatz`, the item with a key of`doughnut`, and the item with a key of `coffee`. We stored these in variables for handy use in any function we write.

Check the MDN docs on localStoragefor other methods on the Storage object to remove and clear all key-value pairs.

JSON and local storage

When we store and read data in local storage, we're actually storing JSONobjects. JSON is text format that is independent from JavaScript but also resembles JavaScript object literal syntax. It's important to note that JSON exists as a *string*.

Websites commonly get JSON back from a server request in the form of a text file with a `.json`extension and a MIME type of `application/json`. We can use JavaScript to parse a JSON response in order to work with it as a regular JavaScript object.

Let's look at the `doughnuts`example from above:

```
localStorage.setItem('doughnuts', '["glazed", "chocolate", "blueberry",
"cream-filled"]');
const doughnuts = localStorage.getItem('doughnuts');
console.log(doughnuts + " is a " + typeof doughnuts);
// prints '["glazed", "chocolate", "blueberry", "cream-filled"] is a string'
```

If we ran the code above in the browser console, we'd see that `doughnuts`is a string value because it's a JSON value. However, we want to be able to store`doughnuts`as an *array*, in order to iterate through it or map it or any other nifty things we can do to arrays.

We can construct a JavaScript value or object from JSON by parsing it:

```
const doughnuts = JSON.parse(localStorage.getItem('doughnuts'));
```

We used JSON.parse()to parse the string into JavaScript. If we printed the parsed value of `doughnuts`to the console, we'd see it's a plain ol' JavaScript array!

See the MDN doc on Working with JSON for more detail about using JSON and JavaScript.

What you learned:

- Why we use local storage
- How to store data in local storage
- How to read data in local storage
- How storage objects are JSON that we need to parse