LocalStorage

Download Demo Code <../localstorage-demo.zip>

Goals

- Utilize localStorage to save information in the browser
- Compare and contrast localStorage and sessionStorage
- Add and remove primitives to/from localStorage
- Add and remove objects to/from localStorage

localStorage

What is localStorage?

localStorage is a mechanism for storing information in the browser for a specific domain

The API is quite easy to use and very minimal - so let's get started with it!

localStorage vs sessionStorage

When you read more about localStorage you will also hear about something called sessionStorage

- data stored in localStorage has no expiration time
- data stored in sessionStorage gets cleared when the browsing session ends

modifying localStorage

The most important thing to remember is that all of your keys in localStorage or sessionStorage must be **strings**.

localStorage stores everything as strings, so it's just good to get in the habit of setting your keys as strings to avoid confusion.

setting an item in localStorage

We do this using the ***setItem*** method

```
localStorage.setItem("firstName", "Colt");
localStorage.setItem("favoriteNumber", 33);
localStorage.setItem("hasChickens", true);
```

retrieving an item in localStorage

To retrieve we use the ***getItem*** method (only passing in the key)

```
localStorage.getItem("firstName"); // "Colt"
```

You can alternatively just access items on the localStorage object:

```
localStorage.firstName // "Colt"
```

If you refresh the page - you should be able to still retrieve these keys in localStorage!

Clearing localStorage

To delete a key we use the removeItem function

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

To clear everything from localStorage we use the clear function

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

Adding Objects to localStorage

So far we have only added primitives, which is nice and easy, but what happens when we start adding objects?

Well, things get a bit trickier…. Let's see what happens:

```
const friends = ["Lana", "Hayden", "Jessie"];

localStorage.setItem("friends", friends);
localStorage.getItem("friends");
```

When we get the friends key from localStorage, we don't have an array anymore - we have a string!

Remember, when dealing with localStorage, everything gets converted into a string

In order to get back our original data type, we need to convert this array to a special string format called JSON

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate

We will be discussing it quite a bit more later on

Working with JSON in the browser

JavaScript has a built-in JSON object, and on this object are two methods used to convert JavaScript data into JSON, and to parse JSON strings into JavaScript

- JSON.stringify - is used to convert JavaScript to JSON (or stringify)
- JSON.parse parses a string as JSON

Using these two methods allows us to preserve the intended data structure when it is something other than a string:

```
const friends = ["Lana", "Hayden", "Jessie"];

// the JSON.stringify function
// converts the friends array into a JSON string

localStorage.setItem("friends", JSON.stringify(friends));

// JSON.parse converts the JSON string
// back into JavaScript (in this case, a valid array)

JSON.parse(localStorage.getItem("friends"));
```

Recap

- localStorage is useful for storing information in the browser
- to store objects, use JSON.stringify when setting and JSON.parse when retrieving
- if you just want to store information for the time a tab is open, use sessionStorage