JS localStorage & JSON

Storing client-side data

Assembler
Institute of Technology

Contents

What is JSON?

JSON

JSON stands for JavaScript Object Notation (JSON).

It is a syntax for serializing objects, arrays, numbers, strings, booleans, and null.

JSON is often used when data is sent from a server to a web page.

JSON Syntax

JSON looks similar to JavaScript's way of writing arrays and objects. Data is also in name/value pairs and is separated by commas. There are a few restrictions.

- All property names have to be surrounded by double quotes
- Only simple data expressions are allowed
- Function calls, bindings, or anything that involves actual computation can´t be used
- Comments are not allowed in JSON.

```
[
    {
        "firstname": "Mark",
        "lastname": "Gulman",
        "age": 34
    },
    {
        "firstname": "Ana",
        "lastname": "Spence",
        "age": 22
    },
    {
        "firstname": "John",
        "lastname": "Hammer",
        "age": 31
    }
]
```

JSON Data Types

The JSON format only allows basic data types such as:

- Numbers
- Strings
- Booleans
- Arrays of other basic types
- Objects with keys and values as strings
- null values

```
{
    "firstName": "John",
    "isAlive": true,
    "age": 27,
    "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "postalCode": "10021-3100"
    },
    "phoneNumbers": [
        {
            "type": "home",
            "number": "212 555-1234"
        }
    ],
    "children": [],
    "spouse": null
}
```

The main JSON methods in JavaScript

JSON.stringify()

The JSON.stringify() method converts a JavaScript object or value to a JSON string.

JSON.parse()

The JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string.

JSON.stringify()

The JSON.stringify() method converts a JavaScript object or value to a JSON string.

- Boolean, Number, and String objects are converted to the corresponding primitive values
- undefined or Functions, are not valid JSON values.
- Dates are converted to strings using the toString() method for each data type

NOTE: As you can see in the JSON string both the properties and values are between quotes.

```
 // JS Object
let person = {
   firstname: "Mark",
   lastname: "Gulman",
   age: 34,
   date: new Date(),
};

let json = JSON.stringify(person);

console.log(json);
//{"firstname":"Mark","lastname":"Gulman","age":34,"dat
e":"2020-05-05T12:16:00.590Z"}
```

JSON.stringify() parameters

The JSON.stringify method accepts as parameters a value to convert to JSON and two optional parameters. The first optional parameter can be a replacer function or an array. The second is the space units.

```
JSON.stringify(value, replacerFunction, spaceUnit);
```

JSON.stringify() parameters

If we provide **null** as the replacer function parameter, all the values will be included in the final JSON output.

If we provide a number as the **space** value, it indicates the number of space characters to use as white space; this number is capped at 10 (if it is greater, the value is just 10).

Values less than 1 indicate that no space should be used.

JSON.stringify() to format JSON data for better debugging

We can use the JSON.stringify(data, null, 2) providing null as the replacer function and 2 for the spaces value to format JSON data for better debugging to, for example, output the JSON to the console or to the DOM to visualize the JSON data.

```
let person = { name: "Mark", age: 34 };

let json = JSON.stringify(person);
console.log(json);
// {"name":"Mark","age":34}

let formatted = JSON.stringify(person, null, 2);
console.log(formatted);
// {
//   "name": "Mark",
//   "age": 34
// }
```

Using JSON.stringify(data, replacer) to filter out items

We can use the JSON.stringify(foo, replacer) to filter out items from the final JSON string.

```
function replacer(key, value) {
   // Filtering out properties
   if (typeof value === "string") {
       return undefined;
   }
   return value;
}

var foo = {
   foundation: "Car maker",
   week: 45,
   transport: "car",
   month: 7,
};

console.log(JSON.stringify(foo, replacer));
// '{"week":45,"month":7}'
```

JSON.parse()

The JSON.parse() method parses a JSON string, constructing the JavaScript value or object described by the string.

- JSON.parse() does not allow trailing commas
- JSON.parse() does not allow single quotes

```
let person = JSON.parse(
   '{"firstname":"Mark","lastname":"Gulman","age":34}'
);

console.log(person.firstname); // "Mark",

// JSON.parse() does not allow trailing commas
// both will throw a SyntaxError
JSON.parse("[1, 2, 3, 4, ]");
JSON.parse('{"foo" : 1, }');

// JSON.parse() does not allow single quotes
// will throw a SyntaxError
JSON.parse("{'foo': 1}");
```

JSON Validator

https://jsonlint.com/

I need to work with data

As you realised in the previous exercises, in which you needed to store data like rankings, once the page was reloaded the data was lost.

To avoid the loss of information, and to be able to persist data, there are a few ways to store information in the client-side.

What is localStorage?

Storing data client-side

Storing data on the client-side

Assembler
Institute of Technology

I need to work with data

As you probably realised, once a page is reloaded the data is lost.

To avoid the loss of information, and to be able to persist data, there are a few ways to store information in the client-side.

What is frontend?

In software engineering, frontend software is the presentational layer which provides a user interface to an application.

It is made up of technologies that users can see and interact with.

Examples of frontend technologies are:

- HTML
- CSS
- Browser JavaScript
- Frontend libraries such as React, Vue or Angular

What is backend?

A backend software is the part of the application that traditionally users cannot interact with. The backend often performs the more complex operations that the client application should not carry on with. Some of this are:

- storing data in databases
- handling the communication with the frontend client
- sending and receiving data
- handling user authentication
- etc

It is usually a server written in languages such as JS, Node.js, PHP, Python, Ruby, or MySQL, MongoDB or Compiled languages such as C#, Java and Go.

The read-only **localStorage** property **is immutable** and allows to access an object stored for the Document's origin.

The stored data is saved across browser sessions.

The data is saved locally only and can’t be read by the server. **This eliminates the security issue** that cookies have present. Everything is done using JavaScript.

Differences between localStorage and sessionStorage

While data in localStorage doesn't expire, data in sessionStorage is cleared when the session ends.

- A session lasts for as long as the browser is open, and it is not affected if the page reloads and is restored.
- Opening a page in a new tab or window creates a new session with the value of the top-level browsing context.
- Opening multiple tabs/windows with the same URL creates a sessionStorage for each tab/window.
- Closing a window ends the session and clears the generated objects in sessionStorage.

localStorage

Is an object that can be used to store data in a way that survives reloading a page.

This object allows you to file string values under names. This stores the information on the browser.

```
// Access to the local storage API
var myStorage = window.localStorage;
```

What is an API?

An application programming interface (API) is a way for computers and applications to communicate between each other and share information and services.

In the browser, we can use the Window.sessionStorage and Window.localStorage API’s to store and read data.

Implementation

The following snippet has examples of accessing and modifying the current domain's local storage object.

```
let myStorage = window.localStorage;

// Adds a data item to localStorage
myStorage.setItem('myCat', 'Tom');

// Get an item in localStorage
myStorage.getItem('myCat');

// Removing the localStorage item
let cat = myStorage.removeItem('myCat');

// Removing all the localStorage items
myStorage.clear();
```

Three ways to set an entry

There are three ways to set a entry into the localStorage object.

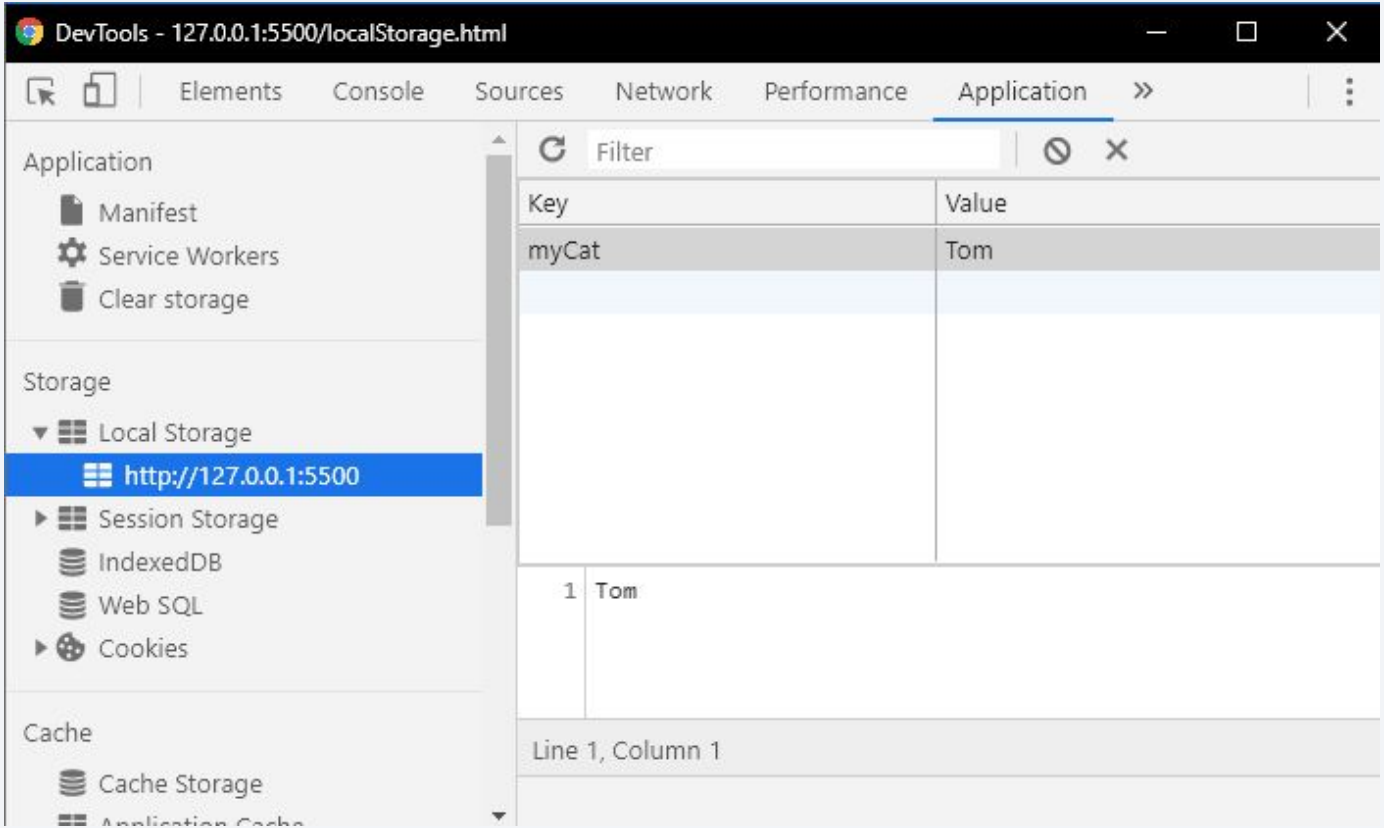
```
localStorage.myCat = 'Tom';

localStorage['myCat'] = 'Tom';

localStorage.setItem('myCat', 'Tom');
```

Limitations of localStorage and sessionStorage

- Both localStorage and sessionStorage have a limit of about 5MB to store data
- They can only be accessed from the client
- Servers cannot read data in localStorage
- Both localStorage and sessionStorage cannot have expired dates and they need to be cleared/removed with JavaScript or by clearing the browser’s cache


DevTools - 127.0.0.1:5500/localStorage.html
Elements
Console
Sources
Network
Performance
Application
Application
Manifest
Service Workers
Clear storage
Storage
Local Storage
http://127.0.0.1:5500
Session Storage
IndexedDB
Web SQL
Cookies
Cache
Cache Storage
Filter
Key
Value
myCat
Tom
1
Tom
Line 1, Column 1

WEB STORAGE

https://mdn.github.io/dom-examples/web-storage/

Questions?

Assembler
Institute of Technology