Session 19

Mohamed Emary

May 31, 2024

1 Local Storage

Local storage is a way to store data in the browser (client-side storage). It is a key-value pair storage limited storage (5MB).

When we say it's Local Storage, it means it's local to the browser. It is not stored on the server. It is stored on the client's machine, so it is not shared with other users.

To see the local storage in the browser, open the developer tools and go to the Application tab. Then, click on Local Storage.

You can *only store strings* in the local storage.

To store a value in the local storage, you can use the `setItem()` method. The `setItem()` method takes two parameters: the key and the value.

```
localStorage.setItem('name', 'Mohamed');
```

Keys are unique. If you set a value to a key that already exists, it will overwrite the old value.

```
localStorage.setItem('name', 'Ahmed');
```

Now the value of the key `name` is `Ahmed`.

To get a value from the local storage, you can use the `getItem()` method. The `getItem()` method takes one parameter: the key.

```
var name = localStorage.getItem('name');
console.log(name); // Ahmed
```

To remove a value from the local storage, you can use the `removeItem()` method. The `removeItem()` method takes one parameter: the key.

```
localStorage.removeItem('name');
var name = localStorage.getItem('name');
console.log(name); // null
```

To know how many items are stored in the local storage, you can use the `length` property.

To clear the local storage, you can use the `clear()` method. The `clear()` method takes no parameters.

```
localStorage.setItem('name', 'Mohamed');
localStorage.setItem('age', '25');
console.log(localStorage.length); // 2
localStorage.clear();
var name = localStorage.getItem('name');
var age = localStorage.getItem('age');
console.log(name); // null
console.log(age);  // null
```

To know which key at a specific index, you can use the `key()` method. The `key()` method takes one parameter: the index.

```
localStorage.setItem('name', 'Mohamed');
localStorage.setItem('age', '25');
console.log(localStorage.key(0)); // name
console.log(localStorage.key(1)); // age
```

You shouldn't store sensitive data in the local storage because it's not secure. It's accessible by anyone who has access to the client's machine.

We don't get all data from backend some data that are not sensitive like language. can be stored in the local storage.

Local storage data are not removed even if you close the browser. It will be removed when you clear the local storage or when you delete the browser's data.

2 Session Storage

Session storage is similar to local storage, but it's for the session only which means it's removed when the session is ended like when you close the tab or the browser.

We have a method called `sessionStorage` that works the same as `localStorage` with the same methods and properties like:

- `setItem()`
- `getItem()`
- `removeItem()`
- `length`
- `clear()`
- `key()`

3 Storing Objects

As we mentioned before, you can only store strings in the local storage. If you want to store an object, you need to convert it to a string using `JSON.stringify()`.

```
var person = {
  name: 'Mohamed',
  age: 25
};

localStorage.setItem('person', JSON.stringify(person));
```

To get the object from the local storage, you need to parse the string using `JSON.parse()`.

```
var person = JSON.parse(localStorage.getItem('person'));
console.log(person.name); // Mohamed
console.log(person.age);  // 25
```

The same can be done with arrays:

```
var people = [
  { name: 'Mohamed', age: 25 },
  { name: 'Ahmed', age: 30 },
  { name: 'Ali', age: 35 }
];

localStorage.setItem('people', JSON.stringify(people));
var people = JSON.parse(localStorage.getItem('people'));
console.log(people[1]); // { name: 'Ahmed', age: 30 }
```

4 Accepting Image As Input

With the input element where the user can select an image, you will specify the type as `file` you can also specify the `accept` attribute to specify the type of files that the user can select, for example, `image/png`, `image/jpeg`, or `image/*` to accept all image types, and you can also use the attribute `multiple` to allow the user to select multiple files.

```
<input type="file" accept="image/*" id="imgInput" />
<button id="upload">Upload</button>
```

This will create an input field that accepts all image types.

In your JavaScript code when you `console.log` the input element value, you will get a `C:\fakepath\` followed by the image file name, so for example if your image file name is `my_image.jpg` the console output will be `C:\fakepath\my_image.jpg`

```
var imgInput = document.getElementById('imgInput');
var upload = document.getElementById('upload');
upload.onclick = function() {
  console.log(imgInput.value); // C:\fakepath\my_image.jpg
};
```

This `C:\fakepath\` is a browser standard that doesn't depend on the operating system and it's used by the browser with any file the user uploads not just images. This is done for security reasons to prevent the website from knowing the user's file system structure.

If the real path was `C:\Users\Mohamed\TopSuperSecretProject\VeryImportantImg.png`, then by uploading it you'd be exposing that your real name is Mohamed and you're working on TopSuperSecretProject which is a security risk.

Since `C:\fakepath\` is a browser standard, you can see it in any operating system even those with no `C:\` partition like macOS or Linux.

So how can you display the image?

You can get the file object from the input element using the `files` property. The `imgInput.files` is a FileList object that contains the files the user selected in case the input element has the `multiple` attribute. If the input element doesn't have the `multiple` attribute, then you can access the file using `imgInput.files[0]`.

You can access the file name using `name` property.

We get the file object from the input element, then we use the `createObjectURL()` method to create a URL for the file object, then we can use that URL to display the image in the browser using the `src` attribute of an image element.

In HTML:

```
<input type="file" accept="image/*" id="imgInput" />
<button id="upload">Upload</button>
<img id="img" />
```

In JavaScript:

```
var imgInput = document.getElementById('imgInput');
var upload = document.getElementById('upload');
var img = document.getElementById('img');
upload.onclick = function() {
  var file = imgInput.files[0];
  if (file) {
    var objectURL = URL.createObjectURL(file);
    // set the src attribute of the image element to the object URL
    img.src = objectURL;
  }
};
```

This is how the page will look like:

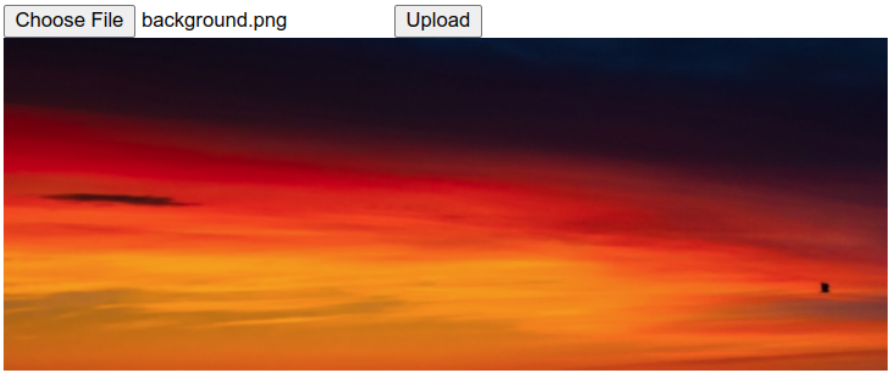

Figure 1: Image Upload