Day - 14 : Local Storage

What is meant by local storage ?

> What do the words local storage mean ?

Storing locally, right? Storing what ? ⇒ data or information. Storing where? ⇒ Locally ⇒ means where? On google drive? No. it'll be stored on our browser where we will run the app. Local storage can be used to store data as a mini database or a local database.

Now, let's see how the data will be stored in the local storage?

> It is stored in the form of key-value pairs but it is not an object because we cannot apply all object methods here.

Actual Definition:

localStorage is a property that allows JavaScript sites and apps to save key-value pairs in a web browser with no expiration date.

When to use local storage ?

- ***You should only use local storage when storing insensitive information, i.e., we cannot store passwords and some sensitive information***
- ***Local storage can help in storing data temporarily before it is pushed to the server, i.e., we can't store data permanantly.***

What are the limitations ?

The major limitations of local storage are:

- ***Insecure data (data is not secured, it can be hacked, i.e., someone who can access your system can have the data)***
- ***Synchronous operations (We'll learn about it later)***
- ***Limited storage capacity, i.e., images and videos cannot be stored, but we can store image and video urls.***

Where to see the local storage ?

- ***Open browser and click on inspect.***
- ***Click*** `Application` ***and inside that you cna see*** `Local Storage`
- ***That is where our data gets stored, we'll see how to store and data and come back again here.***

How does local storage work ?

To use localStorage in your web applications, there are four methods to choose from:

- ***setItem()****: Add key and value to local storage*
- ***getItem():*** *This is how you get items from localStorage*
- ***removeItem()****: Remove an item by key from localStorage*
- ***clear():*** *Clear all local storage*

If I want to store something on local storage, we've to use setItem.

If I want to access something present in the local storage, we've to use getItem

If I want to remove some data from local storage, we've to use removeItem

If I want to clear all local storage, we've to use clear

Note : Local storage stores only strings, booleans and numbers.

What are the other data types we have ?

Arrays, Objects cannot be stored.

Set Item

Syntax

```
setItem("keyName",value)
value can be a string, boolean or a number
```

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Document</title>
</head>
<body>

</body>
</html>

<script>
        Write only one first and then write others to show that
        will get replaced.
    localStorage.setItem("studentName1","Cherry")
        localStorage.setItem("studentName2","Superman")
        localStorage.setItem("studentName2","Spiderman")
```

```
        localStorage.setItem("studentId",1)
        localStorage.setItem("studentPresent", true);
</script>
```

`studentName1` is key and `Cherry` is value.

Get Item

> ***In objects, if we want to get the value of any field, how will we do it?***

For eg:

```
var student = {
    name:"Cherry
}

console.log(student.name)
```

So, if we want to access the values in an object, we write the key only. Similarly, for local storage also, we'll write getItem("keyName").

Syntax

```
getItem("keyName")
```

```
<script>
    var name = localStorage.getItem("studentName1")
    console.log(name)
    var p = document.createElement("p")
    p.innerText = name
    document.querySelector("body").append(p)
</script>
```

> ***Whenever we try to access a key which is not present in local storage, it'll return null.***

Remember the difference between null and undefined?

If I say how many mangoes does an apple tree give? Null, not undefined, right.

Similarly, if we try to access something which is not present in local storage, it'll give us null.

Limitations about storing arrays

```
<script>
    let nums = [1,2,3,4]
    localStorage.setItem("arrOfNums",nums)
</script>
```

Open browser and show how it stores.

```
1,2,3,4,5
```

It just stores with a comma, it does not store an array. So let's see in which datatype is it stored. How can we check what datatype it is?

```
var getArr = localStorage.getItem("arrOfNums")
console.log(typeof getArr) // will return string
```

Show in console and it is storing in the form of "1,2,3,4,5". But that is not what we want

So, how do we store arrays and objects

If we store something like this “[1,2,3,4,5]”, even though it is a string, we can convert it into an array and use it, right?

We need to use JSON in order to store arrays and objects in local storage.

What is JSON ?

Javascript Object Notation

What JSON will do is it will convert anything into string using its methods and since we can store strings as values in local storage, we can now store stringified arrays and stringified objects on local storage.

Actual Definition :

JavaScript Object Notation (JSON) is a representation of structured data based on JavaScript object syntax.

JSON is most widely used

- ***Data is sent and received on Internet in JSON (mostly).***
- ***It is based on Javascript objects.***
- ***used with network requests (AJAX etc.)***

Difference between JSON and objects ?

Let’s first write data in JSON format and object and compare the difference.

```
var studentObj = {
    name:"Cherry",
    gender: "Male",
}

var studentJSON = {
    "name": "Cherry",
```

```
    "gender": "Male",
}
```

In JSON, both keys and values should be in the form of a string but in objects, the values must be in the form of a string, not keys.

So, we've to convert our array to string in order to store it in our local storage. There is a method called JSON.stringify to do it.

```
var stringArr = JSON.stringify(nums)
console.log(stringArr)
localStorage.setItem("arrOfNums", stringArr)
```

> ***Now open local storage and show how the values are being stored.***

Now, we have seen how to store in local storage, now if we want to access the local storage item, we've to use getItem method.

```
var arr = localStorage.getItem("arrOfNums")
console.log(typeof arr,arr) will give us a string format of the
Now can we do any array methods like push, pop or for loops in t
So, we've to convert this string format back to array. The metho
is parse

arr = JSON.parse(arr)
console.log(typeof arr,arr)
This will give us the array and we can use array methods here ar
```

Similar is the case with objects. Let's see how to store objects in local storage now.

```
var studentObj = {name:"Cherry", place:"Bangalore"}
localStorage.setItem("studenInfo",JSON.stringify(studentObj))

var getStudentObj = JSON.parse(localStorage.getItem("studentInf
```

Let's see the following diagram.

`Client` is user. So, whenever we're storing on web server or local storage, we've to `stringify` the data and whenever we get the data from the local storage, we need to `parse` the data in order to modify it further or apply the methods that the datatype has.

Why do we need local storage with example ?

Let's write a small example and see how local storage is helpful.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Image</title>
  </head>
  <body>
    <img
      height="200px"
      src="https://cdn.theatlantic.com/media/img/photo/2019/0
3/national-puppy-day-photos/p01_556211362-1/original.jpg"
      alt=""
    />
    <div>
      <p>Likes : <span id="likes"></span></p>
      <p>Dislikes : <span id="dislikes"></span></p>
    </div>
    <div>
      <button id="likeBtn">Like👍</button>
      <button id="dislikeBtn">Dislike👎</button>
```

```
    </div>
  </body>
</html>
```

Let's write the logic in script tag since it's a small file.

```
document.getElementById("likeBtn").addEventListener("click", lil
  var likeCount = 0;
  document.querySelector("#likes").innerText = likeCount; // ini

  var dislikeCount = 0;
  document.querySelector("#dislikes").innerText = dislikeCount;

  function likeFun() {
    likeCount++;
    document.querySelector("#likes").innerText = likeCount;
  }

  document.getElementById("dislikeBtn").addEventListener("click'

  function dislikeFun() {
    dislikeCount++;
    document.querySelector("#dislikes").innerText = dislikeCount
  }
```

Now, we can see if I click on like, the likes count increases and if I click on dislike button, the dislikes count increases, right. Now if I refresh the page, what will happen ?

What lines of code will be executed inside the script tag ?

But is this the desired behaviour ?

How do like and dislike button in facebook work?

Let's say you liked an image and you click on like. Now, you've logged out of it or closed the fb, did some work and after sometime or few days, opened fb again.

Now tell me the image you've liked, will the like be present or should you like it again?

So, how can we retain the no of likes and dislikes?

By using local storage.

```
  function likeFun() {
    likeCount++;
    document.querySelector("#likes").innerText = likeCount;

        //added line
    localStorage.setItem("dogLike", likeCount);
  }

  document.getElementById("dislikeBtn").addEventListener("cli
ck", dislikeFun);

  function dislikeFun() {
    dislikeCount++;
    document.querySelector("#dislikes").innerText = dislikeCo
unt;

        //added line
    localStorage.setItem("dogDislike", dislikeCount);
  }
```

Now we are setting the no of likes and dislikes, now let's see the behaviour of the like and dislike buttons.

> ***Open web browser local storage and show the value of likes and dislikes.***

Why is it working like this ?

Because we are setting the key but we are not using it or accessing it. So, what should we do ?

```
document.getElementById("likeBtn").addEventListener("click",
likeFun);
//instead of initial value as 0, we should get item from LS
  var likeCount = localStorage.getItem("dogLike");
  document.querySelector("#likes").innerText = likeCount;

//instead of initial value as 0, we should get item from LS
  var dislikeCount = localStorage.getItem("dogDislike");
  document.querySelector("#dislikes").innerText = dislikeCoun
t;
```

Now is it working fine ?

But there is a bug in this code also ? Can anyone tell the bug?

> ***Delete all the keys and start the application again. Now show the bug, initially when there are no keys stored in local storage.***

How to solve this ?

using if condition

```
var likeCount;
  if (localStorage.getItem("dogLike") == null) {
    likeCount = 0;
  } else {
    likeCount = localStorage.getItem("dogLike");
  }

// above if is same as
```

```
console.log(true||false);

 console.log(false||true);

  console.log(null||0)

  console.log(1||null)

  console.log(localStorage.getItem("dadsadadsdda")||0)


------------------

var likeCount = localStorage.getItem("dogLike")||0
```

```
  var likeCount = localStorage.getItem("dogLike") || 0
    var dislikeCount = localStorage.getItem("dogDislike") || 0
```

So, if the keys are not there, then the count will start from 0.

We've created a todo app before evaluation, remember? Let's try to store that data in local storage. Why should we store the todo data in local storage ?

Because what is meant by a todo list? It is a list we write to remember that we've to do certain tasks. So, is it a todo app if we've to write the tasks everytime we open the app?

That is why we'll work on the todo app to store the data in local storage.

Todo List

Previous day solution

```
document.querySelector("form").addEventListener("submit", myT
odo);

function myTodo() {
  event.preventDefault();
  var task = document.querySelector("#task").value;
  var priority = document.querySelector("#priority").value;

  var tr = document.createElement("tr");
  var td1 = document.createElement("td");
  td1.innerText = task;

  var td2 = document.createElement("td");
  td2.innerText = priority;

  if (priority == "High") {
    td2.style.backgroundColor = "red";
  } else {
    td2.style.backgroundColor = "green";
  }

  var td3 = document.createElement("td");
  td3.innerText = "Delete";
  td3.addEventListener("click", deleteRow);
  td3.style.color = "red";
  tr.append(td1, td2, td3);
  document.querySelector("tbody").append(tr);
}

function deleteRow() {
```

```
  event.target.parentNode.remove();
    }
```

```
document.querySelector("form").addEventListener("submit", cre
ateTodo);

var todoArr = JSON.parse(localStorage.getItem("todoList")) ||
[];

displayTable(todoArr); // ignore this line at first

window.addEventListener("load",function(){
        displayTable(todoArr);
}). // explain this event

function createTodo() {
  event.preventDefault();
  // var task = document.querySelector("#taskList").value;
  // var priority = document.querySelector("#priority").valu
e;
    Now, we've to store it in local storage, so what kind of
datatype should
    we store this todo task ?
    object => why ? => easier to access the task and priority
and anyone
    who works on the todo app will know what is being stored,
right?
    So, lets create a todoObj as follows:
  var todoObj = {
    task: document.querySelector("#taskList").value,
    priority: document.querySelector("#priority").value,
  };
    Now, we've created an object, but it is only for one task
```

```
right? So we've
    to push this object into todoArr.
  todoArr.push(todoObj);
  localStorage.setItem("todoList", JSON.stringify(todoArr));
    Now, let's separate the displaying of todos in the table
to a different
    function.
  displayTable(todoArr);
}

function displayTable(data) {
      document.querySelector("tbody").innerHTML = ""; // 1st
do not write this
        //also now
  data.map(function (elem) {
    var tr = document.createElement("tr");
    var td1 = document.createElement("td");
    td1.innerText = elem.task;

    var td2 = document.createElement("td");
    td2.innerText = elem.priority;

    if (elem.priority == "High") {
      td2.style.backgroundColor = "red";
    } else {
      td2.style.backgroundColor = "green";
    }

    var td3 = document.createElement("td");
    td3.innerText = "Delete";
    td3.addEventListener("click", deleteRow);
    td3.style.color = "red";
    tr.append(td1, td2, td3);
    document.querySelector("tbody").append(tr);
  });
}
```

```
function deleteRow() {
  event.target.parentNode.remove();
// tell them to try this as home work - give hint to use spli
ce method
}
//Now, add some tasks and let them observe the error. Then wr
ite 1st
```

> ***After all this, run the app once and then show the problem, if the todo app has some data, that data has to be displayed first as soon as the app runs, right?***

Why isn't it doing like this ?

So, what should we do now ? As soon as we fetch data from local storage , we've to invoke the function displayData in order to show the data if it is present in the tabular format.