Local Storage and Session Storage

Estimated time needed: 10 minutes

In React, Local Storage and Session Storage are two mechanisms that modern web browsers provide to store data on the client-side. These mechanisms allow you to store key-value pairs persistently in Local Storage or per session in Session Storage within the user's browser.

Local Storage

Local Storage allows you to store data in the browser even after the user closes the tab or the browser. The data remains available until it is explicitly cleared or until the user clears their browser's data.
Here is an example of a React component using Local Storage. In this example, we store the `count` state in the Local Storage. When the component mounts, we check if there is a stored value for `count` in the Local Storage and initialize the state with that value. We also update the value in Local Storage whenever the `count` is updated.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
import React, { useState, useEffect } from 'react';

const ExampleComponent = () => {
  const [count, setCount] = useState(0);

  useEffect(() => {
    // When the component mounts, check if there's a 'count' value in Local Storage.
    const storedCount = localStorage.getItem('count');
    if (storedCount) {
      setCount(parseInt(storedCount));
    }
```

```
  }, []);

  const incrementCount = () => {
    const newCount = count + 1;
    setCount(newCount);
    // Save the updated count value to Local Storage.
    localStorage.setItem('count', newCount.toString());
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={incrementCount}>Increment</button>
    </div>
  );
};

export default ExampleComponent;

```

Copied!

Session Storage

Session Storage, as the name suggests, stores data for the duration of a single session. The data persists across different tabs within the same browser window but is cleared when the user closes the entire browser.
Here is an example of using Session Storage in a React component. In this example, we store the `name` state in the Session Storage. When the component mounts, we check if a stored value for the `name` exists in the Session Storage. If yes, we initialize the state with that value. When the user enters a name in the input field, we update the `name` state and save it to the Session Storage.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
```

```
import React, { useState, useEffect } from 'react';

const ExampleComponent = () => {
  const [name, setName] = useState('');

  useEffect(() => {
    // When the component mounts, check if there's a 'name' value in Session Storage.
```

```
    const storedName = sessionStorage.getItem('name');
    if (storedName) {
      setName(storedName);
    }
  }, []);

  const handleInputChange = (event) => {
    const newName = event.target.value;
    setName(newName);
    // Save the entered name to Session Storage.
    sessionStorage.setItem('name', newName);
  };

  return (
    <div>
      <p>Name: {name}</p>
      <input type="text" value={name} onChange={handleInputChange} />
    </div>
  );
};

export default ExampleComponent;

```

Copied!

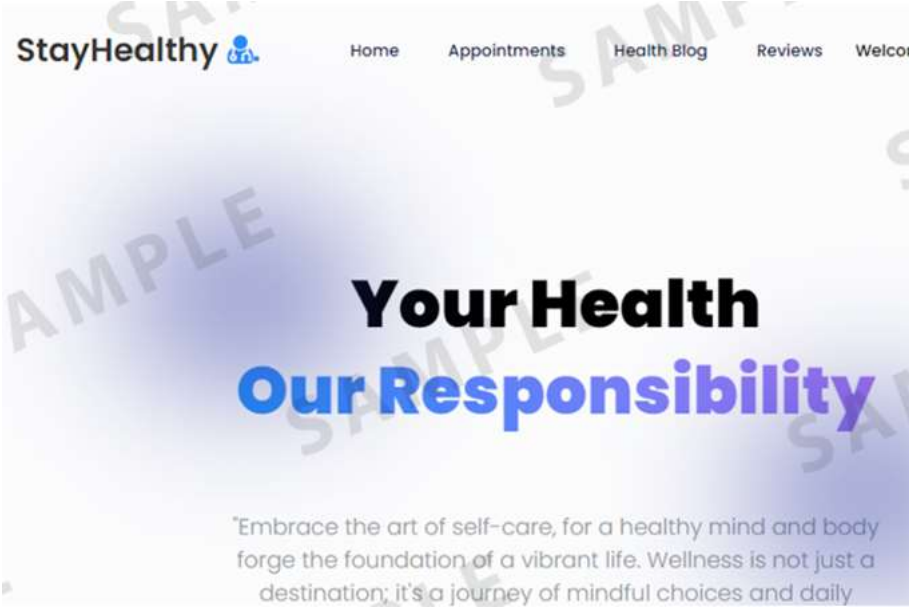
REMEMBER
Local Storage and Session Storage have limitations regarding how much data they can store and that they only support string key-value pairs. Also, they are accessible to JavaScript code running in the same domain. Therefore, be cautious about storing sensitive information in them.

Note

The Session Storage details have already been implemented in the backend code for the labs in this capstone project. When you implement the code logic of the **Sign Up** and **Login** functionalities, you will be able to view the user's name and email ID in the Session Storage or the Local Storage.

To access the Storage:

1. Right-click in the browser after signing up.
2. Then, click **Inspect**.
3. Next, click the **Application** tab.



4. Click **Session Storage**.
5. The email ID and user name details are displayed.

Author(s)

Richa Arora

Changelog

Date	Version	Changed by	Change Description
2023-08-07	0.1	Richa Arora	Initial version created

© IBM Corporation 2023. All rights reserved.