**Practical 09**

**Aim:** Web Storage and State Management: Use localStorage and sessionStorage to persist data across page reloads. Create a theme switcher or preference saver that remembers settings. give detailed lab manual Javascript, with all examples explanation and task

## 

## **Theory**

### **What is Web Storage?**

Web Storage allows web applications to store data locally within the user’s browser.  
 It provides **key–value** storage that is **faster and more secure** than cookies.

There are two main types:

| **Storage Type** | **Lifetime** | **Shared Across Tabs?** | **Typical Use** |
| --- | --- | --- | --- |
| **localStorage** | Persistent (until manually cleared) | Yes | Preferences, theme, login tokens |
| **sessionStorage** | Temporary (cleared when tab closes) | No | Session-based data, temporary settings |

### **Key Differences Between localStorage and sessionStorage**

| **Feature** | **localStorage** | **sessionStorage** |
| --- | --- | --- |
| Lifetime | Stays even after browser closes | Removed when tab or window closes |
| Scope | Shared across all tabs/windows | Specific to one tab |
| Storage Limit | ~5MB | ~5MB |
| Example Use | Saving theme or language preference | Saving form input until user submits |

## 

## 

## 

## **Web Storage API Methods**

| **Method** | **Description** | **Example** |
| --- | --- | --- |
| setItem(key, value) | Stores a value | localStorage.setItem("theme", "dark"); |
| getItem(key) | Retrieves a value | localStorage.getItem("theme"); |
| removeItem(key) | Deletes a key | localStorage.removeItem("theme"); |
| clear() | Clears all storage | localStorage.clear(); |
| key(index) | Returns key at given index | localStorage.key(0); |

## **Example 1: Basic localStorage and sessionStorage**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Web Storage Example</title>

</head>

<body>

<h2>Web Storage Demo</h2>

<input type="text" id="username" placeholder="Enter your name">

<button onclick="saveData()">Save</button>

<button onclick="loadData()">Load</button>

<button onclick="clearData()">Clear</button>

<p id="output"></p>

<script>

function saveData() {

let name = document.getElementById("username").value;

localStorage.setItem("userName", name);

sessionStorage.setItem("sessionName", name);

alert("Data saved!");

}

function loadData() {

let localName = localStorage.getItem("userName");

let sessionName = sessionStorage.getItem("sessionName");

document.getElementById("output").innerHTML =

`LocalStorage: ${localName} <br> SessionStorage: ${sessionName}`;

}

function clearData() {

localStorage.clear();

sessionStorage.clear();

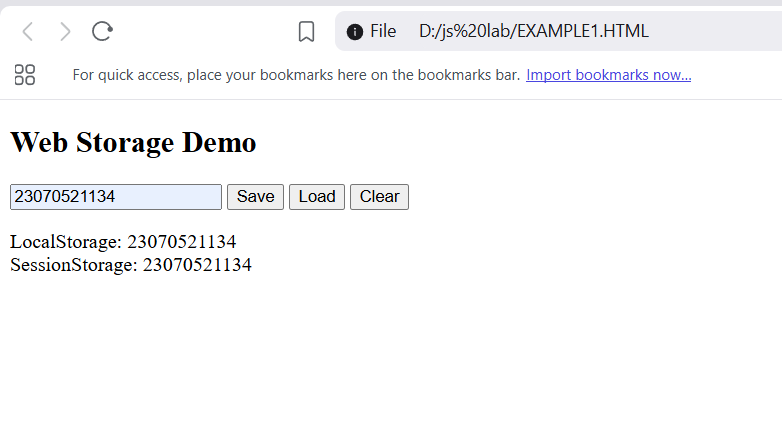
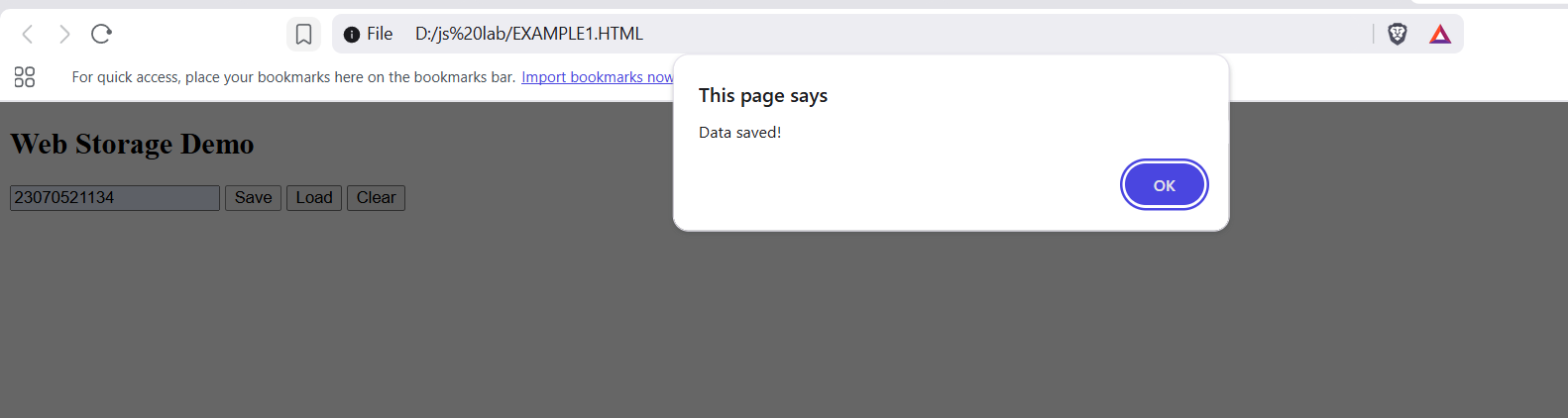
document.getElementById("output").innerHTML = "Storage cleared!";

}

</script>

</body>

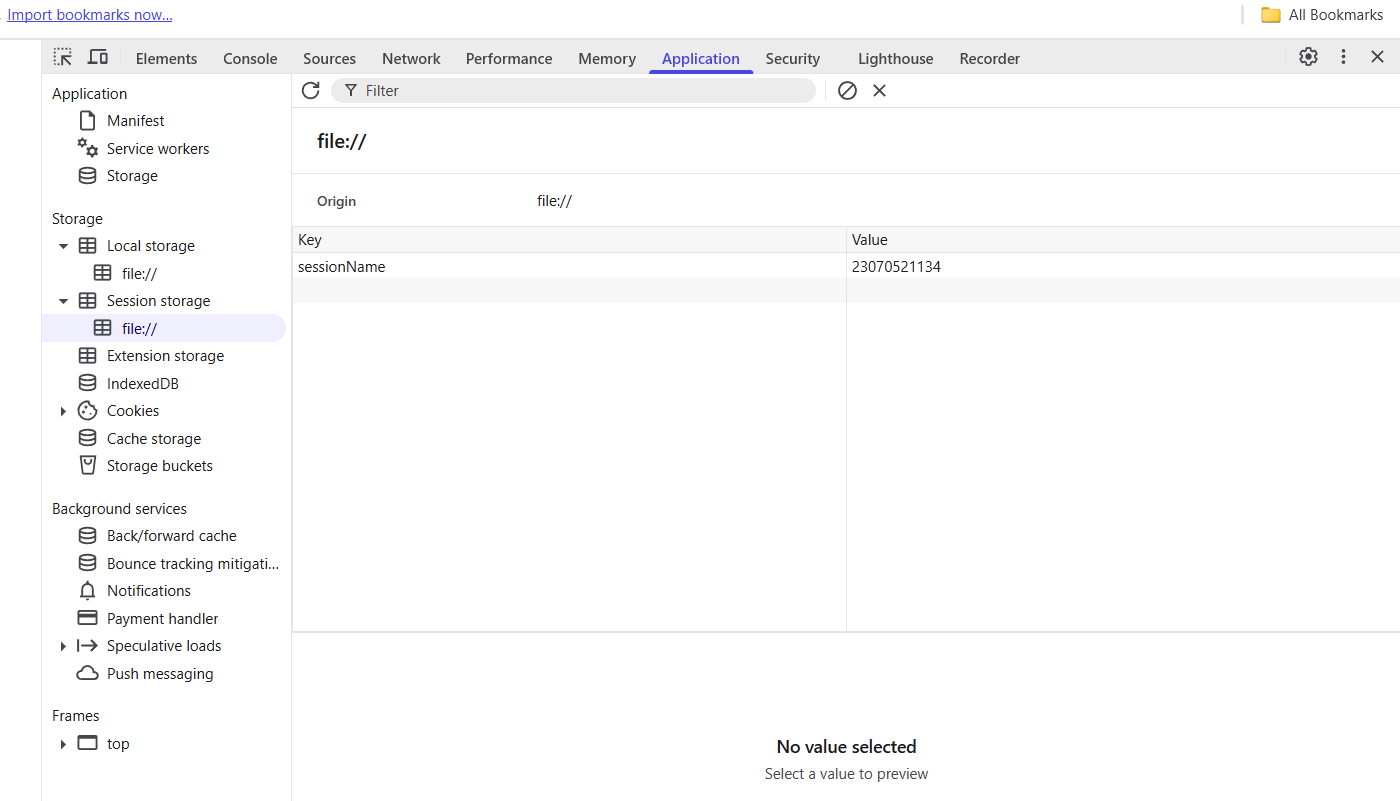
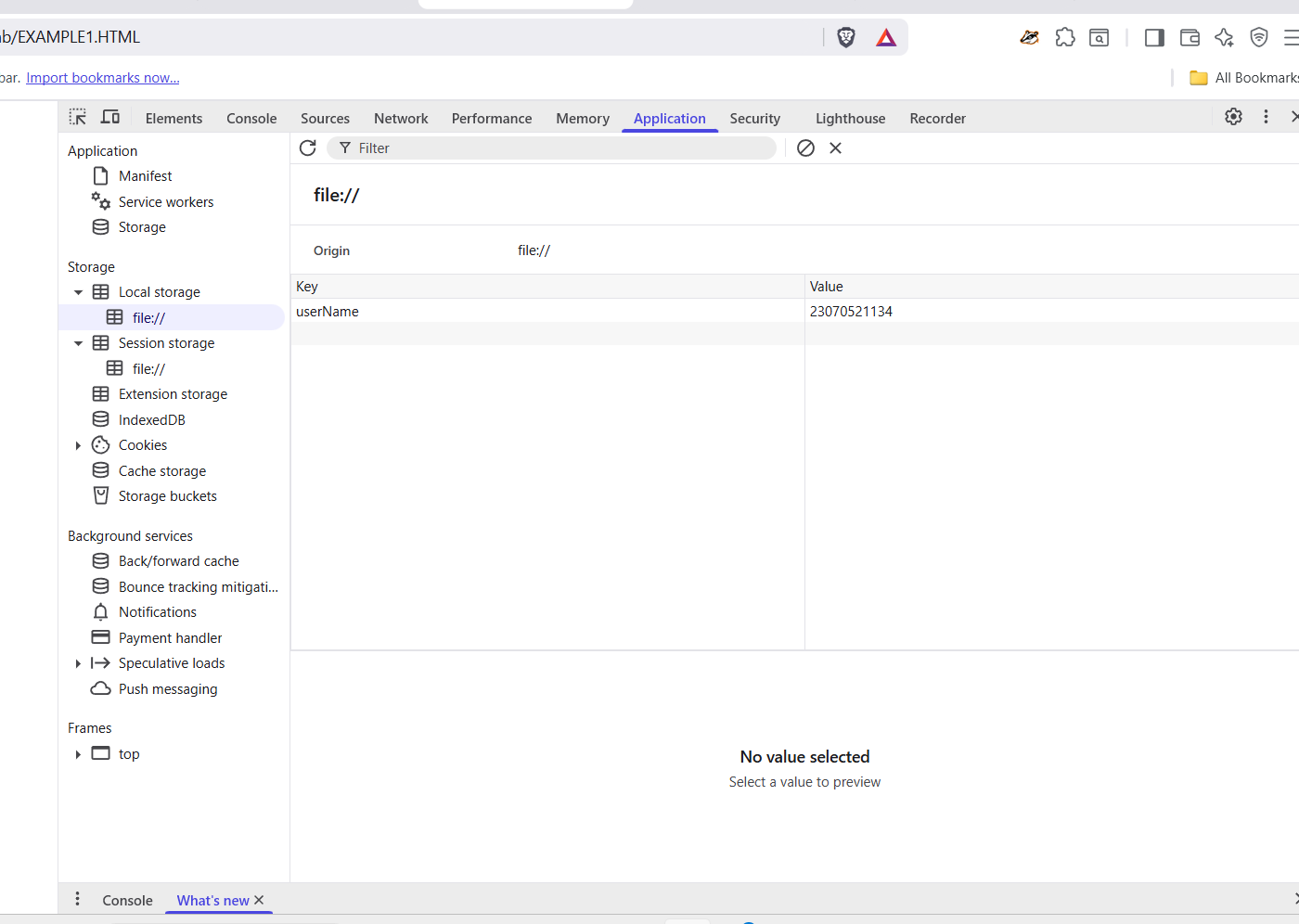
</html>



### **Explanation**

* Data entered in the text box is stored both in localStorage and sessionStorage.
* Reloading the page:  
  localStorage value persists.  
  sessionStorage value remains until tab is closed.
* Closing the tab clears sessionStorage only.

## **Task 1: Verify Persistence**

1. Open Developer Tools → Application → Storage → Local Storage.
2. Enter a name and click “Save”.
3. Reload page → Load data → You’ll still see localStorage data.
4. Close tab → Reopen → sessionStorage data disappears.

## **Example 2: Theme Switcher Using localStorage**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Theme Switcher</title>

<style>

body {

font-family: Arial;

text-align: center;

transition: background-color 0.5s, color 0.5s;

}

.dark {

background-color: #121212;

color: white;

}

.light {

background-color: #ffffff;

color: black;

}

button {

margin: 10px;

padding: 10px 20px;

border: none;

cursor: pointer;

border-radius: 5px;

}

</style>

</head>

<body>

<h1>Theme Switcher with localStorage</h1>

<button id="toggleTheme">Toggle Theme</button>

<script>

const body = document.body;

const btn = document.getElementById("toggleTheme");

// 1. Load theme from localStorage on page load

const savedTheme = localStorage.getItem("theme") || "light";

body.classList.add(savedTheme);

// 2. Update button text

btn.textContent = savedTheme === "dark" ? "Switch to Light Mode" : "Switch to Dark Mode";

// 3. Toggle theme and save to localStorage

btn.addEventListener("click", () => {

const newTheme = body.classList.contains("dark") ? "light" : "dark";

body.className = newTheme; // switch theme

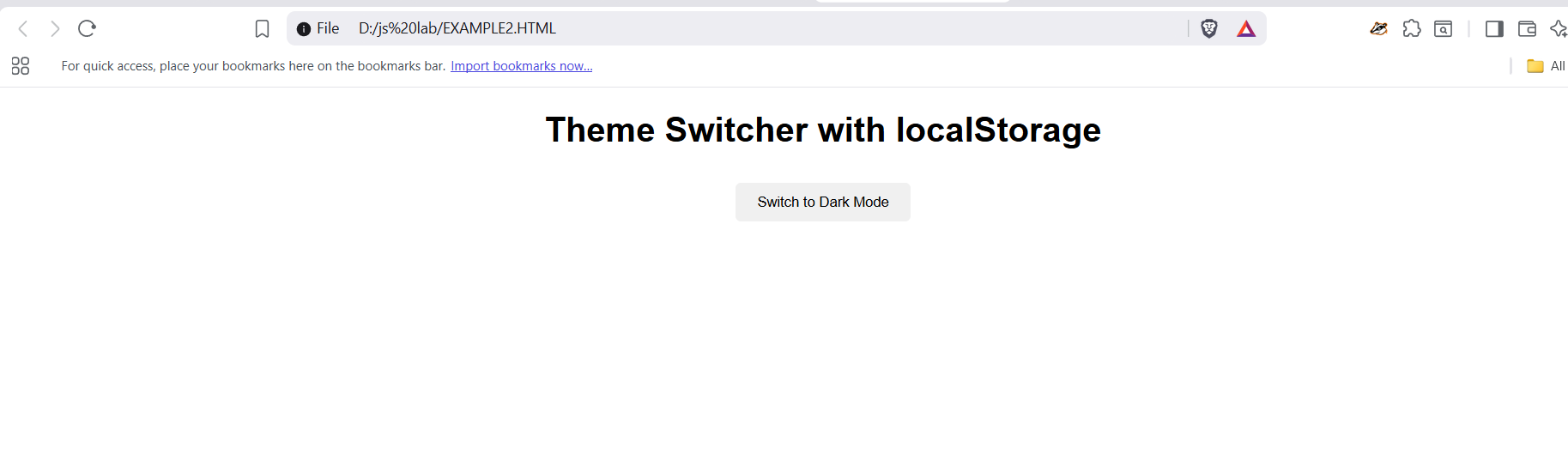
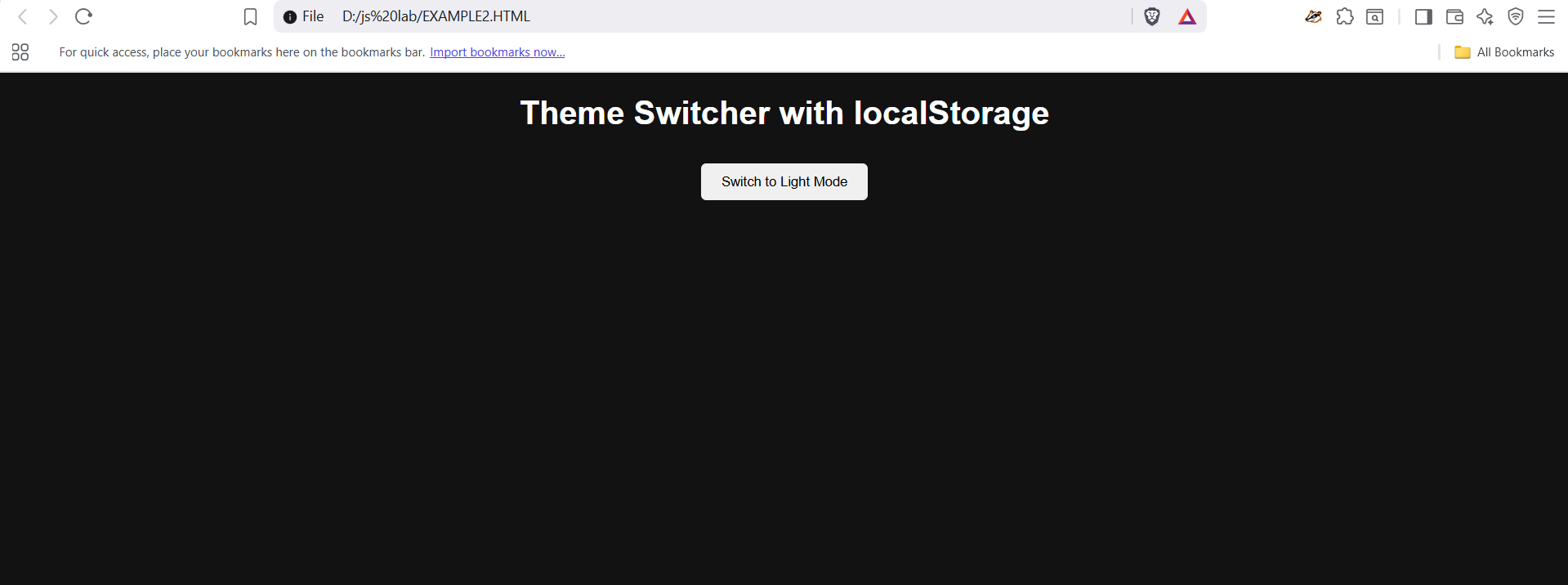
localStorage.setItem("theme", newTheme);

btn.textContent = newTheme === "dark" ? "Switch to Light Mode" : "Switch to Dark Mode";

});

</script>

</body>

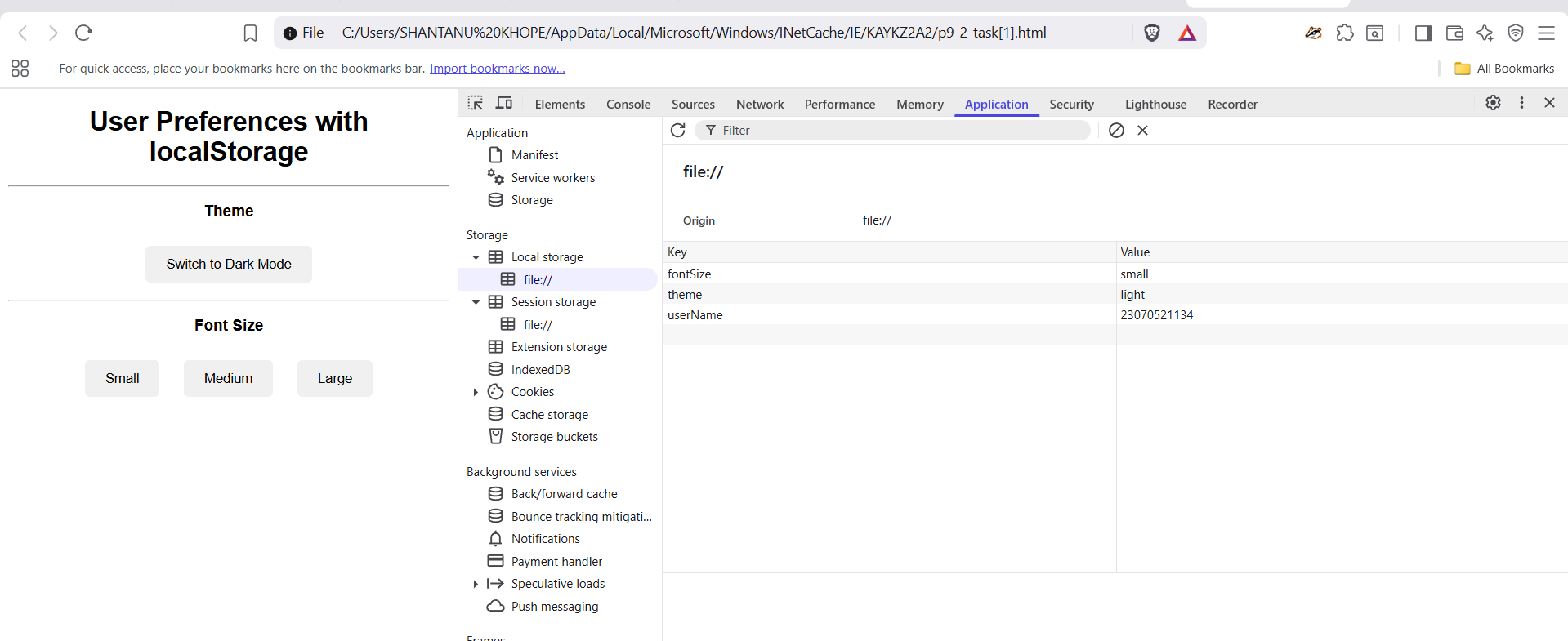
</html>

### **Explanation**

* The page loads with the **last saved theme** from localStorage.
* Clicking the button toggles between **light** and **dark** modes.
* The choice persists after reloads.

## **Task 2: Extend the Theme Switcher**

Add:

1. A **font size** preference (small, medium, large).
2. Store it in localStorage.
3. Apply the font size when the page loads.  
   

**Hint:**

localStorage.setItem("fontSize", "large");

document.body.style.fontSize = localStorage.getItem("fontSize");

## **Example 3: Using sessionStorage for Temporary Data**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<title>Session Counter</title>

</head>

<body>

<h2>Session Visit Counter</h2>

<p id="counter"></p>

<script>

if (sessionStorage.getItem("visits")) {

let count = Number(sessionStorage.getItem("visits")) + 1;

sessionStorage.setItem("visits", count);

} else {

sessionStorage.setItem("visits", 1);

}

document.getElementById("counter").textContent =

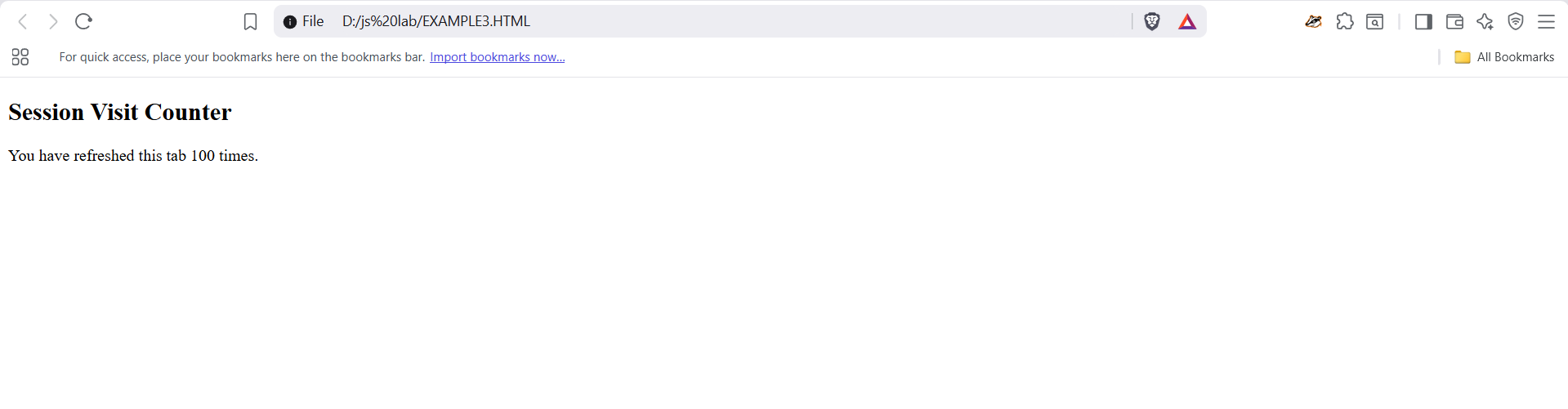
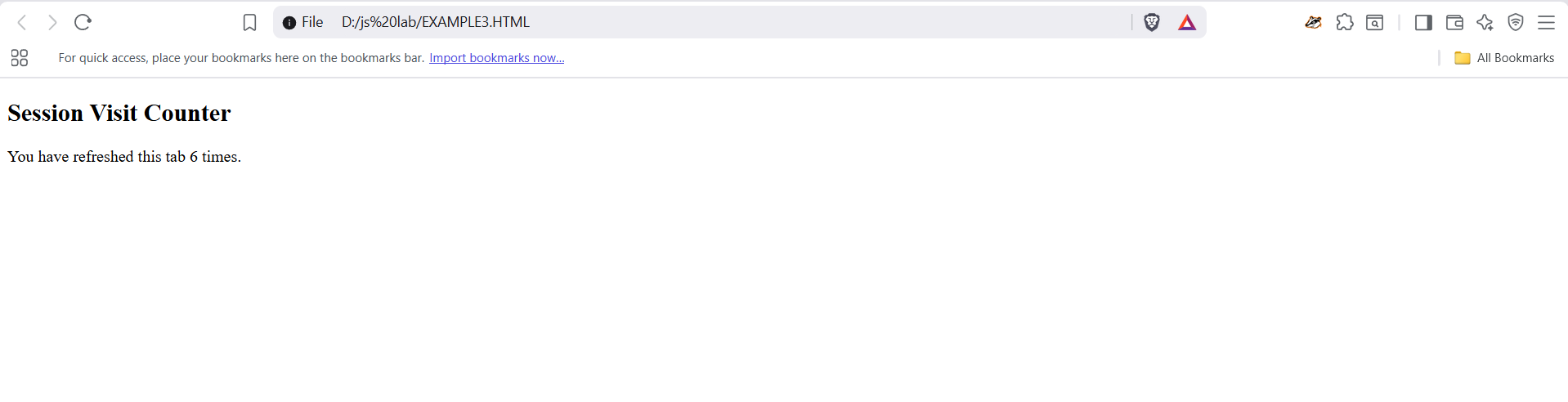
`You have refreshed this tab ${sessionStorage.getItem("visits")} times.`;

</script>

</body>

</html>

### **🧾 Explanation**

* Counts how many times a tab has been refreshed.
* When tab is closed, the counter resets to 0.

## **Task 3:**

* Create a simple **form** (name, email, message).
* Save the inputs in sessionStorage as the user types.
* Restore the values when the page reloads (only if the tab is still open).

**Hint:** Use the input event and sessionStorage.setItem() for each field.

