Local Storage and JSON

Instructor: Pramod Kumar Jena

---

1. Introduction to Local Storage and JSON

- **Local Storage**: A web storage API that allows developers to store key-value pairs in the user's browser without an expiration date.
- **JSON**: A lightweight format for storing and transporting data, often used in APIs to send data between a client and server.

---

2. Deep Dive into Local Storage

2.1 What is Local Storage?

- Local storage allows websites to store data persistently in the browser.
- Data stored remains even after the browser is closed or the computer is restarted.
- Accessible through `localStorage` in JavaScript.
- Limited to about **5MB** per origin.

2.2 Use Cases

- **User Preferences**: Save dark mode settings, language choices, or font sizes.
- **Temporary Data Storage**: Storing shopping cart items or incomplete forms that users can revisit.

2.3 Working with Local Storage in JavaScript

- **Setting Data**: `localStorage.setItem("key", "value");`
- **Getting Data**: `localStorage.getItem("key");`
- **Removing Data**: `localStorage.removeItem("key");`
- **Clearing All Data**: `localStorage.clear();`

---

3. Understanding JSON

3.1 What is JSON?

- JSON (JavaScript Object Notation) is a lightweight format for data exchange.

- It is easy to read and write for humans and easy to parse and generate for machines.

3.2 JSON Syntax

- **Data is stored in key-value pairs** (similar to JavaScript objects).
- Strings should be in **double quotes**.
- Allows **arrays and nested objects**.

3.3 JSON Methods in JavaScript

- **Convert to JSON**: `JSON.stringify(object);`
- **Parse JSON**: `JSON.parse(jsonString);`

3.4 Real-World Example: How JSON is Used in APIs

- JSON is widely used in APIs to transfer data between the client and server.
- Example: Receiving weather data from an API in JSON format and then parsing it to display on a website.

---

4. Practical Example: Storing User Preferences in Local Storage

Mini Project: "Remember User's Theme Preference"

Goal: To store and retrieve a user's theme (dark mode or light mode) using local storage and JSON.

Step 1: Set Up Basic HTML Structure

html
Copy code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>User Theme Preference</title>
    <style>
        body.dark-mode {
            background-color: #333;
            color: white;
        }
```

```
        body.light-mode {
            background-color: #f4f4f4;
            color: black;
        }
    </style>
</head>
<body class="light-mode">
    <button id="toggleTheme">Toggle Theme</button>
    <script src="app.js"></script>
</body>
</html>
```

Step 2: Write JavaScript for Theme Toggle and Local Storage

javascript

Copy code

```
// Select button and define default theme
const toggleThemeButton = document.getElementById("toggleTheme");

// Check if theme preference is already stored in localStorage
const savedTheme = localStorage.getItem("theme");

if (savedTheme) {
    // Apply saved theme
    document.body.className = savedTheme;
}

// Event listener to toggle theme
toggleThemeButton.addEventListener("click", () => {
    // Toggle between dark-mode and light-mode
    if (document.body.className === "light-mode") {
        document.body.className = "dark-mode";
    } else {
        document.body.className = "light-mode";
    }

    // Store the updated theme in local storage
    localStorage.setItem("theme", document.body.className);
});
```

Explanation:

- **Initial Theme Check**: On page load, JavaScript checks `localStorage` for a saved theme. If one is found, it applies that theme to the body.
- **Toggle Theme**: Each time the button is clicked, it switches between light and dark mode.
- **Save Theme**: After each toggle, it saves the current theme in `localStorage`, so when the user revisits the page, it remembers their last selected theme.

Real-Life Relevance

- This approach is used on most modern websites that support theme toggling, remembering user settings even after they leave and return.

Additional Exercises

1. **Save and Retrieve JSON Objects in Local Storage**:
    - Practice storing complex objects (like a user profile) in local storage using `JSON.stringify` and `JSON.parse`.
2. **Form Autofill**:
    - Save form data (like name and email) in local storage to autofill for returning users.