Background Story

It was a cool evening in Quezon City when Odessa closed her laptop—only to remember she'd set her code editor to light mode by mistake. "Ay, sayang! I liked dark mode," she sighed. The next morning, she found her teammates also toggling between light and dark themes every time they refreshed the page. Preferences were lost, scores in the intern coding game disappeared, and even the "welcome back" greeting forgot their names.

Her manager, Ate Liza, laughed and said, "Ods, make this app remember! Parang memory ng lolo natin—dapat di ka nakakalimot." Odessa realized she needed a way to keep settings and small data across browser sessions. That's when she discovered `localStorage`.

She imagined a small barangay hall website where residents choose light or dark theme, track their health scores, and come back days later with everything intact—even if they closed the browser. Odessa got excited. She started by saving a simple "theme" preference:

```
localStorage.setItem("theme", "dark");
```

Then she read it back:

```
const theme = localStorage.getItem("theme"); // "dark"
```

Soon she added JSON support—storing arrays of scores for the startup's weekly game leaderboard:

```
const scores = [85, 92, 78];
localStorage.setItem("scores", JSON.stringify(scores));
const saved = JSON.parse(localStorage.getItem("scores"));
```

Odessa tested it by refreshing her page again and again—the theme stayed, the scores remained. She even built a small toggle button:

```
btn.addEventListener("click", () => {
  const next = theme === "light" ? "dark" : "light";
  localStorage.setItem("theme", next);
  applyTheme(next);
});
```

Her teammates cheered. No more lost preferences! Odessa felt proud—she'd made her app remember like a faithful lolo's diary. As the sun set over QC's skyline, she envisioned bigger apps saving cart items, draft forms, and more. But first, she'd teach you how to save memory with `localStorage`. 🚀

Theory & Lecture Content

Browsers provide a simple key–value store called `localStorage` for persisting small data across sessions. Data survives page reloads and even browser restarts.

1. localStorage.setItem(key, value)

    - Stores a string `value` under `key`.
    - Example:

    ```
    localStorage.setItem("username", "Odessa");
    ```

2. localStorage.getItem(key)

    - Retrieves the string value for `key`.
    - Returns `null` if the key doesn't exist.
    - Example:

    ```
    const name = localStorage.getItem("username"); // "Odessa"
    ```

3. JSON.stringify / JSON.parse

    - `localStorage` stores only strings. To store objects or arrays, convert them:

    ```
    const prefs = { theme: "dark", fontSize: 14 };
    localStorage.setItem("prefs", JSON.stringify(prefs));
    // later…
    const saved = JSON.parse(localStorage.getItem("prefs"));
    ```

4. Removing and Clearing

    - `localStorage.removeItem(key)` deletes one item.
    - `localStorage.clear()` removes all.

Best Practices

- Namespace your keys: e.g., `"myApp-theme"` to avoid collisions.
- Check for `null` before parsing.
- Don't store sensitive data—localStorage is not encrypted.

References:
https://developer.mozilla.org/en-US/docs/Web/API/Window/localStorage

---

Exercises

Exercise 1: Theme Preference Storage

Problem Statement
Implement two functions in `exercise1.js` to save and load a user's theme preference (`"light"` or `"dark"`) using `localStorage`.

TODOs

- Write `saveTheme(theme)` that calls `localStorage.setItem("app-theme", theme)`.
- Write `loadTheme()` that returns the stored theme or `"light"` if none.

Starter Code (exercise1.js)

```
// exercise1.js

/**
 * Saves theme preference.
 * @param {string} theme - "light" or "dark"
 */
export function saveTheme(theme) {
  // TODO: implement
}

/**
 * Loads theme preference.
 * @returns {string}
 */
export function loadTheme() {
  // TODO: implement
}
```

Full Solution (exercise1.js)

```
export function saveTheme(theme) {
  localStorage.setItem("app-theme", theme);
}

export function loadTheme() {
  const stored = localStorage.getItem("app-theme");
  return stored === "dark" ? "dark" : "light";
}
```

Exercise 2: Score List Persistence

Problem Statement
In `exercise2.js`, implement functions to save an array of numeric scores and retrieve them.

TODOs

- Write `saveScores(scores)` that stores JSON string under key `"app-scores"`.
- Write `loadScores()` that returns an array of numbers (empty array if none).

Starter Code (exercise2.js)

```
// exercise2.js

/**
 * @param {number[]} scores
 */
export function saveScores(scores) {
  // TODO: implement
}

/**
 * @returns {number[]}
 */
export function loadScores() {
  // TODO: implement
}
```

Full Solution (exercise2.js)

```
export function saveScores(scores) {
  localStorage.setItem("app-scores", JSON.stringify(scores));
}

export function loadScores() {
  const raw = localStorage.getItem("app-scores");
  if (!raw) return [];
  try {
    const arr = JSON.parse(raw);
    return Array.isArray(arr) ? arr : [];
  } catch {
    return [];
  }
}
```

Exercise 3: Theme Toggle UI

Problem Statement
Build a small HTML page that toggles light/dark theme and persists the choice. Use `saveTheme` and `loadTheme`.

TODOs

- In `index.html`, add a `<button id="themeBtn">`.
- In `script.js`:

1. Import `saveTheme` & `loadTheme`.
2. On load, read theme and apply it (add `<body class="dark">` or none).
3. On button click, toggle theme, save it, and update class.

Starter Code (index.html)

```html
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />
    <title>Theme Toggle</title>
    <style>
      body.dark {
        background: #222;
        color: #eee;
      }
      body.light {
        background: #fff;
        color: #000;
      }
    </style>
  </head>
  <body>
    <button id="themeBtn">Toggle Theme</button>
    <script type="module" src="script.js"></script>
  </body>
</html>
```

Starter Code (script.js)

```js
// script.js
import { saveTheme, loadTheme } from "./exercise1.js";

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

// TODO: implement theme load and toggle
```

Full Solution (script.js)

```js
import { saveTheme, loadTheme } from "./exercise1.js";

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

// apply saved theme on load
let current = loadTheme();
document.body.className = current;

// update button text
```

```
btn.textContent = `Switch to ${current === "light" ? "dark" : "light"}`;

// toggle on click
btn.addEventListener("click", () => {
  current = current === "light" ? "dark" : "light";
  document.body.className = current;
  saveTheme(current);
  btn.textContent = `Switch to ${current === "light" ? "dark" : "light"}`;
});
```

Test Cases

exercise1.test.js

```
/**
 * @jest-environment jsdom
 */
import { saveTheme, loadTheme } from "./exercise1.js";

describe("Theme Preference", () => {
  beforeEach(() => localStorage.clear());

  test("default theme is light", () => {
    expect(loadTheme()).toBe("light");
  });

  test("saves and loads dark theme", () => {
    saveTheme("dark");
    expect(localStorage.getItem("app-theme")).toBe("dark");
    expect(loadTheme()).toBe("dark");
  });

  test("invalid value defaults to light", () => {
    localStorage.setItem("app-theme", "purple");
    expect(loadTheme()).toBe("light");
  });
});
```

exercise2.test.js

```
/**
 * @jest-environment jsdom
 */
import { saveScores, loadScores } from "./exercise2.js";

describe("Score Persistence", () => {
  beforeEach(() => localStorage.clear());
  test("loads empty array when none", () => {
    expect(loadScores()).toEqual([]);
  });

  test("saves and loads scores correctly", () => {
    const arr = [10, 20, 30];
    saveScores(arr);
    expect(localStorage.getItem("app-scores")).toBe(JSON.stringify(arr));
    expect(loadScores()).toEqual(arr);
  });

  test("handles invalid JSON gracefully", () => {
    localStorage.setItem("app-scores", "not-json");
    expect(loadScores()).toEqual([]);
  });
});
```

exercise3.test.js

```
/**
 * @jest-environment jsdom
 */
import fs from "fs";
import path from "path";

beforeAll(() => {
  document.body.innerHTML = fs.readFileSync(
    path.resolve(__dirname, "index.html"),
    "utf8"
  );
  require("./script.js");
});

describe("Theme Toggle UI", () => {
  const btn = () => document.getElementById("themeBtn");

  beforeEach(() => localStorage.clear());

  test("initializes with light theme", () => {
    btn();
    expect(document.body.className).toBe("light");
    expect(btn().textContent).toMatch(/Switch to dark/);
  });

  test("toggles to dark and persists", () => {
    btn().click();
    expect(document.body.className).toBe("dark");
    expect(localStorage.getItem("app-theme")).toBe("dark");
    expect(btn().textContent).toMatch(/Switch to light/);
  });
```

```
    test("persists theme across reload", () => {
      // set dark in storage
      localStorage.setItem("app-theme", "dark");
      // re-require script to simulate reload
      document.body.innerHTML = fs.readFileSync(
        path.resolve(__dirname, "index.html"),
        "utf8"
      );
      require("./script.js");
      expect(document.body.className).toBe("dark");
      expect(btn().textContent).toMatch(/Switch to light/);
    });
  });
```

Closing Story

With her new `localStorage` skills, Odessa's apps remembered everything—user themes, game scores, even draft notes—just like her lolo's old diary. The logistics dashboard now stayed in dark mode if drivers preferred it, and the barangay tool kept residents' preferences intact.

Later that night, Odessa reviewed her code and smiled. "Saving memory isn't just tech; it's about respecting users' choices," she thought. Her next challenge? Advanced storage with IndexedDB and syncing data across devices. But for now, her apps would never forget—and neither would she. 🚀