# INTERNSHIP REPORT

# WEEK 5 DAY 4

| | | | |
|---|---|---|---|
| Submitted to: | **Ali Hyder** | Submission Date: | **24th July, 2025** |
| Internship Domain: | **Front Development** | Internship Name: | **ProSensia** |
| Student Name: | **Yasal Qamar** | Roll No. | **S25031** |

# JavaScript Advanced

# Topics: JavaScript JSON & LocalStorage

### Objective

To learn how JSON is used for storing and transferring data in web applications and how to utilize the browser's LocalStorage for persistent data storage.

### Topics Covered

### 1. JSON (JavaScript Object Notation)

- **Definition:** JSON is a lightweight data-interchange format that is easy for humans to read and write, and easy for machines to parse and generate.
- **Syntax Rules:**
  - Data is written in key-value pairs.
  - Strings must be in double quotes.
  - Supports objects, arrays, numbers, strings, booleans, and null.
- **Methods:**
  - **JSON.stringify(object)** – Converts a JavaScript object into a JSON string.
  - **JSON.parse(string)** – Converts a JSON string into a JavaScript object.

### Example:

```
let user = { name: "Yasal", age: 21 };
let jsonString = JSON.stringify(user);   // Convert object to JSON
console.log(jsonString);                 // {"name":"Yasal","age":21}
```

```
let parsedUser = JSON.parse(jsonString);  // Convert JSON back to object
console.log(parsedUser.name);          // Yasal
```

## 2. LocalStorage

- **Definition:** LocalStorage allows you to store key-value pairs in the browser with **no expiration date** (persistent even after the browser is closed).
- **Common Methods:**
    - localStorage.setItem(key, value) – Stores data.
    - localStorage.getItem(key) – Retrieves data.
    - localStorage.removeItem(key) – Removes a specific key.
    - localStorage.clear() – Clears all data.

**Example:**

```
// Store data
localStorage.setItem("username", "Yasal");

// Retrieve data
let userName = localStorage.getItem("username");
console.log(userName);  // Yasal

// Remove data
localStorage.removeItem("username");

// Clear all data
localStorage.clear();
```

## 3. JSON with LocalStorage

Since LocalStorage only stores strings, JSON is used to store objects/arrays.

## Example:

```
let student = { name: "Yasal", course: "Frontend" };
localStorage.setItem("student", JSON.stringify(student));

let storedData = JSON.parse(localStorage.getItem("student"));
console.log(storedData.course); // Frontend
```

# CODING

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Advanced To-Do List</title>
  <style>
    * { box-sizing: border-box; }
    body {
      font-family: "Segoe UI", sans-serif;
      background: var(--bg);
      color: var(--text);
      display: flex;
      justify-content: center;
      padding: 40px;
      transition: background 0.3s, color 0.3s;
    }
    :root {
      --bg: #f1f5f9;
      --text: #333;
      --card: #fff;
      --border: #e5e7eb;
    }
    body.dark {
      --bg: #1e293b;
      --text: #e2e8f0;
      --card: #334155;
      --border: #475569;
    }
    .todo-container {
      background: var(--card);
      width: 100%;
      max-width: 450px;
      padding: 20px;
      border-radius: 12px;
      box-shadow: 0 4px 10px rgba(0, 0, 0, 0.2);
      transition: background 0.3s;
    }
```

```
    h1 { text-align: center; color: #2563eb; }
    form { display: flex; gap: 10px; margin: 20px 0; }
    input[type="text"] {
      flex: 1; padding: 10px;
      border: 1px solid var(--border);
      border-radius: 8px;
      background: var(--bg);
      color: var(--text);
    }
    button {
      padding: 10px 16px; border: none; border-radius: 8px;
      cursor: pointer; font-weight: bold;
      background: #2563eb; color: #fff;
      transition: background 0.3s;
    }
    button:hover { background: #1e4fc9; }
    ul { list-style: none; padding: 0; margin: 0; }
    li {
      display: flex; justify-content: space-between; align-items: center;
      padding: 8px 10px; border: 1px solid var(--border);
      margin-bottom: 8px; border-radius: 8px;
      transition: transform 0.3s ease, opacity 0.3s ease;
    }
    li.done span { text-decoration: line-through; opacity: 0.6; }
    li.enter { transform: translateY(-10px); opacity: 0; }
    .filters, .actions { text-align: center; margin-top: 10px; }
    .filters button, .actions button {
      margin: 5px; background-color: #e5e7eb; color: #333;
    }
    .filters button.active { background-color: #2563eb; color: #fff; }
    .summary { margin-top: 14px; text-align: center; font-size: 0.9rem; }
    .theme-toggle { float: right; margin-bottom: 10px; cursor: pointer; }
    .edit-input {
      flex: 1; border: 1px solid #ccc; border-radius: 4px;
      padding: 4px 6px; font-size: 0.9rem;
    }
  </style>
</head>
<body>
  <div class="todo-container">
    <button class="theme-toggle" id="theme-toggle">🌙</button>
    <h1>Advanced To-Do List</h1>

    <form id="todo-form">
```

```
      <input id="todo-input" type="text" placeholder="Add a new task..."
required />
      <button>Add</button>
    </form>

    <ul id="todo-list"></ul>

    <div class="filters">
      <button data-filter="all" class="active">All</button>
      <button data-filter="active">Active</button>
      <button data-filter="completed">Completed</button>
    </div>

    <div class="actions">
      <button id="clear-completed">Clear Completed</button>
    </div>

    <div class="summary" id="summary"></div>
  </div>

  <script>
    const STORAGE_KEY = "todos_v2";
    const THEME_KEY = "theme_mode";

    let state = {
      todos: load(STORAGE_KEY, [
        { id: crypto.randomUUID(), text: "Learn JavaScript", done: false },
        { id: crypto.randomUUID(), text: "Complete Internship Task", done:
true },
        { id: crypto.randomUUID(), text: "Go for a walk", done: false },
      ]),
      filter: "all",
    };

    const $form = document.getElementById("todo-form");
    const $input = document.getElementById("todo-input");
    const $list = document.getElementById("todo-list");
    const $filters = document.querySelector(".filters");
    const $summary = document.getElementById("summary");
    const $clearCompleted = document.getElementById("clear-completed");
    const $themeToggle = document.getElementById("theme-toggle");

    // Theme setup
    if (localStorage.getItem(THEME_KEY) === "dark")
document.body.classList.add("dark");
```

```
    render();

    $themeToggle.addEventListener("click", () => {
      document.body.classList.toggle("dark");
      localStorage.setItem(THEME_KEY, document.body.classList.contains("dark")
? "dark" : "light");
    });

    $form.addEventListener("submit", (e) => {
      e.preventDefault();
      const text = $input.value.trim();
      if (!text) return;
      state.todos.push({ id: crypto.randomUUID(), text, done: false });
      $input.value = "";
      persist();
      render(true);
    });

    $list.addEventListener("click", (e) => {
      const li = e.target.closest("li[data-id]");
      if (!li) return;
      const id = li.dataset.id;
      if (e.target.matches(".toggle")) toggleTodo(id);
      if (e.target.matches(".delete")) animateDelete(id, li);
    });

    // Double-click to edit
    $list.addEventListener("dblclick", (e) => {
      const span = e.target.closest("span");
      if (!span) return;
      const li = span.closest("li");
      const id = li.dataset.id;
      startEditTask(id, span);
    });

    $filters.addEventListener("click", (e) => {
      if (!e.target.dataset.filter) return;
      [...$filters.querySelectorAll("button")].forEach((b) =>
b.classList.remove("active"));
      e.target.classList.add("active");
      state.filter = e.target.dataset.filter;
      render();
    });
```

```
$clearCompleted.addEventListener("click", () => {
  state.todos = state.todos.filter((t) => !t.done);
  persist();
  render();
});

function toggleTodo(id) {
  const t = state.todos.find((t) => t.id === id);
  if (t) t.done = !t.done;
  persist();
  render();
}

function animateDelete(id, li) {
  li.style.opacity = "0";
  li.style.transform = "translateX(40px)";
  setTimeout(() => {
    state.todos = state.todos.filter((t) => t.id !== id);
    persist();
    render();
  }, 300);
}

function startEditTask(id, span) {
  const t = state.todos.find((t) => t.id === id);
  const input = document.createElement("input");
  input.type = "text";
  input.value = t.text;
  input.className = "edit-input";
  span.replaceWith(input);
  input.focus();
  input.addEventListener("blur", () => finishEditTask(id, input));
  input.addEventListener("keydown", (e) => {
    if (e.key === "Enter") input.blur();
  });
}

function finishEditTask(id, input) {
  const t = state.todos.find((t) => t.id === id);
  if (t) t.text = input.value.trim() || t.text;
  persist();
  render();
}

function filteredTodos() {
```

```
      if (state.filter === "active") return state.todos.filter((t) =>
!t.done);
      if (state.filter === "completed") return state.todos.filter((t) =>
t.done);
      return state.todos;
    }

    function render(isNew = false) {
      $list.innerHTML = "";
      for (const t of filteredTodos()) {
        const li = document.createElement("li");
        li.dataset.id = t.id;
        li.className = t.done ? "done" : "";
        li.innerHTML = `
          <input type="checkbox" class="toggle" ${t.done ? "checked" : ""}/>
          <span>${escapeHTML(t.text)}</span>
          <button class="delete">✕</button>
        `;
        if (isNew && t === state.todos[state.todos.length - 1]) {
          li.classList.add("enter");
          $list.appendChild(li);
          requestAnimationFrame(() => li.classList.remove("enter"));
        } else {
          $list.appendChild(li);
        }
      }
      updateSummary();
    }

    function updateSummary() {
      const left = state.todos.filter((t) => !t.done).length;
      const total = state.todos.length;
      $summary.textContent = `${left} task${left !== 1 ? "s" : ""} left
(${total} total)`;
    }

    function persist() { save(STORAGE_KEY, state.todos); }
    function save(key, value) { localStorage.setItem(key,
JSON.stringify(value)); }
    function load(key, fallback) {
      try { const raw = localStorage.getItem(key); return raw ?
JSON.parse(raw) : fallback; }
      catch { return fallback; }
    }
```

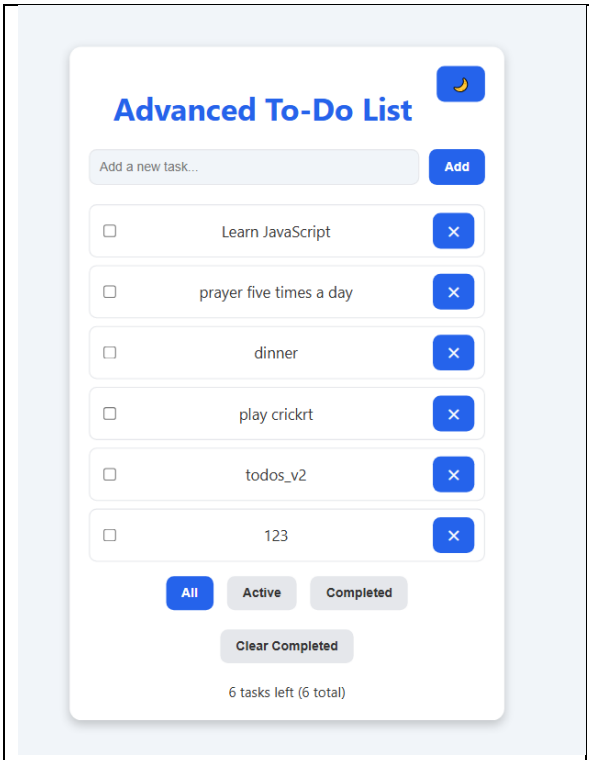
```
    function escapeHTML(str) { const div = document.createElement("div");
div.textContent = str; return div.innerHTML; }
  </script>
</body>
</html>
```

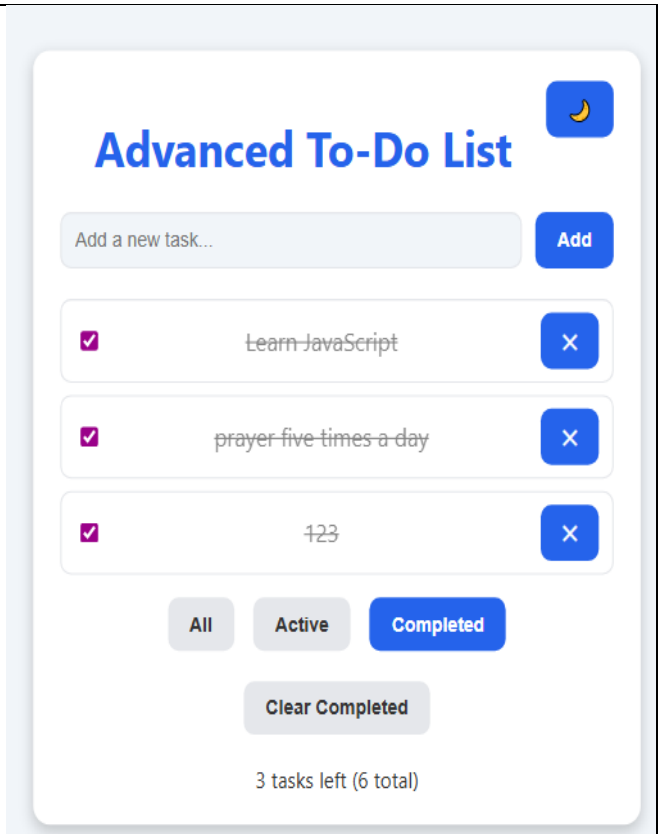
**.js**

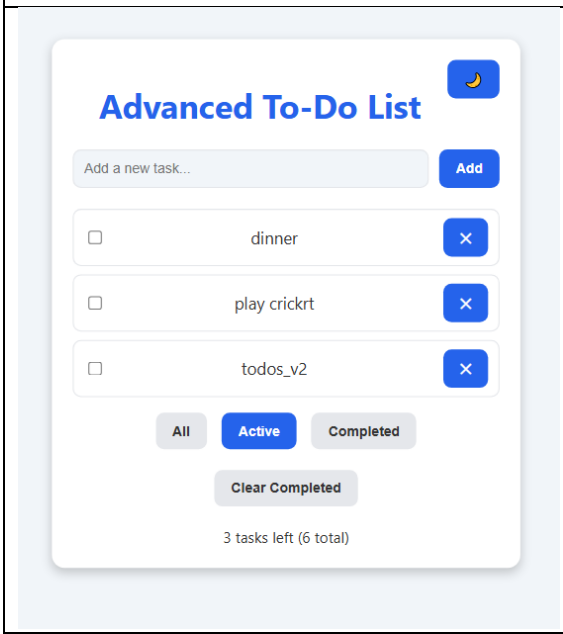
```
const STORAGE_KEY = "todos_v1";

let state = {
  todos: load(STORAGE_KEY, []),
  filter: "all",
};

const $form = document.getElementById("todo-form");
const $input = document.getElementById("todo-input");
const $list = document.getElementById("todo-list");
const $filters = document.querySelector(".filters");

render();

$form.addEventListener("submit", (e) => {
  e.preventDefault();
  const text = $input.value.trim();
  if (!text) return;

  state.todos.push({
    id: crypto.randomUUID(),
    text,
    done: false,
    createdAt: Date.now(),
  });

  $input.value = "";
  persist();
  render();
});

$list.addEventListener("click", (e) => {
  const li = e.target.closest("li[data-id]");
  if (!li) return;
```

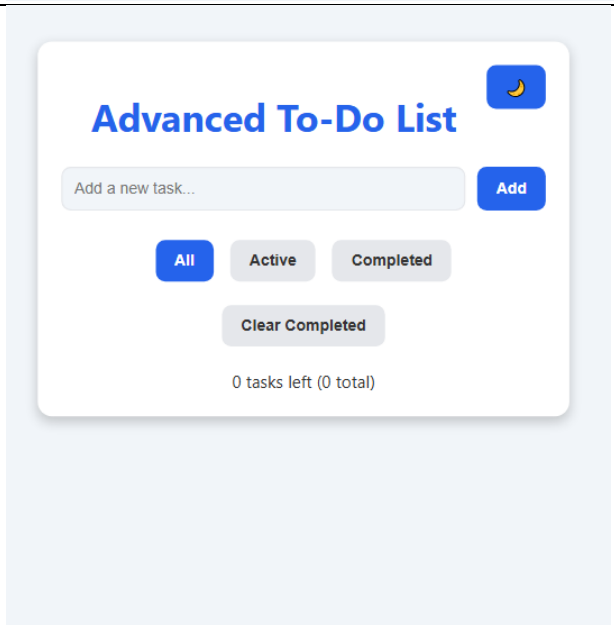
```
  const id = li.dataset.id;

  if (e.target.matches(".toggle")) {
    toggleTodo(id);
  } else if (e.target.matches(".delete")) {
    deleteTodo(id);
  }
});

$filters.addEventListener("click", (e) => {
  if (!e.target.dataset.filter) return;
  [...$filters.querySelectorAll("button")].forEach(b =>
b.classList.remove("active"));
  e.target.classList.add("active");
  state.filter = e.target.dataset.filter;
  render();
});

function toggleTodo(id) {
  const t = state.todos.find(t => t.id === id);
  if (t) t.done = !t.done;
  persist();
  render();
}

function deleteTodo(id) {
  state.todos = state.todos.filter(t => t.id !== id);
  persist();
  render();
}

function filteredTodos() {
  if (state.filter === "active") return state.todos.filter(t => !t.done);
  if (state.filter === "completed") return state.todos.filter(t => t.done);
  return state.todos;
}

function render() {
  $list.innerHTML = "";
  for (const t of filteredTodos()) {
    const li = document.createElement("li");
    li.dataset.id = t.id;
    li.className = t.done ? "done" : "";
    li.innerHTML = `
```

```
      <input type="checkbox" class="toggle" ${t.done ? "checked" : ""}/>
      <span>${escapeHTML(t.text)}</span>
      <button class="delete">✕</button>
    `;
    $list.appendChild(li);
  }
}

function persist() {
  save(STORAGE_KEY, state.todos);
}

function save(key, value) {
  localStorage.setItem(key, JSON.stringify(value));
}

function load(key, fallback) {
  try {
    const raw = localStorage.getItem(key);
    return raw ? JSON.parse(raw) : fallback;
  } catch {
    return fallback;
  }
}

function escapeHTML(str) {
  const div = document.createElement("div");
  div.textContent = str;
  return div.innerHTML;
}
```


Advanced To-Do List
Add a new task...
Add
Learn JavaScript
prayer five times a day
dinner
play crickrt
todos_v2
123
All
Active
Completed
Clear Completed
6 tasks left (6 total)


Advanced To-Do List
Add a new task...
Add
Learn JavaScript
prayer five times a day
123
All
Active
Completed
Clear Completed
3 tasks left (6 total)


Advanced To-Do List
Add a new task...
Add
dinner
play crickrt
todos_v2
All
Active
Completed
Clear Completed
3 tasks left (6 total)


Advanced To-Do List
Add a new task...
Add
All
Active
Completed
Clear Completed
0 tasks left (0 total)

## Explanation:

### A) Saving Todos to LocalStorage

```
function persist() {
  save(STORAGE_KEY, state.todos);
}

function save(key, value) {
  localStorage.setItem(key, JSON.stringify(value));
}
```

- **state.todos** is an array of task objects like:

```
    [
      { id: "123", text: "Learn JavaScript", done: false },
      { id: "456", text: "Complete Internship Task", done: true }
    ]
```

- We **convert this array to JSON string** with JSON.stringify(value) before saving:

```
'[{"id":"123","text":"Learn JavaScript","done":false}, {"id":"456","text":"Complete Internship Task","done":true}]'
```

- Then localStorage.setItem() stores this string under the key "todos_v2".

### B) Loading Todos from LocalStorage

```
let state = {
  todos: load(STORAGE_KEY, [ ... default tasks ... ]),
  filter: "all",
};

function load(key, fallback) {
  try {
    const raw = localStorage.getItem(key);
    return raw ? JSON.parse(raw) : fallback;
  } catch {
    return fallback;
  }
}
```

- localStorage.getItem(key) retrieves the string we saved earlier.
- JSON.parse(raw) converts the JSON string **back to a JavaScript array of objects**.

- If there's nothing saved yet, we use a **fallback** array of default tasks.

### Why This Matters in Our App?

- Without LocalStorage, all tasks would disappear every time we refresh the browser.
- By using JSON & LocalStorage, the tasks persist:
    - Add a new task.
    - Refresh the page.
    - The tasks remain because they are loaded back from LocalStorage using JSON.parse.

## CONCUSLION:

In this mini To-Do app:

1. **Tasks are stored persistently** using localStorage. Even if you refresh or close the browser, tasks remain.
2. **JSON.stringify()** converts the JavaScript array of tasks to a string for storage, and **JSON.parse()** retrieves it back as an array.
3. **CRUD operations**:
    - **Create:** Add new tasks with the "Add Task" button.
    - **Read:** Tasks are rendered from LocalStorage on every page load (renderTasks()).
    - **Delete (Single):** Clicking the "✕" button removes a task from the list and updates LocalStorage.
    - **Delete All:** "Clear All" removes all tasks and clears LocalStorage.
4. **Security:** escapeHTML() ensures that any text input is displayed safely, preventing malicious HTML injection.

---