Internship Report – Frontend Dev

Week 5: JavaScript Advanced Topics

Name: Zainab
Father Name: Assad Qayyum
Date: 25th July, 2025
Internship Domain: Front-end Intern
Task: Mini Project: Quiz App or LocalStorage Note App

Task Overview: (Day5)

The task assigned for Week-5 Day5 was to apply the advanced JavaScript concepts learned during the week by building a mini-project. The options given were either to build a Quiz App or a Note App using LocalStorage. I chose to build the **LocalStorage Note App**, as it allowed me to practically implement a wide range of JavaScript concepts, especially those involving data persistence and dynamic DOM manipulation.

Content Covered:

The objective of this mini-project was to:

- Implement a real-world application using advanced JavaScript features.
- Practice working with user input, browser storage, and interactive UI updates.
- Showcase understanding of concepts like events, objects, arrays, ES6 features, and asynchronous JavaScript in a functional web application.
- Develop clean, readable, and reusable code following modern front-end development practices.

A Mini project: LocalStorage Note App

This is a simple but fully functional **Note-Taking Web App** built using HTML, CSS, and JavaScript. Users can:

- Write and save text notes
- View all previously saved notes (even after refreshing the browser)
- Delete individual notes

The app stores notes in the **browser's LocalStorage**, which means they persist between sessions without requiring a server. Each note is created as an object and dynamically rendered on the page using DOM methods.

Concepts Used:

Here are the advanced JavaScript concepts integrated into this project:

1. Events

Used addEventListener() to handle user actions like adding notes and DOMContentLoaded.

Also used onclick for deleting individual notes.

2. Objects & Object Methods

Each note is stored as an object with properties like id and content.

Object destructuring is used when displaying notes.

3. Array Methods

.map() is used to loop through the notes array and display each note dynamically.

.filter() is used to remove a note from the array when it is deleted.

4. JSON & LocalStorage

Notes are stored using localStorage.setItem() after converting the array of note objects into a JSON string using JSON.stringify().

Notes are retrieved using localStorage.getItem() and parsed using JSON.parse().

5. ES6+ Features

Template literals are used to build dynamic HTML strings.

Destructuring is used when accessing note properties.

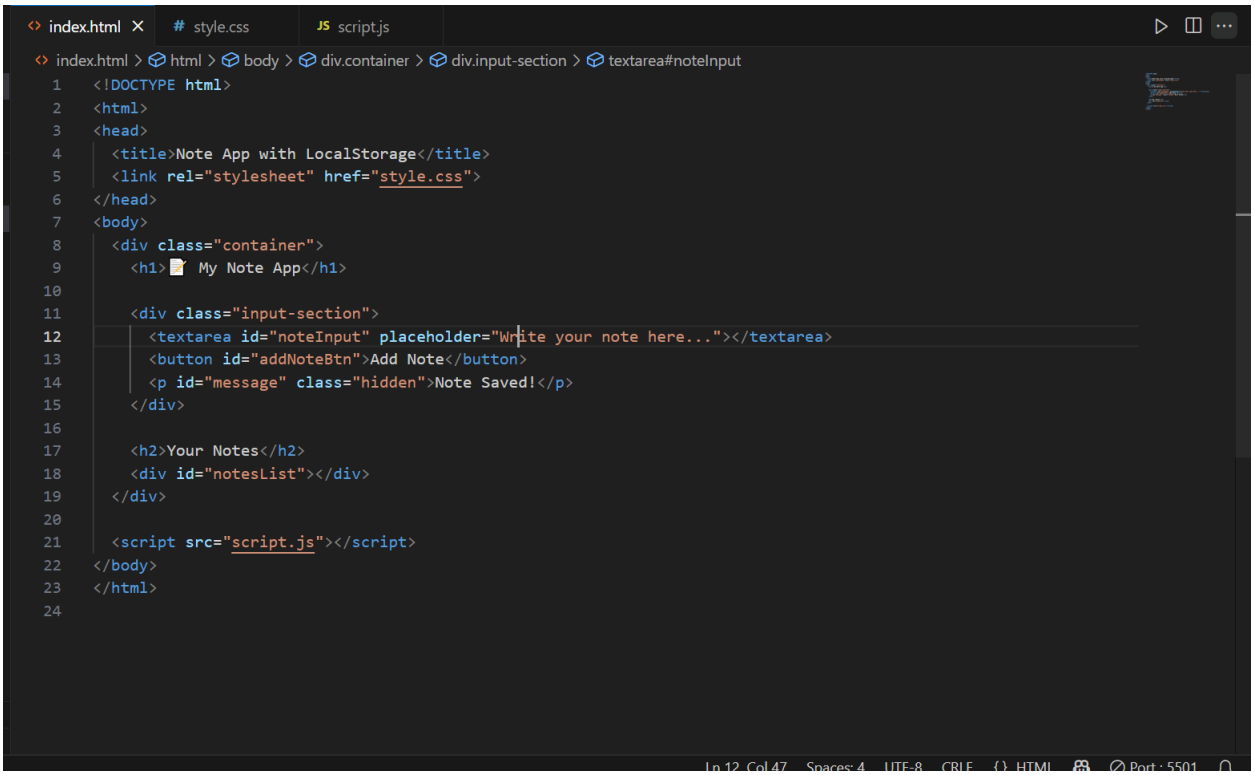
Spread operator (...) is used to add a new note to the existing array in an immutable way.

6. Asynchronous JavaScript

setTimeout() is used to display a temporary success message ("Note Saved!") after adding a note.

Practice Code:

Html:



CSS:

index.html | style.css | script.js

style.css > ...

```css
body {
  font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
  background-color: #121212;
  color: #e0e0e0;
  margin: 0;
  padding: 20px;
}

.container {
  max-width: 700px;
  margin: auto;
  background-color: #1e1e1e;
  padding: 25px;
  border-radius: 16px;
  box-shadow: 0 0 15px rgba(0, 188, 212, 0.15);
}

h1, h2 {
  text-align: center;
  color: #00bcd4;
  margin-bottom: 20px;
}
textarea {
  width: 100%;
  height: 100px;
  padding: 12px;
  border: 1px solid #2c2c2c;
  border-radius: 10px;
  font-size: 1rem;
  background: #121212;
```

Ln 17, Col 1 Spaces: 4 UTF-8 CRLF {} CSS Port : 5501

index.html | style.css | script.js

style.css > ...

```css
textarea {
  font-size: 1rem;
  background: #121212;
  color: #e0e0e0;
}

button {
  padding: 10px 20px;
  background-color: #00bcd4;
  color: #121212;
  border: none;
  border-radius: 10px;
  cursor: pointer;
  font-weight: bold;
  margin-top: 10px;
  transition: background-color 0.3s ease;
}
button:hover {
  background-color: #0097a7;
}

.note {
  background-color: #232323;
  border-left: 6px solid #00bcd4;
  padding: 12px 16px;
  margin: 10px 0;
  border-radius: 10px;
  position: relative;
  transition: background-color 0.3s ease;
}
```

Ln 17, Col 1 Spaces: 4 UTF-8 CRLF {} CSS Port : 5501

index.html # style.css JS script.js

style.css > ...

```css
.note {
}
.note:hover {
  background-color: #2a2a2a;
}

.delete-btn {
  position: absolute;
  right: 15px;
  top: 15px;
  background: #f44336;
  border: none;
  color: white;
  padding: 6px 12px;
  border-radius: 8px;
  cursor: pointer;
  font-size: 0.9rem;
}
.delete-btn:hover {
  background: #e53935;
}

#message {
  color: #4caf50;
  font-weight: bold;
  margin-top: 5px;
}
.hidden {
  display: none;
}
```

Ln 17, Col 1 Spaces: 4 UTF-8 CRLF {} CSS Port : 5501

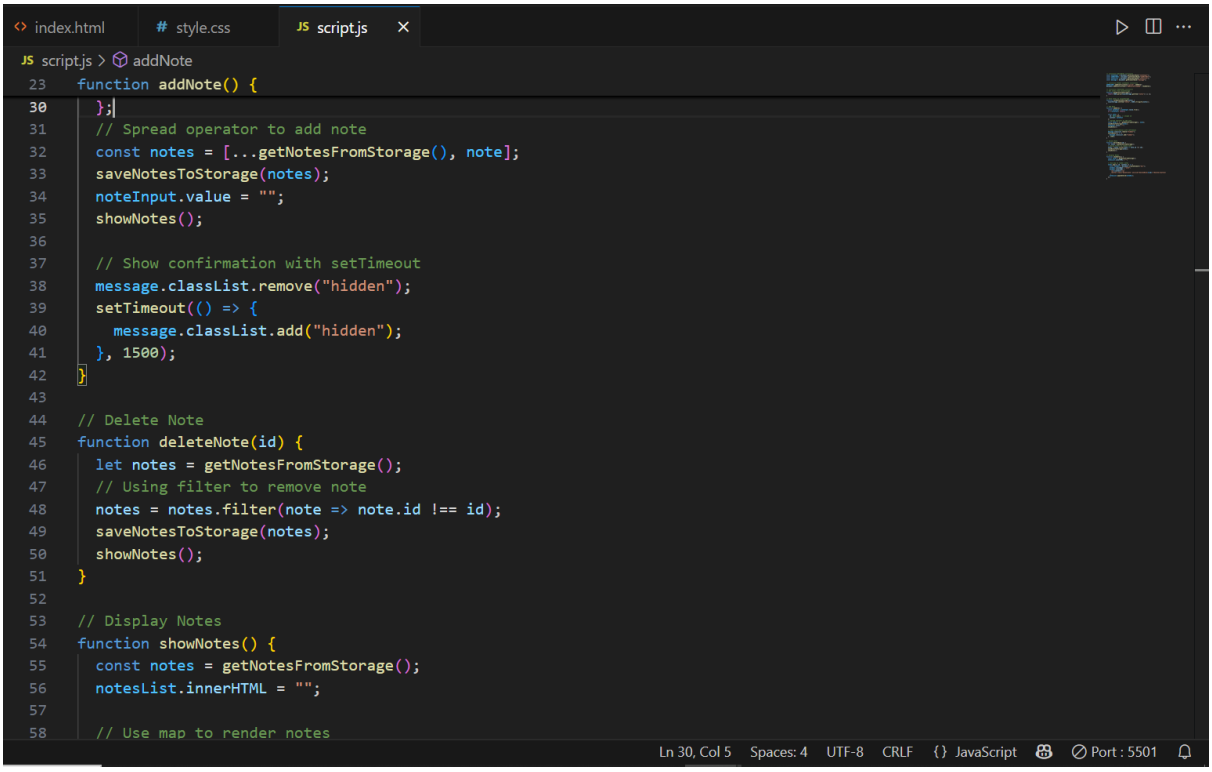
JavaScript:

index.html # style.css JS script.js

JS script.js > addNote

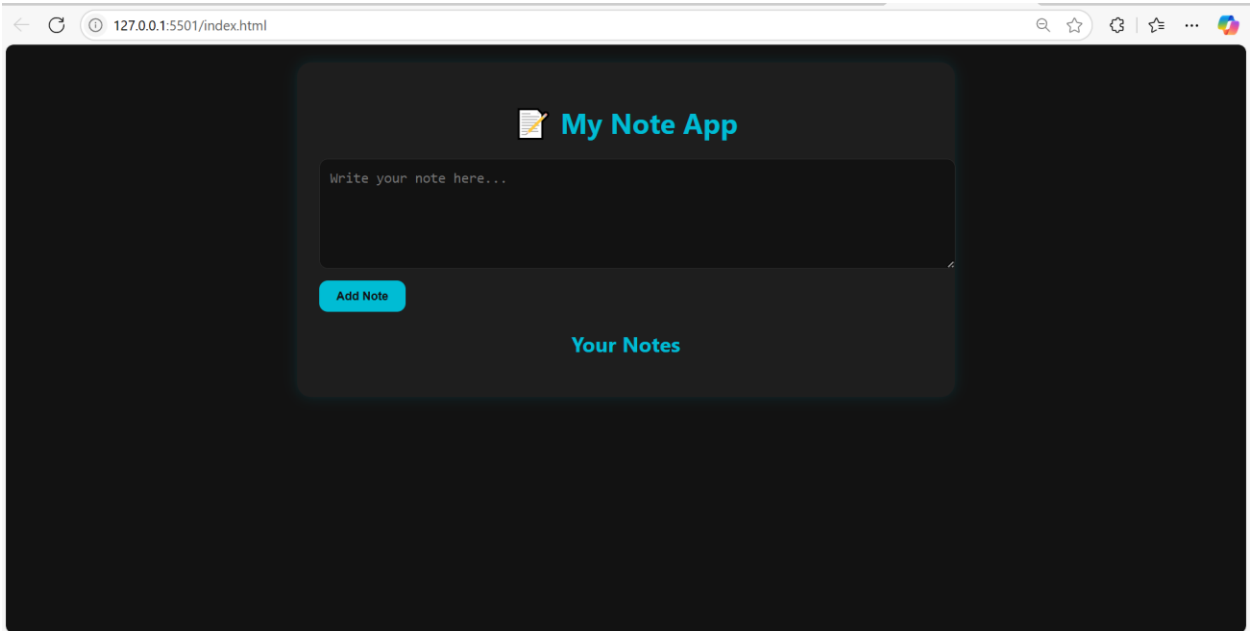
```javascript
// ========== Variables ==========
const noteInput = document.getElementById("noteInput");
const addNoteBtn = document.getElementById("addNoteBtn");
const notesList = document.getElementById("notesList");
const message = document.getElementById("message");

// ========== Event Listeners ==========
addNoteBtn.addEventListener("click", addNote);
document.addEventListener("DOMContentLoaded", showNotes);

// ========== Functions ==========
// Get notes from localStorage
function getNotesFromStorage() {
  return JSON.parse(localStorage.getItem("notes")) || [];
}

// Save notes to localStorage
function saveNotesToStorage(notes) {
  localStorage.setItem("notes", JSON.stringify(notes));
}

// Add Note
function addNote() {
  const noteText = noteInput.value.trim();
  if (!noteText) return;

  const note = {
    id: Date.now(), // unique id
    content: noteText
  };
```

Ln 26, Col 3 Spaces: 4 UTF-8 CRLF {} JavaScript Port : 5501

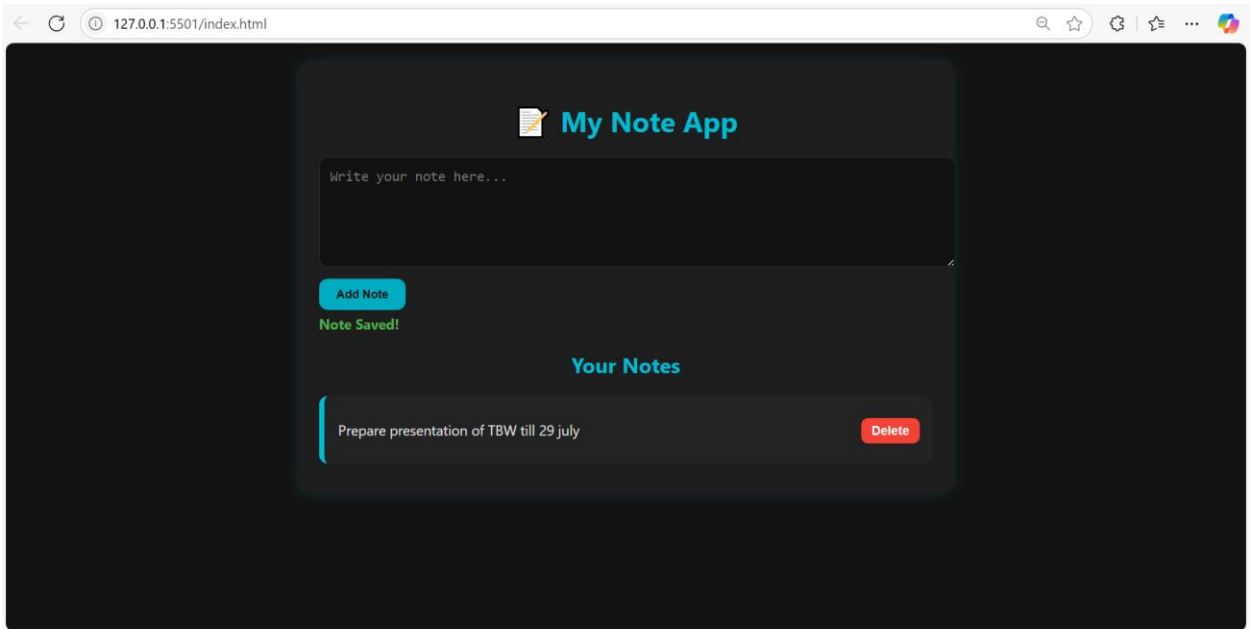

index.html
style.css
script.js
script.js > addNote
function addNote() {
};
// Spread operator to add note
const notes = [...getNotesFromStorage(), note];
saveNotesToStorage(notes);
noteInput.value = "";
showNotes();
// Show confirmation with setTimeout
message.classList.remove("hidden");
setTimeout(() => {
message.classList.add("hidden");
}, 1500);
}
// Delete Note
function deleteNote(id) {
let notes = getNotesFromStorage();
// Using filter to remove note
notes = notes.filter(note => note.id !== id);
saveNotesToStorage(notes);
showNotes();
}
// Display Notes
function showNotes() {
const notes = getNotesFromStorage();
notesList.innerHTML = "";
// Use map to render notes
Ln 30, Col 5
Spaces: 4
UTF-8
CRLF
{} JavaScript
Port : 5501


index.html
style.css
script.js
script.js > addNote
function deleteNote(id) {
}
// Display Notes
function showNotes() {
const notes = getNotesFromStorage();
notesList.innerHTML = "";
// Use map to render notes
notes.map(({ id, content }) => {
const noteDiv = document.createElement("div");
noteDiv.className = "note";
noteDiv.innerHTML = `
${content}
<button class="delete-btn" onclick="deleteNote(${id})">Delete</button>
`;
notesList.appendChild(noteDiv);
});
}

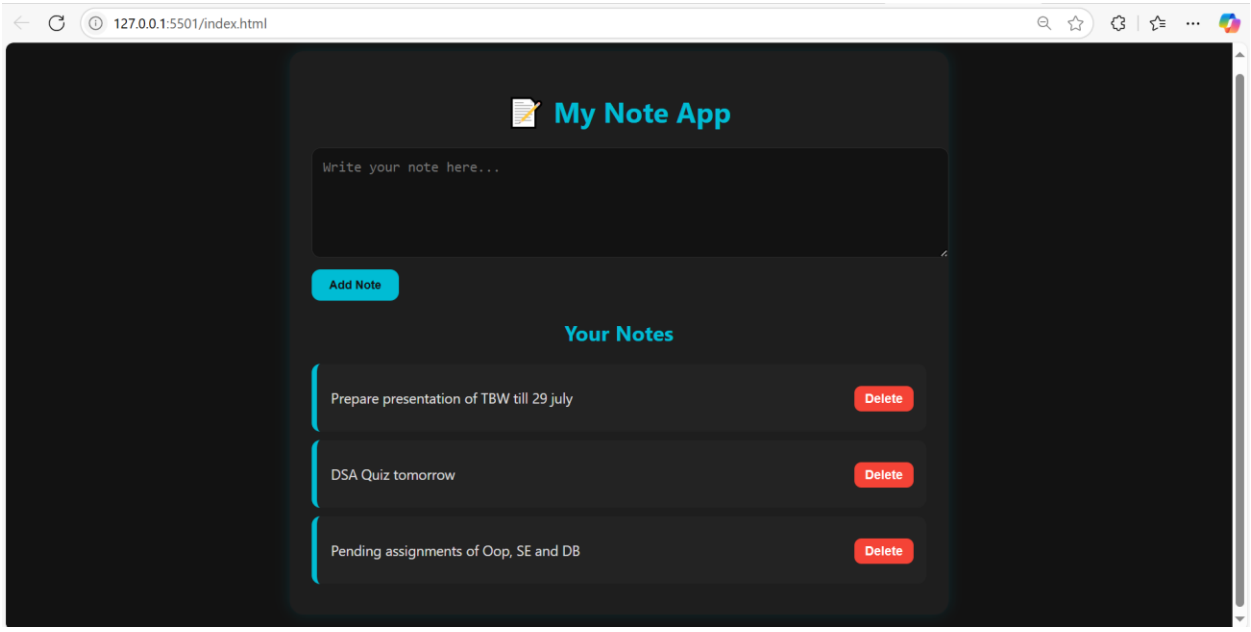
UI of Note Apps:



Adding notes:



Saved notes displayed:



Conclusion:

This mini-project helped me apply the core JavaScript concepts I learned throughout the week in a real-world scenario. I gained hands-on experience working with events, LocalStorage, ES6+ features, and DOM manipulation. The LocalStorage Note App is a useful and fully functional web application that reflects both my technical understanding and design skills.