INTERNSHIP REPORT

WEEK 5 DAY 5

Submitted to:	**Ali Hyder**	Submission Date:	**25th July, 2025**
Internship Domain:	**Front Development**	Internship Name:	**ProSensia**
Student Name:	**Yasal Qamar**	Roll No.	**S25031**

JavaScript Advanced

Topics: JavaScript ES6+ Concepts: Destructuring, Spread/Rest, Template Literals

Objective

To learn modern ES6+ JavaScript concepts that simplify code structure, improve readability, and provide more flexibility in handling arrays, objects, and strings.

Topics Covered

1. Destructuring Assignment

- Used to extract values from arrays or objects into separate variables.

Example (Array):

```
const fruits = ["Apple", "Banana", "Cherry"];
const [first, second] = fruits;
console.log(first); // Apple
console.log(second); // Banana
```

2. Spread Operator (...)

- Expands elements of an array or object into another array or object.
- Often used for copying, merging, or passing multiple arguments.

Example:

```
const arr1 = [1, 2, 3];
```

```
const arr2 = [4, 5, ...arr1];
console.log(arr2); // [4, 5, 1, 2, 3]
```

3. Rest Parameter (...)

- Collects multiple function arguments into a single array.

Example:

```
function sum(...numbers) {
  return numbers.reduce((total, num) => total + num, 0);
}
console.log(sum(5, 10, 15)); // 30
```

4. Template Literals

- Use backticks (`) for creating strings.
- Allows embedding variables and supports multi-line strings.

Example:

```
const name = "Yasal";
const greeting = `Hello, ${name}! Welcome to ES6.`;
console.log(greeting);
```

Skills Learned

- Efficiently extract values from arrays/objects with destructuring.
- Use spread/rest operators to handle dynamic data.
- Create cleaner strings with template literals.
- Write shorter, modern JavaScript code.

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>
```

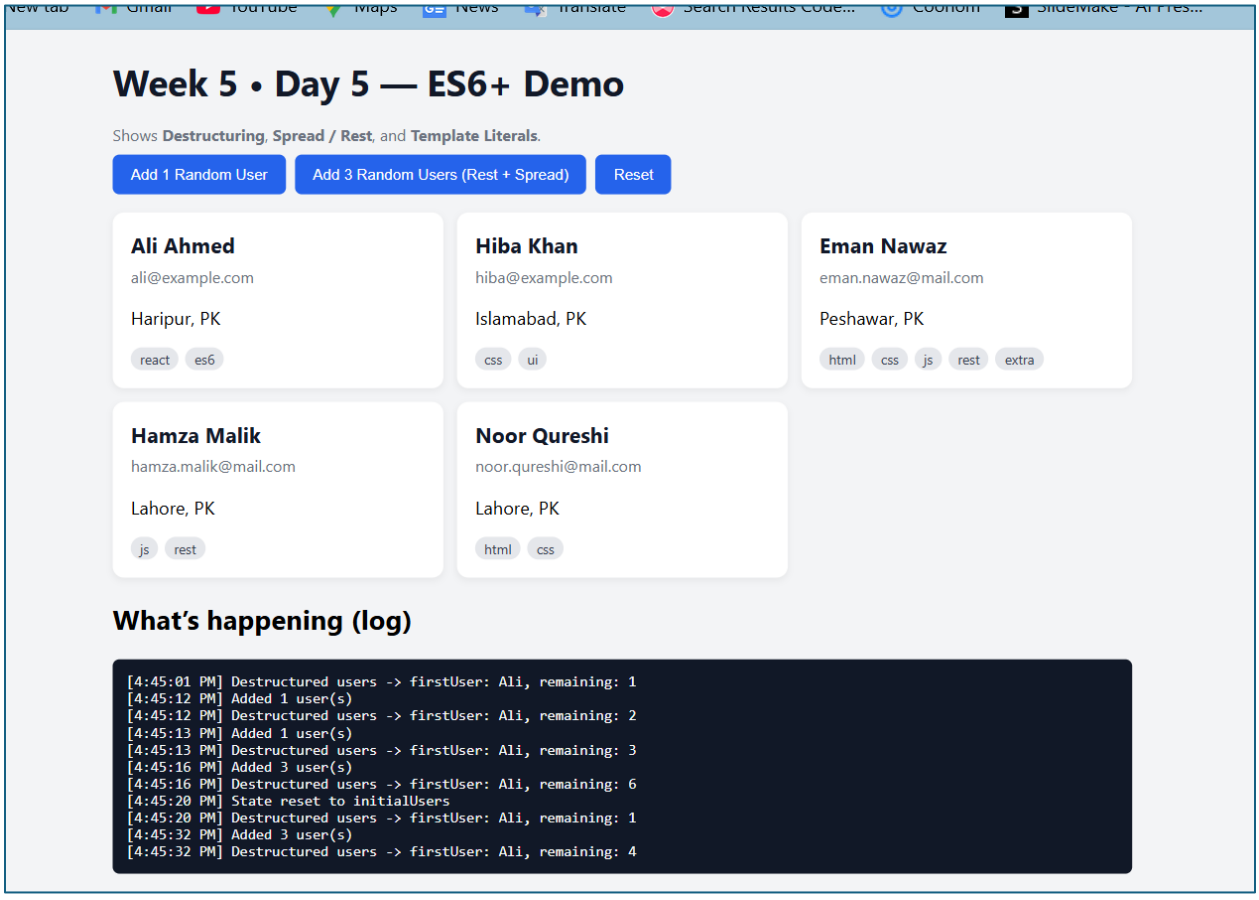
.js

```
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;
}
```



For practical implementation, I built a **To-Do List app** enhanced with ES6+ features and **localStorage** support.

- **CRUD Operations:** Add, edit, delete tasks.
- **Filters:** View All, Active, Completed tasks.
- **Dark Mode:** Toggle theme stored in localStorage.
- **ES6+ Concepts:** Destructuring, spread/rest operators, template literals, and arrow functions.

CONCUSLION:

In this task, I learned how **ES6 features** can simplify code, especially in **real-world applications like the To-Do list**.

- **Persistence:** localStorage ensures tasks stay saved after refresh.
- **Data Safety:** escapeHTML() prevents XSS or HTML injection.
- **Flexibility:** Spread/rest operators made array operations (add, merge) cleaner.
- **Code Readability:** Template literals helped in dynamic UI rendering.
