Advanced JavaScript Task Manager Application

(Features: CRUD operations, Local Storage, Asynchronous Data Fetching, Optimized Performance)

javascript

```
// taskManager.js - A Scalable Task Management System in JavaScript


class Task {
    /**
     * Represents a single task.
     * @param {string} title - The title of the task.
     * @param {string} description - The detailed description of the task.
     * @param {string} priority - Task priority: low, medium, or high.
     * @param {boolean} completed - Status of task completion.
     */
    constructor(title, description, priority = "medium", completed = false) {
        this.id = Task.generateId();
        this.title = title;
        this.description = description;
        this.priority = priority;
        this.completed = completed;
        this.createdAt = new Date().toISOString();
    }

    static generateId() {
        return `task-${Math.floor(Math.random() * 1000000)}`;
```

```
    }
}

class TaskManager {
    constructor() {
        this.tasks = this.loadTasks();
    }

    /**
     * Adds a new task to the list and saves to local storage.
     * @param {Task} task - The task object to add.
     */
    addTask(task) {
        if (!(task instanceof Task)) throw new Error("Invalid task format.");
        this.tasks.push(task);
        this.saveTasks();
    }

    /**
     * Retrieves all tasks, optionally filtering by completion status.
     * @param {boolean} [completed] - Optional filter for completed tasks.
     * @returns {Task[]} - List of tasks.
     */
    getTasks(completed = null) {
```

```
        return completed === null ? this.tasks : this.tasks.filter(task => task.completed === completed);
    }

    /**
     * Updates a task's status or priority.
     * @param {string} taskId - ID of the task to update.
     * @param {Object} updates - Fields to update.
     */
    updateTask(taskId, updates) {
        const task = this.tasks.find(t => t.id === taskId);
        if (!task) throw new Error("Task not found.");

        Object.assign(task, updates);
        this.saveTasks();
    }

    /**
     * Deletes a task by ID and updates local storage.
     * @param {string} taskId - ID of the task to delete.
     */
    deleteTask(taskId) {
        this.tasks = this.tasks.filter(t => t.id !== taskId);
        this.saveTasks();
    }
```

```
// Local Storage Handling

saveTasks() {
    localStorage.setItem("tasks", JSON.stringify(this.tasks));
}

loadTasks() {
    return JSON.parse(localStorage.getItem("tasks")) || [];
}

// Async Function: Simulated API Call

/**
 * Fetches a random task title from an API to demonstrate async programming.
 */
async fetchRandomTaskTitle() {
    try {
        const response = await fetch("https://jsonplaceholder.typicode.com/todos/1");
        const data = await response.json();
        return data.title;
    } catch (error) {
        console.error("Failed to fetch task title:", error);
    }
}
```

```
}

// □ Example Usage:

const taskManager = new TaskManager();

// Adding new tasks
const task1 = new Task("Learn JavaScript", "Master ES6+ concepts.", "high");
const task2 = new Task("Build a Web App", "Create a scalable project.", "medium");

taskManager.addTask(task1);
taskManager.addTask(task2);

console.log("All Tasks:", taskManager.getTasks());

// Updating a task
taskManager.updateTask(task1.id, { completed: true });

console.log("Completed Tasks:", taskManager.getTasks(true));

// Fetching a random task title (Async Example)
taskManager.fetchRandomTaskTitle().then(title => console.log("Random Task:", title));
```