פרויקט: אפליקציית ניהול משימות עם ייבוא נתונים מ-API

תיאור הפרויקט:

בנו אפליקציית ווב לניהול משימות. האפליקציה תאפשר למשתמשים להוסיף, להציג, לסמן כמשימות שהושלמו ולמחוק משימות. בנוסף, האפליקציה תשפר את הפונקציונליות על ידי טעינת משימות התחלתיות מ-API חיצוני.

מטרות הפרויקט:

- תרגול והבנה של HTML, CSS ו-JavaScript.
- העמקת הידע בטיפול (ב-DOM (Document Object Model.
- שימוש ב-(localStorage) Web Storage API לאחסון נתונים בצד הלקוח.
- הבנת עקרונות בסיסיים של שליחת בקשות HTTP וטיפול בתגובות (fetch API).
- תרגול עבודה עם נתונים בפורמט JSON.
- יישום תכונות מתקדמות כמו סינון ומיון נתונים.
- יצירת ממשק משתמש (UI) מודרני וידידותי.

דרישות מפורטות:

HTML

- צרו קובץ HTML עם הכותרת "Task Manager".
- השתמשו בתגיות HTML5 סמנטיות לדוגמה: `<header>`, `<main>`, `<footer>`
- הוסיפו כותרת ראשית (`<h1>`) לאפליקציה.
- צרו חלק (`<div>`) עבור קלט המשימות, הכולל:
  - שדה טקסט (`<input type="text">`) להזנת תיאור המשימה (עם placeholder מתאים).
  - שדה תאריך (`<input type="date">`) להזנת תאריך היעד של המשימה.
  - כפתור (`<button>`) להוספת המשימה.
- צרו חלק (`<div>`) עבור כפתורי סינון המשימות:
  - כפתור "הכל" - להצגת כל המשימות.
  - כפתור "הושלמו" - להצגת משימות שהושלמו בלבד.
  - כפתור "פעילות" - להצגת משימות שטרם הושלמו.
- צרו חלק (`<div>`) עבור כפתור מיון המשימות:
  - כפתור "מיין לפי תאריך יעד" - למיון המשימות לפי תאריך היעד.
- צרו רשימה לא מסודרת (`<ul>`) להצגת המשימות. המשימות יתווספו לרשימה זו באמצעות JavaScript.
- קשרו את קובץ CSS (`style.css`-ה) לעיצוב ואת קובץ JavaScript -ה (`script.js`)

CSS

- עצבו את האפליקציה במראה מודרני ונקי.
- השתמשו בגופן קריא (לדוגמה, 'Inter', 'Arial', 'sans-serif').
- הגדירו צבעי רקע, טקסט וגבולות מתאימים.
- השתמשו ב-Flexbox או Grid Layout לעימוד רכיבי הדף.
- הוסיפו ריווח (padding) וMargin מתאימים בין הרכיבים.
- עצבו את הכפתורים עם אפקטי ריחוף (hover) ולחיצה (active) נעימים.
- השתמשו במעברים (transitions) כדי ליצור אנימציות עדינות.
- התאימו את העיצוב לתצוגה רספונסיבית (Responsive Design) באמצעות Media Queries.

JavaScript

- **קבלת אלמנטים מה-DOM:**
  - השתמשו ב-`document.getElementById()` כדי לקבל הפניות לאלמנטים הרלוונטיים ב-HTML (שדות קלט, כפתורים, רשימה).
- **אחסון נתונים (localStorage):**
  - צרו פונקציות `getTasks()` ו-`saveTasks()` לטיפול באחסון המשימות ב-localStorage.
  - `getTasks()`:
    - נסו לקבל את הנתונים מה-localStorage באמצעות `localStorage.getItem('tasks')`.
    - אם קיימים נתונים, פרסו אותם מ-JSON באמצעות `JSON.parse()`.
    - אם אין נתונים, החזירו מערך ריק `[]`
  - `saveTasks(tasks)`:
    - קבלו את מערך המשימות כפרמטר.
    - המירו את המערך לפורמט JSON באמצעות `JSON.stringify()`.
    - שמרו את הנתונים ב-localStorage באמצעות `localStorage.setItem('tasks', jsonString)`.
- **מערך משימות גלובלי:**
  - השתמשו במשתנה גלובלי בשם `tasks` לאחסון מערך של אובייקטי משימות.
  - טענו את המשימות הקיימות מה-localStorage בעת טעינת הדף באמצעות `getTasks()`.
- **פונקציה** `renderTasks()`:
  - פונקציה זו אחראית לעדכון התצוגה של רשימת המשימות.
  - נקו את הרשימה הקיימת באמצעות `taskList.innerHTML = '';`.
  - סננו את המשימות בהתאם למסנן הנוכחי (ראו סעיף "סינון משימות").
  - עברו על כל משימה במערך המשימות המפורש באמצעות `forEach()`.
  - עבור כל משימה, צרו אלמנט `<li>` חדש.

- הגדירו את התוכן של ה-<li> כך שיציג את תיאור המשימה, תאריך היעד וכפתורי "השלמה" ו"מחיקה".
- אם המשימה הושלמה, הוסיפו לה קלאס CSS מתאים (לדוגמה, 'completed') לעיצוב הטקסט (קו חוצה).
- הוסיפו מאזיני אירועים (event listeners) לכפתורי "השלמה" ו"מחיקה" (ראו סעיפים "סימון משימה כהושלמה" ו"מחיקת משימה").
- הוסיפו את ה-<li> לרשימת המשימות (taskList).

- **פונקציה** :addTask()
  - פונקציה זו מטפלת בהוספת משימה חדשה.
  - קבלו את הערכים משדה הטקסט ותאריך היעד.
  - בדקו שהערכים תקינים (לא ריקים).
  - צרו אובייקט משימה חדש עם המאפיינים הבאים:
    - text: תיאור המשימה.
    - dueDate: תאריך היעד.
    - completed: ערך בוליאני (true/false) המציין האם המשימה הושלמה (false כברירת מחדל).
  - הוסיפו את המשימה החדשה למערך ה-tasks.
  - שמרו את המערך המעודכן ב-localStorage באמצעות saveTasks().
  - רנדרו מחדש את רשימת המשימות באמצעות renderTasks().
  - נקו את שדות הקלט.
- **פונקציה** :filterTasks(tasks, filter)
  - פונקציה זו מקבלת מערך של משימות ומחזירה מערך מסונן בהתאם לפרמטר filter.
  - השתמשו ב-switch או בשרשרת if...else if...else כדי לבדוק את ערך ה-filter:
    - 'all': החזירו את כל המשימות (המערך המקורי).
    - 'completed': החזירו רק את המשימות שהושלמו (השתמשו ב-filter() של מערך).
    - 'active': החזירו רק את המשימות שטרם הושלמו.
    - ברירת מחדל: החזירו את כל המשימות.
- **פונקציה** :sortTasks(tasks)

- פונקציה זו מקבלת מערך של משימות ומחזירה מערך ממוין לפי תאריך היעד.
  השתמשו בפונקציה sort() של מערך.

- **טיפול באירועים:**
  - כפתור "הוסף משימה":
    - הוסיפו מאזין אירוע click לכפתור.
    - כאשר הכפתור נלחץ, קראו לפונקציה addTask().
  - כפתורי סינון:
    - הוסיפו מאזיני אירוע click לכל כפתורי הסינון.
    - כאשר כפתור מסוים נלחץ, עדכנו את המסנן הגלובלי (currentFilter) וקראו לפונקציה renderTasks().
  - כפתור "מיין לפי תאריך יעד":
    - הוסיפו מאזין אירוע click לכפתור.
    - כאשר הכפתור נלחץ, קראו לפונקציה sortTasks()
    - שמרו את המערך הממוין ב localStorage
    - רנדרו את המשימות.
  - כפתורי "השלמה" ו"מחיקה":
    - הוסיפו מאזיני אירועים לכפתורים אלו דינמית, כאשר אתם יוצרים את רשימת המשימות בתוך renderTasks().
    - השתמשו ב-event.target.dataset.id כדי לקבל את ה-ID של המשימה שאליה משויך הכפתור.
    - כפתור "השלמה":
      - כאשר נלחץ, הפכו את סטטוס ההשלמה של המשימה (true/false).
      - שמרו את המערך המעודכן ב-localStorage.
      - רנדרו מחדש את רשימת המשימות.
    - כפתור "מחיקה":
      - כאשר נלחץ, מחקו את המשימה מהמערך.
      - שמרו את המערך המעודכן ב-localStorage.
      - רנדרו מחדש את רשימת המשימות.

- **טעינת משימות התחלתיות מ-API:**
  - צרו פונקציה אסינכרונית בשם `fetchInitialTasks()`.
  - השתמשו ב-`fetch()` כדי לשלוח בקשת GET ל-API חיצוני (לדוגמה, 'https://jsonplaceholder.typicode.com/todos?_limit=5' - להבאת 5 משימות).
  - ודאו שהתגובה תקינה (200 status) לפני שממשיכים.
  - חלצו את הנתונים מהתגובה באמצעות `response.json()`.
  - עברו על הנתונים שהתקבלו והמירו אותם למבנה אובייקט המשימה בו משתמשת האפליקציה (text, dueDate, completed).
  - הוסיפו את המשימות שהתקבלו למערך ה-`tasks`.
  - שמרו את המשימות ב localStorage
  - קראו לפונקציה renderTasks()
  - קראו לפונקציה `fetchInitialTasks()` כאשר הדף נטען.

הערות נוספות:

- הקפידו על כתיבת קוד ברורה ומתועדת היטב באמצעות הערות (comments).
- חלקו את הקוד לפונקציות קטנות ומוגדרות היטב לשיפור הקריאות והתחזוקה.
- השתמשו בשמות משמעותיים למשתנים ופונקציות.
- טפלו בשגיאות בצורה נכונה (לדוגמה, הצגת הודעה למשתמש במקרה של כישלון בטעינת נתונים מה-API).
- בדקו את האפליקציה בדפדפנים שונים כדי לוודא תאימות.

טיפים:

- התחילו עם המבנה הבסיסי של HTML ו-CSS לפני שאתם מתחילים לכתוב JavaScript.
- כתבו את הקוד שלכם בהדרגה ובדקו אותו בכל שלב.
- השתמשו בכלי הפיתוח של הדפדפן (Developer Tools) כדי לבדוק את ה- HTML, CSS וה-JavaScript, ולזהות שגיאות.
- אל תהססו לשאול שאלות ולהיעזר בחברים או במדריך.
- הכי חשוב - תהנו מהתהליך!