<!DOCTYPE html>

<!-- Coding By CodingNepal - youtube.com/codingnepal -->

<html lang="en" dir="ltr">

<head>

<meta charset="utf-8">

<title>Todo List App in JavaScript | CodingNepal</title>

<link rel="stylesheet" href="style.css">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<!-- Iconscout Link For Icons -->

<link rel="stylesheet" href="https://unicons.iconscout.com/release/v4.0.0/css/line.css">

</head>

<body>

<div class="wrapper">

<div class="task-input">

<img src="bars-icon.svg" alt="icon">

<input type="text" placeholder="Add a new task">

</div>

<div class="controls">

<div class="filters">

<span class="active" id="all">All</span>

<span id="pending">Pending</span>

<span id="completed">Completed</span>

</div>

<button class="clear-btn">Clear All</button>

</div>

<ul class="task-box"></ul>

</div>

<script src="script.js"></script>

</body>

</html>

/\* Import Google Font - Poppins \*/

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;500;600;700&display=swap');

\*{

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Poppins', sans-serif;

}

body{

width: 100%;

height: 100vh;

overflow: hidden;

background: linear-gradient(135deg, #4AB1FF, #2D5CFE);

}

::selection{

color: #fff;

background: #3C87FF;

}

.wrapper{

max-width: 405px;

padding: 28px 0 30px;

margin: 137px auto;

background: #fff;

border-radius: 7px;

box-shadow: 0 10px 30px rgba(0,0,0,0.1);

}

.task-input{

height: 52px;

padding: 0 25px;

position: relative;

}

.task-input img{

top: 50%;

position: absolute;

transform: translate(17px, -50%);

}

.task-input input{

height: 100%;

width: 100%;

outline: none;

font-size: 18px;

border-radius: 5px;

padding: 0 20px 0 53px;

border: 1px solid #999;

}

.task-input input:focus,

.task-input input.active{

padding-left: 52px;

border: 2px solid #3C87FF;

}

.task-input input::placeholder{

color: #bfbfbf;

}

.controls, li{

display: flex;

align-items: center;

justify-content: space-between;

}

.controls{

padding: 18px 25px;

border-bottom: 1px solid #ccc;

}

.filters span{

margin: 0 8px;

font-size: 17px;

color: #444444;

cursor: pointer;

}

.filters span:first-child{

margin-left: 0;

}

.filters span.active{

color: #3C87FF;

}

.controls .clear-btn{

border: none;

opacity: 0.6;

outline: none;

color: #fff;

cursor: pointer;

font-size: 13px;

padding: 7px 13px;

border-radius: 4px;

letter-spacing: 0.3px;

pointer-events: none;

transition: transform 0.25s ease;

background: linear-gradient(135deg, #1798fb 0%, #2D5CFE 100%);

}

.clear-btn.active{

opacity: 0.9;

pointer-events: auto;

}

.clear-btn:active{

transform: scale(0.93);

}

.task-box{

margin-top: 20px;

margin-right: 5px;

padding: 0 20px 10px 25px;

}

.task-box.overflow{

overflow-y: auto;

max-height: 300px;

}

.task-box::-webkit-scrollbar{

width: 5px;

}

.task-box::-webkit-scrollbar-track{

background: #f1f1f1;

border-radius: 25px;

}

.task-box::-webkit-scrollbar-thumb{

background: #e6e6e6;

border-radius: 25px;

}

.task-box .task{

list-style: none;

font-size: 17px;

margin-bottom: 18px;

padding-bottom: 16px;

align-items: flex-start;

border-bottom: 1px solid #ccc;

}

.task-box .task:last-child{

margin-bottom: 0;

border-bottom: 0;

padding-bottom: 0;

}

.task-box .task label{

display: flex;

align-items: flex-start;

}

.task label input{

margin-top: 7px;

accent-color: #3C87FF;

}

.task label p{

user-select: none;

margin-left: 12px;

word-wrap: break-word;

}

.task label p.checked{

text-decoration: line-through;

}

.task-box .settings{

position: relative;

}

.settings :where(i, li){

cursor: pointer;

}

.settings .task-menu{

z-index: 10;

right: -5px;

bottom: -65px;

padding: 5px 0;

background: #fff;

position: absolute;

border-radius: 4px;

transform: scale(0);

transform-origin: top right;

box-shadow: 0 0 6px rgba(0,0,0,0.15);

transition: transform 0.2s ease;

}

.task-box .task:last-child .task-menu{

bottom: 0;

transform-origin: bottom right;

}

.task-box .task:first-child .task-menu{

bottom: -65px;

transform-origin: top right;

}

.task-menu.show{

transform: scale(1);

}

.task-menu li{

height: 25px;

font-size: 16px;

margin-bottom: 2px;

padding: 17px 15px;

cursor: pointer;

justify-content: flex-start;

}

.task-menu li:last-child{

margin-bottom: 0;

}

.settings li:hover{

background: #f5f5f5;

}

.settings li i{

padding-right: 8px;

}

@media (max-width: 400px) {

body{

padding: 0 10px;

}

.wrapper {

padding: 20px 0;

}

.filters span{

margin: 0 5px;

}

.task-input{

padding: 0 20px;

}

.controls{

padding: 18px 20px;

}

.task-box{

margin-top: 20px;

margin-right: 5px;

padding: 0 15px 10px 20px;

}

.task label input{

margin-top: 4px;

}

}

const taskInput = document.querySelector(".task-input input"),

filters = document.querySelectorAll(".filters span"),

clearAll = document.querySelector(".clear-btn"),

taskBox = document.querySelector(".task-box");

// Initialize variables

let editId,

isEditTask = false,

todos = JSON.parse(localStorage.getItem("todo-list"));

// Event listeners for filter buttons

filters.forEach(btn => {

btn.addEventListener("click", () => {

document.querySelector("span.active").classList.remove("active");

btn.classList.add("active");

showTodo(btn.id);

});

});

// Function to display tasks based on filter

function showTodo(filter) {

let liTag = "";

if (todos) {

todos.forEach((todo, id) => {

// Determine if task is completed or pending

let completed = todo.status == "completed" ? "checked" : "";

if (filter == todo.status || filter == "all") {

// Create task list item

liTag += `<li class="task">

<label for="${id}">

<input onclick="updateStatus(this)" type="checkbox" id="${id}" ${completed}>

<p class="${completed}">${todo.name}</p>

</label>

<div class="settings">

<i onclick="showMenu(this)" class="uil uil-ellipsis-h"></i>

<ul class="task-menu">

<li onclick='editTask(${id}, "${todo.name}")'><i class="uil uil-pen"></i>Edit</li>

<li onclick='deleteTask(${id}, "${filter}")'><i class="uil uil-trash"></i>Delete</li>

</ul>

</div>

</li>`;

}

});

}

// Update task box with tasks or a message if no tasks

taskBox.innerHTML = liTag || `<span>You don't have any task here</span>`;

let checkTask = taskBox.querySelectorAll(".task");

// Toggle clear all button based on task presence

!checkTask.length ? clearAll.classList.remove("active") : clearAll.classList.add("active");

// Add or remove overflow class based on task box height

taskBox.offsetHeight >= 300 ? taskBox.classList.add("overflow") : taskBox.classList.remove("overflow");

}

showTodo("all");

// Function to display menu for task actions

function showMenu(selectedTask) {

let menuDiv = selectedTask.parentElement.lastElementChild;

menuDiv.classList.add("show");

// Hide menu if clicked outside

document.addEventListener("click", e => {

if (e.target.tagName != "I" || e.target != selectedTask) {

menuDiv.classList.remove("show");

}

});

}

// Function to update task status (completed or pending)

function updateStatus(selectedTask) {

let taskName = selectedTask.parentElement.lastElementChild;

if (selectedTask.checked) {

taskName.classList.add("checked");

todos[selectedTask.id].status = "completed";

} else {

taskName.classList.remove("checked");

todos[selectedTask.id].status = "pending";

}

localStorage.setItem("todo-list", JSON.stringify(todos))

}

// Function to prepare for editing a task

function editTask(taskId, textName) {

editId = taskId;

isEditTask = true;

taskInput.value = textName;

taskInput.focus();

taskInput.classList.add("active");

}

// Function to delete a task

function deleteTask(deleteId, filter) {

isEditTask = false;

todos.splice(deleteId, 1);

localStorage.setItem("todo-list", JSON.stringify(todos));

showTodo(filter);

}

// Event listener for clear all button

clearAll.addEventListener("click", () => {

isEditTask = false;

todos.splice(0, todos.length);

localStorage.setItem("todo-list", JSON.stringify(todos));

showTodo()

});

// Event listener for task input

taskInput.addEventListener("keyup", e => {

let userTask = taskInput.value.trim();

if (e.key == "Enter" && userTask) {

if (!isEditTask) {

todos = !todos ? [] : todos;

let taskInfo = { name: userTask, status: "pending" };

todos.push(taskInfo);

} else {

isEditTask = false;

todos[editId].name = userTask;

}

taskInput.value = "";

localStorage.setItem("todo-list", JSON.stringify(todos));

showTodo(document.querySelector("span.active").id);

}

});