

**CODTECH IT SOLUTIONS**

**Title: Comprehensive Documentation on TO-DO List using HTML, CSS, and JavaScript**

**Table of Contents:**

* Introduction

The purpose of this documentation is to provide a detailed guide on creating a TO-DO list application using HTML, CSS, and JavaScript. The TO-DO list allows users to manage their tasks and keep track of their progress. Throughout this documentation, we will cover the step-by-step process of building the application, starting from the basic HTML structure to incorporating CSS for styling and JavaScript for interactivity.  
A to-do list is a simple but powerful tool for managing tasks and goals. It is a list of tasks or actions that need to be completed, often organized by priority or due date. The concept of a to-do list has been around for centuries, and it remains a popular method for staying organized and productive in both personal and professional settings.

The purpose of a to-do list is to help you remember and keep track of all the things you need to do, so you can focus on actually doing them. By writing down your tasks, you can free up mental space and reduce the cognitive load of trying to remember everything. This can help you to stay focused and avoid feeling overwhelmed.

To-do lists can be created using a variety of tools, from simple pen and paper to sophisticated digital apps. The key is to find a system that works for you and that you will actually use. Some people prefer to keep their lists simple and straightforward, while others like to add additional information and context to their tasks.

In the following sections, we will discuss some tips and best practices for creating and using to-do lists, as well as some common mistakes to avoid.

**Benefits of To-Do Lists**

There are many benefits to using to-do lists, including:

* **Memory aid:** To-do lists help you remember all the things you need to do, so you don't have to rely on your memory alone.
* **Focus and productivity:** By writing down your tasks, you can focus on actually completing them, rather than trying to remember what you need to do. This can help you to be more productive and efficient.
* **Goal setting:** To-do lists can help you to set and achieve your goals by breaking them down into smaller, manageable tasks.
* **Stress reduction:** By organizing your tasks and making a plan for completing them, you can reduce stress and anxiety.
* **Flexibility:** To-do lists can be easily adapted and updated as your priorities and tasks change.
* HTML Structure

HTML Code



* Adding Interactivity with JavaScript

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

const filters = document.querySelectorAll(".filters span");

const clearAll = document.querySelector(".clear-btn");

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

let editId;

let isEditTask = false

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

filters.forEach(btn => {

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

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

        btn.classList.add("active");

        showTodo(btn.id);

    });

});

function showTodo(filter) {

    let liTag = "";

    if(todos) {

        todos.forEach((todo, id) => {

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

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

                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>`;

            }

        });

    }

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

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

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

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

}

showTodo("all");

function showMenu(selectedTask) {

    let menuDiv = selectedTask.parentElement.lastElementChild;

    menuDiv.classList.add("show");

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

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

            menuDiv.classList.remove("show");

        }

    });

}

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 editTask(taskId, textName) {

    editId = taskId;

    isEditTask = true;

    taskInput.value = textName;

    taskInput.focus();

    taskInput.classList.add("active");

}

function deleteTask(deleteId, filter) {

    isEditTask = false;

    todos.splice(deleteId, 1);

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

    showTodo(filter);

}

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

    isEditTask = false;

    todos.splice(0, todos.length);

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

    showTodo()

});

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);

    }

});

This JavaScript code is for a todo list web app. It selects HTML elements using **document.querySelector** and **document.querySelectorAll**. It retrieves todos from local storage and adds event listeners to filter buttons, clear all button, and input field. It defines functions to show todos, update todo status, edit todo, delete todo, and clear all todos. It also adds event listeners to input field for adding new todos and editing existing todos. It updates local storage when todos are added, edited, or deleted. It toggles classes for styling and shows or hides elements based on user actions.

5. Conclusion

In the final section, we'll summarize the key points covered in this documentation. We'll emphasize the importance of understanding the HTML, CSS, and JavaScript concepts used in building the TO-DO list application. Additionally, we'll provide suggestions on how to expand and customize the application further, such as implementing additional features or integrating with backend technologies.

By following the steps outlined in this documentation, readers will gain a comprehensive understanding of building a TO-DO list application using HTML, CSS, and JavaScript. They will be equipped with the knowledge required to create their own personalized versions of the TO-DO list and apply the learned concepts to future web development projects.

Note: This documentation provides a high-level overview of the TO-DO list application. For a more detailed implementation, including code snippets and examples, please refer to the accompanying code repository or tutorial materials.