**TITLE: CodTech IT Solutions Internship - Task Documentation:**

**“To-DO LIST” Using CSS, HTML, JAVASCRIPT”.**

**INTERN INFORMATION:**

**Name:** CHETAN N KALLIMANI

**ID:** C0D7068

**INTRODUCTION**

In the realm of personal productivity and organizational tools, the to-do list occupies a central role, universally recognized for its simplicity and effectiveness in managing tasks and priorities. The evolution from paper-based lists to digital platforms has significantly expanded the functionality and accessibility of to-do lists, making them an indispensable tool for individuals looking to optimize their time and manage their responsibilities efficiently. This project aims to further innovate in this space by developing a To-Do List application utilizing the power of modern web technologies: JavaScript, HTML, and CSS.

The transition to digital to-do lists has opened a plethora of possibilities for customization, real-time collaboration, and accessibility across devices. Despite the availability of numerous applications in the market, there remains a considerable opportunity to create a more intuitive, user-friendly, and flexible tool that caters to the varied needs of users. This project is driven by the vision to harness the capabilities of JavaScript and the versatility of web technologies to deliver superior task management experience.

**Implementation**

* JavaScript Framework: Utilize a modern JavaScript framework for building the frontend application.
* HTML/CSS: Use HTML5 and CSS3 for structuring and styling the user interface, ensuring compatibility with various web browsers.
* Responsive Design: Implement responsive design principles to ensure optimal viewing experience across desktop and mobile devices.
* User Interface Components: Utilize UI libraries for designing interactive and visually appealing components.

**CODE: -**

**HTML: -**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

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

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

    <title>TO-DO LIST</title>

</head>

<body>

    <div class="container">

        <div class="box">

            <h2>To Do List <br> <span> By </span></h2>

            <input type="text" placeholder="Write here..." id="mainBx">

            <ul id="list"></ul>

        </div>

    </div>

</body>

</html>

**CSS:-**

@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&display=swap');

\*{

    margin: 0;

    padding: 0;

    box-sizing: border-box;

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

}

body{

    display: grid;

    justify-content: center;

    align-items: center;

    min-height: 100vh;

    background: #8d99ae;

}

.box{

    position: relative;

    width: 450px;

    height: 550px;

    background: #8d99ae;

    padding: 30px 50px;

    box-shadow: rgba(159, 204, 244, 0.48) 6px 2px 16px 0px, rgba(214, 239, 255, 0.8) -6px -2px 16px 0px;

}

.box h2{

    width: 100%;

    font-weight: 600;

    text-align: center;

    color: #2b2d42;

    font-size: 1.75em;

    margin-bottom: 10px;

    text-transform: uppercase;

}

.box span{

    color: #fff;

}

#mainBx{

    position: relative;

    width: 100%;

    border: none;

    outline: none;

    padding: 10px 20px;

    border-radius: 30px;

    font-size: 1em;

    color: #2f363e;

    font-weight: 500;

    box-shadow: rgba(50, 50, 93, 0.25) 0px 30px 60px -12px inset, rgba(0, 0, 0, 0.3) 0px 18px 36px -18px inset;

}

#mainBx::placeholder{

    color: #2f363e;

    font-weight: 500;

}

.box li{

    position: relative;

    width: calc(100% - 40px);

    display: grid;

    align-items: center;

    justify-content: space-between;

    background: #9fffcb;

    margin: 15px 0;

    padding: 10px 10px 10px 45px;

    min-height: 35px;

    cursor: pointer;

    border-radius: 22.5px;

    color: #fff;

    box-shadow: rgba(0, 0, 0, 0.17) 0px -23px 25px 0px inset, rgba(0, 0, 0, 0.15) 0px -36px 30px 0px inset, rgba(0, 0, 0, 0.1) 0px -79px 40px 0px inset, rgba(0, 0, 0, 0.06) 0px 2px 1px, rgba(0, 0, 0, 0.09) 0px 4px 2px, rgba(0, 0, 0, 0.09) 0px 4px 2px, rgba(0, 0, 0, 0.09) 0px 6px 4px, rgba(0, 0, 0, 0.09) 0px 10px 6px;

}

.box li i{

    position: absolute;

    right: -40px;

    width: 30px;

    height: 30px;

    display: flex;

    justify-content: center;

    align-items: center;

    background: #ff2c74;

    border-radius: 50%;

    box-shadow: rgba(0, 0, 0, 0.17) 0px -23px 25px 0px inset, rgba(0, 0, 0, 0.15) 0px -36px 30px 0px inset, rgba(0, 0, 0, 0.1) 0px -79px 40px 0px inset, rgba(0, 0, 0, 0.06) 0px 2px 1px, rgba(0, 0, 0, 0.09) 0px 4px 2px, rgba(0, 0, 0, 0.09) 0px 8px 4px, rgba(255, 255, 255, 0.5) 2px 2px 10px, rgba(0, 0, 0, 0.09) 0px 32px 16px;

}

.box li i::before{

    content: '';

    position: absolute;

    width: 15px;

    height: 2px;

    background: #fff;

    transform: rotate(45deg);

}

.box li i::after{

    content: '';

    position: absolute;

    width: 15px;

    height: 2px;

    background: #fff;

    transform: rotate(-45deg);

}

.box li::before{

    content: '';

    position: absolute;

    left: 8px;

    width: 30px;

    height: 30px;

    background: #2f363e;

    border-radius: 50%;

}

.box li.done{

    background: #607d8b;

}

.box li.done::after{

    content: '';

    position: absolute;

    left: 16px;

    bottom: 19px;

    width: 12px;

    height: 6px;

    border-bottom: 2px solid #fff;

    border-left: 2px solid #fff;

    transform: rotate(315deg);

}

#list{

    position: relative;

    margin-top: 20px;

    height: 350px;

    overflow-y: auto;

    padding-right: 10px;

}

::-webkit-scrollbar-track{

    box-shadow: inset 0 0 15px rgba(0, 0, 0, 0.5);

}

::-webkit-scrollbar{

    width: 5px;

}

::-webkit-scrollbar-thumb{

    background: #fff;}

**JS: -**

let mainBx = document.querySelector('#mainBx');

let list = document.querySelector('#list');

mainBx.addEventListener("keyup", function(event){

    if(event.key == "Enter"){

        addItem(this.value)

        this.value = ""

    }

})

let addItem = (mainBx) => {

    let listItem = document.createElement("li");

    listItem.innerHTML = `${mainBx}<i></i>`;

    listItem.addEventListener("click", function(){

        this.classList.toggle('done');

    })

    listItem.querySelector("i").addEventListener("click", function(){

        listItem.remove();

    })

    list.appendChild(list Item);

}

**CODE EXPLAINATION**

**HTML Structure:**

1. <!DOCTYPE html>: Declares the document type and version of HTML being used.
2. <html lang="en">: Specifies the language of the document (English).
3. <head>: Contains meta-information about the HTML document, such as character encoding, viewport settings, and links to external resources.
4. <meta charset="UTF-8">: Specifies the character encoding of the document as UTF-8, which supports a wide range of characters.
5. <meta name="viewport" content="width=device-width, initial-scale=1.0">: Sets the viewport properties for responsive design.
6. <script src="./script.js" defer></script>: Links an external JavaScript file named "script.js" and specifies the "defer" attribute to defer script execution until after the document has been parsed.
7. <link rel="stylesheet" href="./style.css">: Links an external CSS stylesheet named "style.css" to style the HTML elements.
8. <title>TO-DO LIST</title>: Sets the title of the HTML document, which appears in the browser's title bar or tab.
9. <body>: Contains the visible content of the HTML document.
10. <div class="container">: Acts as a container for the main content of the page.
11. <div class="box">: Contains the main content of the to-do list application.
12. <h2>To Do List <br> <span> By CHETAN </span></h2>: Displays the heading of the to-do list, with a subheading indicating the author's name.
13. <input type="text" placeholder="Write here..." id="mainBx">: Provides an input field where users can type their to-do items.
14. <ul id="list"></ul>: Defines an unordered list element where the to-do items will be displayed.
15. This HTML code sets up the basic structure of a to-do list web application, with placeholders for adding tasks and displaying them in a list format. The styling and interactivity (such as adding and removing tasks) would typically be handled by the linked CSS and JavaScript files, respectively.

**CSS Styling:**

1. The below code is a CSS code that styles the layout and appearance of a To-Do list application. The code is organized into several sections based on the HTML elements being styled. Here's a breakdown of the code:
2. Importing Font:
3. @import url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&display=swap');: This line imports the Google Font "Poppins" with various weights to be used throughout the document.
4. Global Styles:
5. \*: Selects all elements on the page.
6. margin: 0; padding: 0; box-sizing: border-box;: Resets margins and padding to zero and sets box-sizing to border-box for consistent layout calculations.
7. font-family: 'Poppins', sans-serif;: Sets the default font family to "Poppins" with a fallback to the sans-serif font.
8. Body Styles:
9. body: Selects the body element.
10. display: grid; justify-content: center; align-items: center;: Sets the body to a grid layout with content centered horizontally and vertically.
11. min-height: 100vh;: Sets the minimum height of the body to 100% of the viewport height.
12. background: #8d99ae;: Sets the background color of the body to a light blue-gray.
13. Container Styles:
14. .container: Selects elements with the class "container".
15. Defines styles for a container element, but it's not explicitly defined in the provided HTML.
16. Box Styles:
17. . box: Selects elements with the class "box".
18. Defines styles for a box element containing the to-do list.
19. Sets width, height, background color, padding, and box shadow.
20. Styles the heading and input field inside the box.
21. List Item Styles:
22. . box li: Selects list items inside the box.
23. Styles each list item, including background color, margin, padding, cursor, border radius, color, and box shadow.
24. . box li.done: Selects list items with the class "done".
25. Styles completed list items differently, with a different background color and a checkmark symbol.
26. Scrollbar Styles:
27. ::-webkit-scrollbar-track, ::-webkit-scrollbar, ::-webkit-scrollbar-thumb: Styles the scrollbar appearance in WebKit browsers like Chrome and Safari.
28. These CSS styles define the appearance and layout of various elements within the to-do list application, including the container, box, input field, list items, and scrollbar. The styles aim to create a visually appealing and user-friendly interface for managing tasks.

**JavaScript Functionality:**

This JavaScript code is for a To-Do list application that allows users to add, check off, and delete tasks. Here's a breakdown of the code:

1. Variable Declarations:

let mainBx = document.querySelector('#mainBx');: Selects the input field with the id "mainBx" and assigns it to the variable mainBx.

let list = document.querySelector('#list');: Selects the unordered list with the id "list" and assigns it to the variable list.

2 Event Listener for Keyup:

mainBx.addEventListener("keyup", function(event){ ... });: Listens for keyup events on the mainBx input field.

Inside the event listener function:

if (event.key == "Enter"){ ... }: Checks if the key pressed is the Enter key.

addItem(this.value): Calls the addItem function with the value of the input field as an argument.

this.value = "": Clears the input field after adding the item.

3. add Item Function:

let add Item = (mainBx) => {...}: Defines a function named addItem that takes a parametermainBx,

which represents the text of the to-do item.

Inside the function:

let listItem = document.createElement("li");: Creates a new list item element.

listItem.innerHTML = ${mainBx}<i></i>; Sets the inner HTML of the list item, including the text of the to-do item and an empty <i> element for a checkbox.

listItem.addEventListener("click", function (){ ... });: Listens for click events on the list item.

this.classList.toggle('done');: Toggles the 'done' class on the list item, indicating whether the task is completed or not.

listItem.querySelector("i").addEventListener("click", function(){ ... });: Listens for click events on the <i> element (checkbox).

listItem.remove(); Removes the list item from the DOM when the checkbox is clicked.

list.appendChild(listItem);: Appends the list item to the unordered list.

This JavaScript code sets up functionality for adding new to-do items to the list, marking items as done when clicked, and removing items when their checkboxes are clicked. It enhances the interactivity and functionality of the to-do list application.

**USAGE**

1. A to do list application is a simple yet powerful tool that can help individuals and organizations manage their tasks and workflow more efficiently. Here are some of the ways a to do list application can be used:
2. Personal task management: a to do list application can be used to manage personal tasks such as grocery shopping, household chores, and exercise routines. By creating a list of tasks, individuals can prioritize and manage their time more effectively, reducing stress and increasing productivity.
3. Project management: in a project management context, a to do list application can be used to break down complex projects into smaller, manageable tasks. This allows team members to track their progress and ensure that all aspects of the project are completed on time.
4. Collaboration: a to do list application can be used as a collaboration tool, allowing team members to assign tasks to one another and track their progress. This can help ensure that everyone is on the same page and working towards the same goals.
5. Time management: a to do list application can help individuals and organizations manage their time more effectively. By prioritizing tasks and allocating time slots for each task, individuals can ensure that they are making the most of their time and avoiding procrastination.
6. Goal setting: a to do list application can be used to set and track goals. By breaking down long-term goals into smaller, manageable tasks, individuals can track their progress and stay motivated.
7. Prioritization: a to do list application can help individuals and organizations prioritize their tasks. By identifying the most important tasks and allocating resources accordingly, individuals and organizations can ensure that they are focusing on the tasks that will have the greatest impact.
8. Stress reduction: by providing a clear overview of tasks and deadlines, a to do list application can help reduce stress and anxiety. Individuals can focus on one task at a time, knowing that they have a system in place to manage their workload.
9. In summary, a to do list application is a versatile tool that can be used in a variety of contexts to manage tasks, collaborate with team members, manage time, set and track goals, prioritize tasks, and reduce stress.
10. Regarding the JavaScript code you provided, it is a simple to do list application that allows users to add tasks to a list, mark tasks as complete, and remove tasks from the list. The code uses the document object model (dome) to interact with the html elements, and the local storage Api to save and retrieve the list data.
11. The code initializes by selecting the input field, add button, and list element using the query selector method. An event listener is added to the add button, which triggers the create task function when clicked. The create task function creates a new list item, adds the user's input as the text content, and appends two span elements for the check and close buttons. The check and close buttons are given the appropriate classes for styling and functionality.
12. An event listener is added to the list element, which triggers a function when a list item is clicked. The function checks if the clicked element is the list item, check button, or close button, and performs the appropriate action (toggling the checked class, toggling the checked class of the parent list item, or removing the list item).
13. The save data function is called whenever a task is added, completed, or removed. The function saves the current state of the list to the local storage using the set item method. The show data function is called when the page loads, which retrieves the list data from the local storage and populates the list element.

Overall, the code is a simple yet effective implementation of a to do list application using JavaScript and the local storage Api.

**CONCLUSION: -**

In conclusion, the To Do List app is a simple yet powerful tool that can help individuals and organizations manage their tasks and workflow more efficiently. The app allows users to add tasks, mark them as completed, and delete them from the list. The app also saves the data locally, so the user's tasks are preserved even if they close the browser or shut down their device.

During the development of this app, I gained valuable experience in using HTML, CSS, and JavaScript to build a functional web application. I learned how to use the Document Object Model (DOM) to interact with HTML elements, how to use event listeners to trigger functionality, and how to use the local Storage API to save and retrieve data.

Throughout the development process, I also practiced important software development skills such as debugging, testing, and iterative development. I used the console.log () method to debug my code and ensure that it was functioning as intended. I tested the app thoroughly to ensure that it was user-friendly and free of errors. I also iterated on the design and functionality of the app based on user feedback and my own observations.

.

A screenshot of a computer screen

Description automatically generatedA screenshot of a to do list

Description automatically generated**OUTPUT: -**

A screenshot of a to do list

Description automatically generated