**TITLE:** CodTech IT Solutions Internship - Task Documentation: “To-DO LIST” Using CSS, HTML, JAVASCRIPT.

**INTERN INFORMATION:**

**Name:** THRISHA REDDY BUDDANNAGARI

**ID:** COD6016

**INTRODUCTION**:

In today’s fast-paced world, staying organized is crucial for productivity. One effective way to manage tasks is through a Todo List application. In this blog post, we’ll guide you through the process of creating a stylish and functional Todo List web app using HTML, CSS, and JavaScript.

**STEP 1 (HTML CODE):**

To get started, we will first need to create a basic HTML file. In this file, we will include the main structure for our to-do list.

After creating the files just paste the following codes into your file. Make sure to save your HTML document with a .html extension, so that it can be properly viewed in a web browser.

Let's break down the different sections and elements:

**1. <!DOCTYPE html>**: This declaration specifies that the document is an HTML5 document type

**2. <html lang="en">**: The opening tag of the HTML document, with the "lang" attribute set to "en" (English), indicating the language of the document's content.

**3. <head>**: The head section of the HTML document, which contains metadata and links to external resources.

**4. <title>**To-Do List**</title>**: Sets the title of the web page displayed in the browser's title bar or tab.

**5. <meta charset="UTF-8" />**: Specifies the character encoding for the document (UTF-8, which supports a wide range of characters and symbols).

**6. <meta name="viewport" content="width=device-width" />**: Configures the viewport width to match the device's width, providing better responsiveness for various screen sizes.

**7. <link rel="stylesheet" href="styles.css" />**: Links to an external CSS file named "styles.css" to apply styling to the HTML elements.

**8. <body>**: The body section of the HTML document, which contains the visible content of the web page.

**9. <section class="container">**: A section element with the class "container," used to group related content.

**10. <div class="heading">**: A division element with the class "heading" for the title and image at the top of the page.

**11. <img class="heading\_\_img" src="...">**: An image element with the class "heading\_\_img" that displays an image of a laptop.

**12. <h1 class="heading\_\_title">**To-Do List**</h1>**: A heading level 1 element with the class "heading\_\_title," displaying the text "To-Do List."

**13. <form class="form">**: A form element with the class "form" for user input.

**14. <label class="form\_\_label" for="todo">**...: A label for the input field, indicating what the user should input.

**15. <input class="form\_\_input" type="text" id="todo" name="to-do" size="30" required>**: An input field for the user to enter their to-do item. It has a class "form\_\_input," a unique ID "todo," a name "to-do," a size of 30 characters, and is required.

**16. <button class="button"><span>**Submit**</span></button>**: A button element with the class "button," containing the text "Submit."

**17. <div>**: A division element containing an unordered list (ul) for displaying the to-do list items.

**18. <ul class="toDoList">**: An unordered list with the class "toDoList," where to-do list items will be added dynamically.

**19. <script src="script.js"></script>**: Links to an external JavaScript file named "script.js" for adding interactivity and functionality to the web page.

This is the basic structure of our to-do list using HTML, and now we can move on to styling it using CSS.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>ToDo App</title>

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

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

</head>

<body>

<h1>todos</h1>

<form id="form">

<input

type="text"

id="input"

class="input"

placeholder="Enter your todo"

autocomplete="off"

/>

<ul class="todos" id="todos"></ul>

</form>

<small

>Left click to toggle complete. <br />Right click to delete the

todo.</small

>

</body>

</html>

**STEP 2 (CSS CODE):**

Once the basic HTML structure of the to-do list is in place, the next step is to add styling to the to-do list using CSS.

Next, we will create our CSS file. In this file, we will use some basic CSS rules to style our to-do list.

**1. @import**: This line imports a Google Fonts stylesheet, making the "Gochi Hand" font available for use on the page.

**2. body**: This block of code styles the entire body of the webpage. It sets the background color, minimum height, padding, box-sizing, and various other properties to center the content both horizontally and vertically. It also sets the text color and font family to "Gochi Hand" for a cursive font style.

**3. @media only screen and (min-width: 500px)**: Inside this media query, the minimum height of the body is adjusted to take up the full viewport height (100vh) when the screen width is at least 500px.

**4. .container**: This class styles a container element. It sets the width, minimum and maximum width, background color, border-radius, box shadow, and padding. It also adds a radial gradient background pattern.

**5. .heading**: This class styles a container for the heading. It aligns the content both horizontally and vertically and adds a margin at the bottom.

**6. .heading\_\_title**: This class styles the heading title, giving it a rotated appearance, rounded border, background color, and font size. Inside the media query, the font size is increased for larger screens.

**7. .form\_\_label**: This class styles labels for form inputs, adding a margin at the bottom.

**8. .form\_\_input**: This class styles form input fields. It sets the box-sizing, background color, padding, border radius, and border style. The focus state adjusts the border color.

**9. @media only screen and (min-width: 500px)**: Inside this media query, the width of the input fields are adjusted for larger screens.

**10. .button**: This class styles a button element. It sets various properties for the button appearance, including padding, rotation, font family, border radius, box shadow, and background image.

**11. .button span**: This styles the text inside the button, giving it a background color, padding, border, and border-radius.

**12. .button:active**,**.button:focus**: These styles define the appearance of the button when it is active (clicked) or focused (selected)

**13. .toDoList**: This class styles a to-do list on the webpage, aligning the text to the left.

**14. .toDoList li**: This class styles individual list items in the to-do list, adding padding.

**15. .toDoList li:hover**: This style is applied when hovering over a list item, giving it a line-through decoration with a wavy effect and changing the color.

This will give our to-do list an upgraded presentation. Create a CSS file with the name styles.css and paste the given codes into your CSS file. Remember that you must create a file with the .css extension.

@import url("https://fonts.googleapis.com/css2?family=Poppins:wght@200;400;600&display=swap");

\* {

box-sizing: border-box;

}

body {

background-color: #f5f5f5;

color: #444;

display: flex;

align-items: center;

flex-direction: column;

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

margin: 0;

min-height: 100vh;

}

h1 {

color: rgb(179, 131, 226);

font-size: 10rem;

text-align: center;

opacity: 0.4;

}

form {

box-shadow: 0 4px 10px rgba(0, 0, 0, 0.1);

max-width: 100%;

width: 400px;

}

.input {

border: none;

color: #444;

font-size: 2rem;

padding: 1rem 2rem;

display: block;

width: 100%;

}

.input::placeholder {

color: #d5d5d5;

}

.todos {

background-color: #fff;

padding: 0;

margin: 0;

list-style-type: none;

}

.todos li {

border-top: 1px solid #e5e5e5;

cursor: pointer;

font-size: 1.5rem;

padding: 1rem 2rem;

}

.todos li.completed {

color: #b6b6b6;

text-decoration: line-through;

}

small {

color: #b5b5b5;

margin-top: 3rem;

text-align: center;

}

**Step 3 (JavaScript Code):**

Finally, we need to create a function in JavaScript.

Let's break down the code step by step:

1. An **Immediately Invoked Function Expression (IIFE)** is used to encapsulate the entire code block. This pattern is often used to create a private scope for variables, preventing them from polluting the global scope.

2. Inside the IIFE, several state variables and UI variables are declared and initialized:

**toDoListArray**: An array that will store the To-Do list items.

**form**, **input**, and **ul**: These variables are used to reference specific elements in the HTML structure. form refers to the HTML form element, input refers to the input field within the form, and ul refers to an unordered list where the To-Do list items will be displayed.

3. Two event listeners are set up:

* The first listener is for the form's submit event. When the form is submitted (user presses Enter or clicks the "Submit" button), the function prevents the default form submission behavior (page reloads), generates a unique ID for the To-Do item, retrieves the input value, and then invokes two functions: addItemToDOM() and addItemToArray().
* The second listener is for the unordered list (ul). It listens for click events within the list. When an item within the list is clicked, it checks if the clicked element has a data-id attribute. If it does, it invokes two functions: removeItemFromDOM() and removeItemFromArray().

4. Four functions are defined within the IIFE:

* **addItemToDOM(itemId, toDoItem)**: This function creates a new list item (**<li>**) element, assigns it a data-id attribute with the provided itemId, sets the inner text of the list item to the provided toDoItem, and appends the list item to the unordered list (ul).
* **addItemToArray(itemId, toDoItem)**: This function adds a new object to the toDoListArray array, where each object represents a To-Do item with its associated itemId and toDoItem text.
* **removeItemFromDOM(id)**: This function finds the list item element with the specified data-id attribute (id) and removes it from the unordered list (ul).
* **removeItemFromArray(id)**: This function filters the toDoListArray to exclude the object with the matching itemId, effectively removing the corresponding To-Do item from the array.

5. The **IIFE** is immediately invoked by enclosing the entire code block in parentheses and adding () at the end. This ensures that the code within the IIFE is executed as soon as the JavaScript file is loaded.

Create a JavaScript file with the name of script.js and paste the given codes into your JavaScript file and make sure it's linked properly to your HTML document, so that the scripts are executed on the page. Remember, you’ve to create a file with .js extension.

const form = document.getElementById("form");

const input = document.getElementById("input");

const todosUL = document.getElementById("todos");

const todos = JSON.parse(localStorage.getItem("todos"));

if (todos) {

todos.forEach((todo) => {

addTodo(todo);

});

}

form.addEventListener("submit", (e) => {

e.preventDefault();

addTodo();

});

function addTodo(todo) {

let todoText = input.value;

if (todo) {

todoText = todo.text;

}

if (todoText) {

const todoEl = document.createElement("li");

if (todo && todo.completed) {

todoEl.classList.add("completed");

}

todoEl.innerText = todoText;

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

todoEl.classList.toggle("completed");

updateLS();

});

todoEl.addEventListener("contextmenu", (e) => {

e.preventDefault();

todoEl.remove();

updateLS();

});

todosUL.appendChild(todoEl);

input.value = "";

updateLS();

}

}

function updateLS() {

const todosEl = document.querySelectorAll("li");

const todos = [];

todosEl.forEach((todoEl) => {

todos.push({

text: todoEl.innerText,

completed: todoEl.classList.contains("completed"),

});

});

localStorage.setItem("todos", JSON.stringify(todos));

}

**CONCLUSION:**

Congratulations! You've successfully created a responsive to-do list app using HTML, CSS, and JavaScript. You've mastered the art of structuring web content, styling interfaces, and adding functionality. Now, you can efficiently manage your tasks whether you're at your desk or on the go.

