**HTML:**

# Simple Calculator using HTML,CSS and JAVASCRIPT

<!DOCTYPE html>

<html>

<head>

<title>Simple Calculator</title>

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

</head>

<body>

<h1>Simple Calculator using html, css & js </h1>

<div class="calculator">

<input type="text" id="display" disabled>

<button onclick="clearDisplay()">C</button>

<button onclick="appendToDisplay('7')">7</button>

<button onclick="appendToDisplay('8')">8</button>

<button onclick="appendToDisplay('9')">9</button>

<button onclick="appendToDisplay('/')">/</button>

<button onclick="appendToDisplay('4')">4</button>

<button onclick="appendToDisplay('5')">5</button>

<button onclick="appendToDisplay('6')">6</button>

<button onclick="appendToDisplay('\*')">\*</button>

<button onclick="appendToDisplay('1')">1</button>

<button onclick="appendToDisplay('2')">2</button>

<button onclick="appendToDisplay('3')">3</button>

<button onclick="appendToDisplay('-')">-</button>

<button onclick="appendToDisplay('0')">0</button>

<button onclick="appendToDisplay('.')">.</button>

<button onclick="calculate()">=</button>

<button onclick="appendToDisplay('+')">+</button>

</div>

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

</body>

</html>

# Css:

body {

font-family: Arial, sans-serif; display: flex;

flex-direction: column; justify-content: center; align-items: center; height: 100vh;

margin: 0;

}

.calculator { display: grid;

grid-template-columns: repeat(4, 1fr); grid-gap: 10px;

max-width: 300px;

}

input {

grid-column: span 4; padding: 10px;

font-size: 20px;

}

button { padding: 10px; font-size: 20px; cursor: pointer;

background-color: #f1f1f1; border: 1px solid #ccc;

}

button:hover { background-color: #ddd;

}

**JavaScript:**

let display = document.getElementById('display');

function appendToDisplay(value) { display.value += value;

}

function calculate() { try {

display.value = eval(display.value);

} catch (error) { display.value = 'Error';

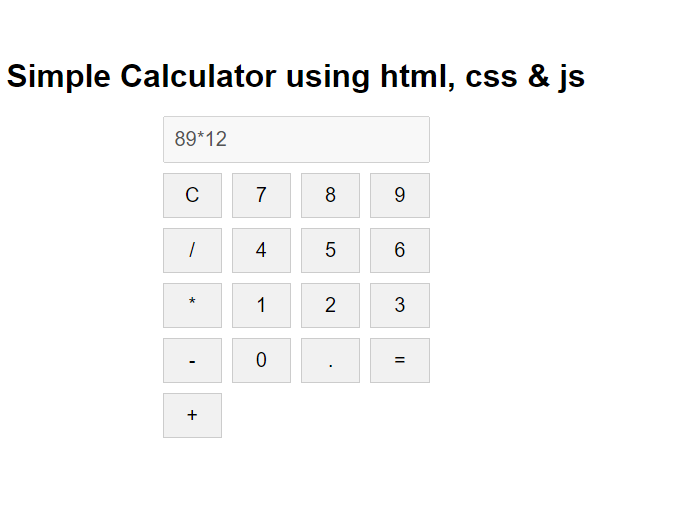
}

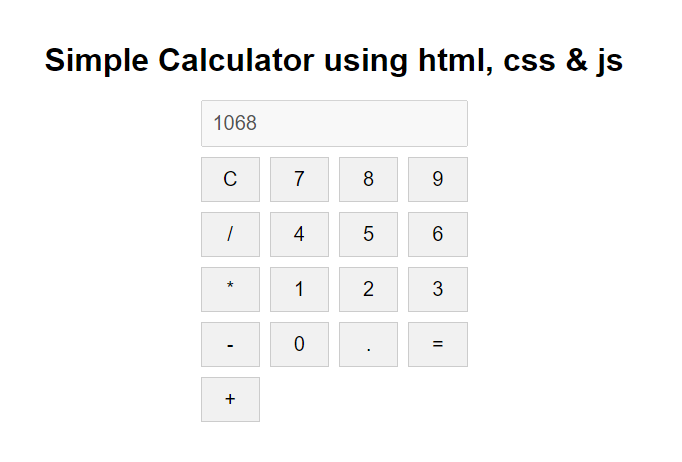
}

function clearDisplay() { display.value = '';

}

# OUTPUT:





**HTML:**

# E-mail validation using HTML, CSS and JAVA SCRIPT

<!DOCTYPE html>

<html>

<head>

<title>Email Validation</title>

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

</head>

<body>

<div class="container">

<h1>Email Validation using Html,Css & Js</h1>

<form id="emailForm" onsubmit="return validateEmail()">

<input type="email" id="emailInput" placeholder="Enter your email address" required>

<button type="submit">Submit</button>

<p id="errorText"></p>

</form>

</div>

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

</body>

</html>

**Css:**

body {

font-family: Arial, sans-serif; background-color: #f7f7f7; margin: 0;

padding: 0;

}

.container {

max-width: 400px; margin: 0 auto;

padding: 20px;

border: 1px solid #ddd; background-color: #fff;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

h1 {

text-align: center;

}

input[type="email"] { width: 100%; padding: 10px; margin-bottom: 10px;

border: 1px solid #ddd;

}

button {

background-color: #4CAF50; color: #fff;

padding: 10px 20px; border: none; cursor: pointer;

}

p#errorText { color: red;

text-align: center; margin-top: 10px;

}

**JavaScript:**

function validateEmail() {

const emailInput = document.getElementById('emailInput'); const errorText = document.getElementById('errorText'); const email = emailInput.value.trim();

const emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/; if (!emailRegex.test(email)) {

errorText.textContent = 'Please enter a valid email address.'; return false;

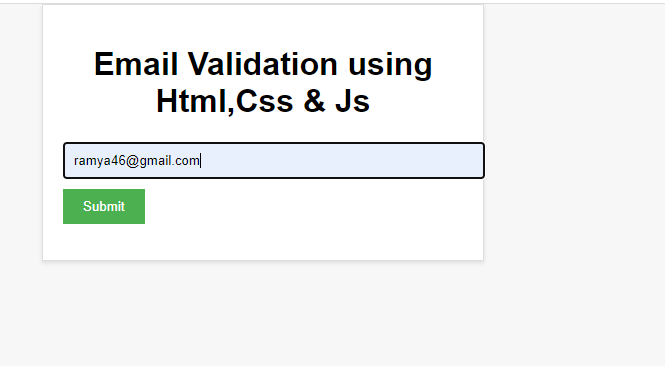
}

// If email is valid, clear the error message and submit the form

errorText.textContent = ''; return true;

}

# OUTPUT:



**HTML:**

# Temperature Conversion using HTML, CSS and JAVA SCRIPT

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Temperature Converter</title>

<link rel="stylesheet" href="temp2.css">

</head>

<body>

<div class="container">

<h1>Temperature Converter</h1>

<div class="input-container">

<label for="celsius">Celsius:</label>

<input type="number" id="celsius" placeholder="Enter temperature in Celsius">

</div>

<div class="input-container">

<label for="fahrenheit">Fahrenheit:</label>

<input type="number" id="fahrenheit" placeholder="Enter temperature in Fahrenheit">

</div>

<button id="convertBtn">Convert</button>

<div class="result" id="result"></div>

</div>

<script src="temp3.js"></script>

</body>

</html>

**CSS:**

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

}

.container {

max-width: 500px; margin: 50px auto; padding: 20px; background-color: #fff;

box-shadow: 0 0 5px rgba(0, 0, 0, 0.2); text-align: center;

}

.input-container { margin-bottom: 10px;

}

label {

display: inline-block; width: 120px;

text-align: left;

}

input { width: 80%;

padding: 8px;

border: 1px solid #ccc; border-radius: 4px;

}

button {

padding: 10px 20px; background-color: #007bff; color: #fff;

border: none; border-radius: 4px; cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

.result {

margin-top: 20px; font-weight: bold;

}

**JavaScript:**

const celsiusInput = document.getElementById('celsius');

const fahrenheitInput = document.getElementById('fahrenheit'); const convertBtn = document.getElementById('convertBtn'); const resultDiv = document.getElementById('result');

convertBtn.addEventListener('click', () => { if (celsiusInput.value !== '') {

const celsius = parseFloat(celsiusInput.value); const fahrenheit = (celsius \* 9/5) + 32; fahrenheitInput.value = fahrenheit.toFixed(2);

resultDiv.textContent = `${celsius}°C is ${fahrenheit.toFixed(2)}°F`;

} else if (fahrenheitInput.value !== '') {

const fahrenheit = parseFloat(fahrenheitInput.value); const celsius = (fahrenheit - 32) \* 5/9; celsiusInput.value = celsius.toFixed(2);

resultDiv.textContent = `${fahrenheit}°F is ${celsius.toFixed(2)}°C`;

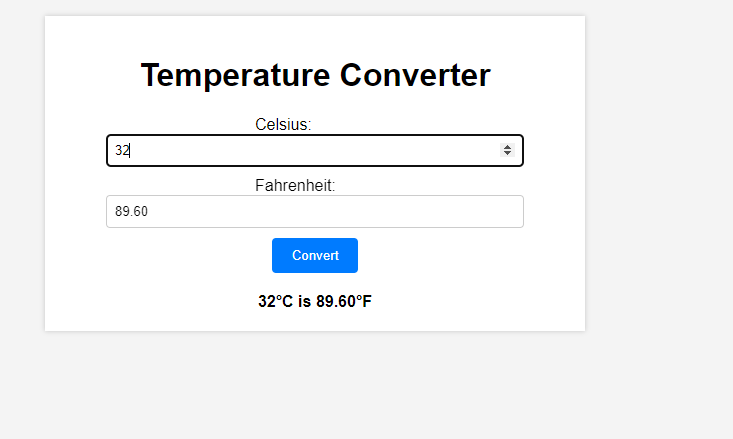
} else {

resultDiv.textContent = "Please enter a temperature.";

}

});

# OUTPUT:



**Button Loading Animation using HTML, CSS and JAVASCRIPT HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>Button Loading Animation</title>

</head>

<body>

<button id="loading-btn" class="btn">Click me</button>

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

</body>

</html>

# CSS:

**.**btn {

padding: 10px 20px; font-size: 16px;

background-color: #007bff; color: white;

border: none; cursor: pointer;

transition: background-color 0.3s;

}

.btn.loading { background-color: #ccc; cursor: not-allowed;

}

# JavaScript:

const button = document.getElementById('loading-btn'); button.addEventListener('click', () => { button.classList.add('loading');

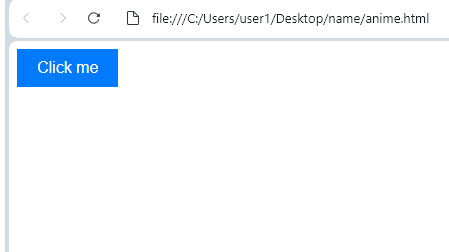
button.innerHTML = 'Loading...';

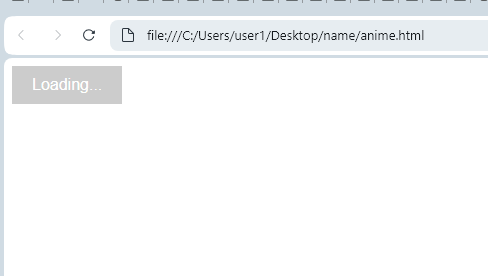
// Simulate a task delay (e.g., fetching data) setTimeout(() => { button.classList.remove('loading'); button.innerHTML = 'Click me';

}, 2000); // Change this delay as needed

});

# OUTPUT:





**Stop Watch Monitoring using HTML, CSS and JAVASCRIPT PROGRAM:**

<!DOCTYPE html>

<html>

<head>

<style>

.stopwatch { font-size: 2em; margin: 20px;

text-align: center;

}

</style>

</head>

<body>

<div class="stopwatch" id="stopwatch">00:00:00</div>

<button onclick="startStopwatch()">Start</button>

<button onclick="stopStopwatch()">Stop</button>

<button onclick="resetStopwatch()">Reset</button>

<script>

let startTime; let intervalId;

function startStopwatch() {

startTime = Date.now() - (startTime ? startTime : 0); intervalId = setInterval(updateStopwatch, 1000);

}

function stopStopwatch() { clearInterval(intervalId);

}

function resetStopwatch() {

clearInterval(intervalId); startTime = null;

document.getElementById('stopwatch').textContent = '00:00:00';

}

function updateStopwatch() {

const elapsedTime = Date.now() - startTime;

const hours = Math.floor(elapsedTime / 3600000);

const minutes = Math.floor((elapsedTime % 3600000) / 60000); const seconds = Math.floor((elapsedTime % 60000) / 1000); const hoursStr = hours.toString().padStart(2, '0');

const minutesStr = minutes.toString().padStart(2, '0'); const secondsStr = seconds.toString().padStart(2, '0');

document.getElementById('stopwatch').textContent =

`${hoursStr}:${minutesStr}:${secondsStr}`;

}

</script>

</body>

</html>

# OUTPUT:

START





STOP



**HTML:**

# Password strength Checker using HTML, CSS and JAVASCRIPT

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>Password Strength Checker</title>

</head>

<body>

<div class="container">

<h1>Password Strength Checker</h1>

<input type="password" id="password" placeholder="Enter your password">

<div id="strength-indicator"></div>

</div>

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

</body>

</html>

**CSS:**

body {

font-family: Arial, sans-serif; background-color: #f4f4f4; margin: 0;

display: flex;

justify-content: center; align-items: center; height: 100vh;

}

.container {

text-align: center; background-color: #fff; border-radius: 10px; padding: 20px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

input[type="password"] { width: 100%;

padding: 10px; margin: 10px 0; border: 1px solid #ccc; border-radius: 5px; font-size: 16px;

}

#strength-indicator { margin-top: 10px;

}

**JavaScript:**

const passwordInput = document.getElementById('password');

const strengthIndicator = document.getElementById('strength-indicator');

passwordInput.addEventListener('input', updatePasswordStrength);

function updatePasswordStrength() { const password = passwordInput.value;

const strength = calculatePasswordStrength(password); const strengthText = getStrengthText(strength);

strengthIndicator.textContent = strengthText; strengthIndicator.style.color = getColorForStrength(strength);

}

function calculatePasswordStrength(password) {

// You can implement your own password strength calculation logic here

// For simplicity, let's just count the characters for this example const minLength = 6;

const maxLength = 10;

const length = password.length;

if (length < minLength) return 0;

if (length >= minLength && length <= maxLength) return 1; return 2;

}

function getStrengthText(strength) {

const strengthTexts = ['Weak', 'Medium', 'Strong']; return strengthTexts[strength];

}

function getColorForStrength(strength) {

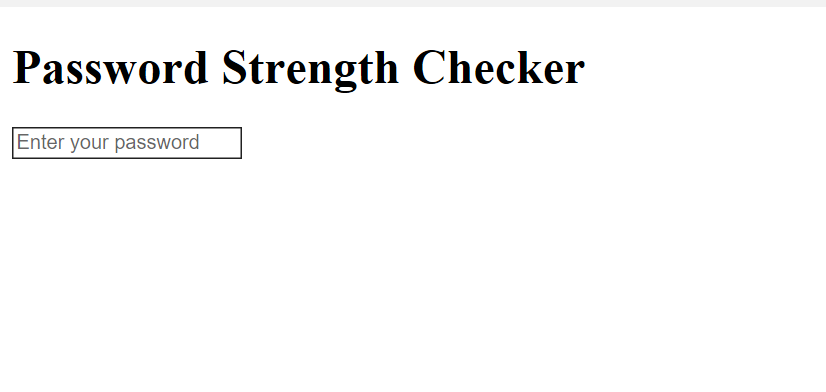
const colors = ['#FF5733', '#FFC300', '#4CAF50'];

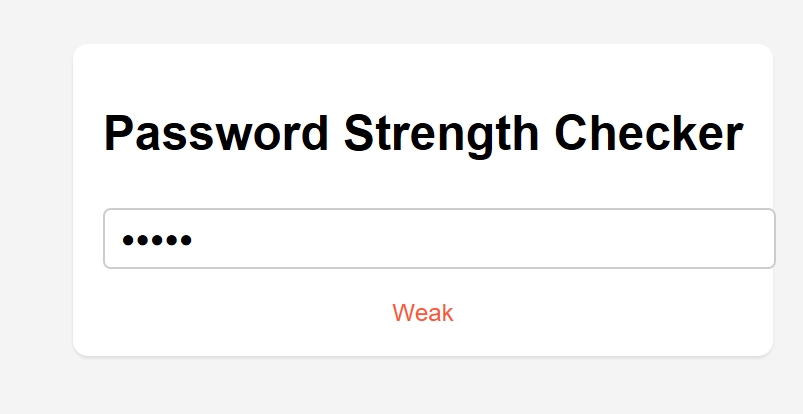
return colors[strength];

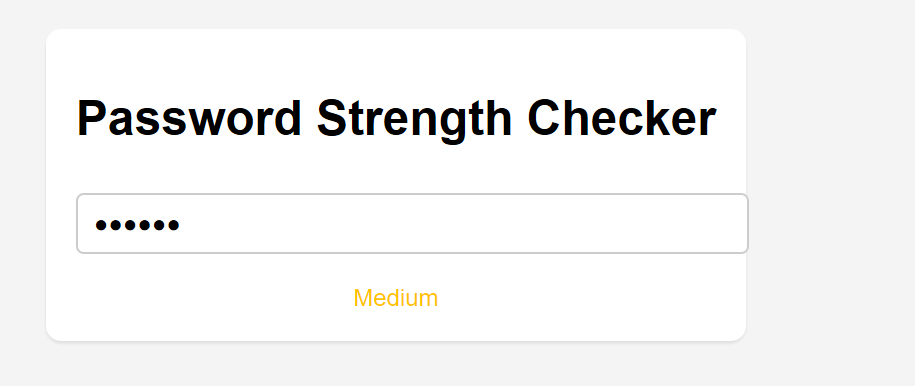
}

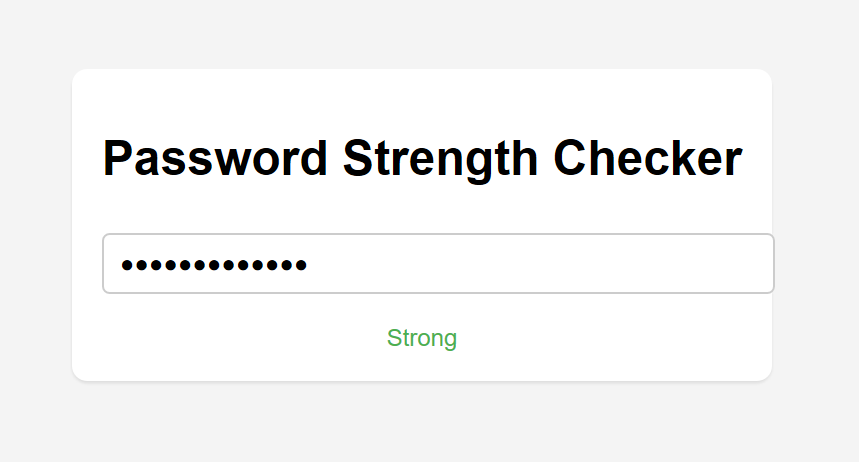


**OUTPUT:**









# To-do list using HTML, CSS and JAVASCRIPT

**HTML:**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>To-Do List</title>

</head>

<body>

<div class="todo-container">

<h1>To-Do List</h1>

<input type="text" id="task" placeholder="Enter a task...">

<button id="addTask">Add Task</button>

<ul id="taskList"></ul>

</div>

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

</body>

</html>

# CSS:

body {

font-family: Arial, sans-serif; margin: 0;

padding: 0;

background-color: #f4f4f4;

}

.todo-container { max-width: 400px; margin: 50px auto;

background-color: #fff; padding: 20px;

border-radius: 5px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

h1 {

text-align: center;

}

input[type="text"] { width: 100%; padding: 10px; margin-bottom: 10px;

border: 1px solid #ccc; border-radius: 3px;

}

button {

display: block; width: 100%; padding: 10px;

background-color: #007bff; color: #fff;

border: none; border-radius: 3px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

ul {

list-style-type: none; padding: 0;

}

li {

display: flex;

justify-content: space-between; align-items: center; background-color: #f9f9f9; padding: 10px;

margin-bottom: 5px; border-radius: 3px;

}

.delete-button {

background-color: #ff4136; color: #fff;

border: none; border-radius: 3px; padding: 5px 10px; cursor: pointer;

}

# Java Script:

const addTaskButton = document.getElementById('addTask'); const taskInput = document.getElementById('task');

const taskList = document.getElementById('taskList');

addTaskButton.addEventListener('click', addTask);

function addTask() {

const taskText = taskInput.value; if (taskText.trim() !== '') {

const listItem = document.createElement('li'); listItem.innerHTML = `

<span>${taskText}</span>

<button class="delete-button">Delete</button>

`; taskList.appendChild(listItem); taskInput.value = '';

const deleteButton = listItem.querySelector('.delete-button'); deleteButton.addEventListener('click', () => {

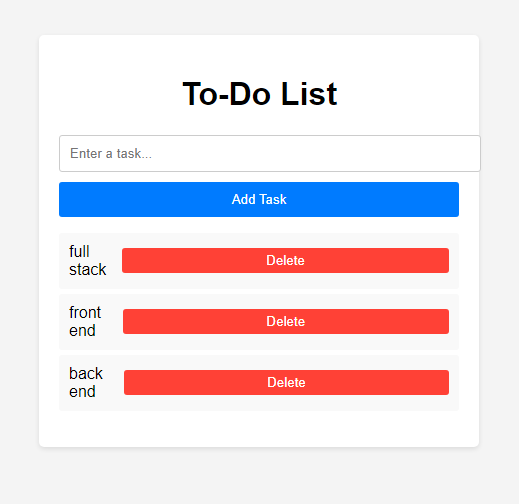
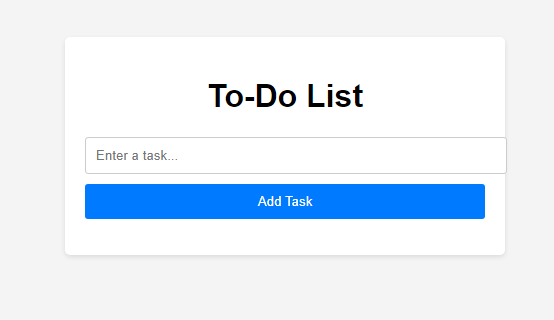
taskList.removeChild(listItem);

});

}

}

# OUTPUT:



**Palindrome Checker using HTML,CSS and JS**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>Palindrome Checker</title>

</head>

<body>

<div class="container">

<h1>Palindrome Checker</h1>

<input type="text" id="inputText" placeholder="Enter a word or phrase">

<button onclick="checkPalindrome()">Check</button>

<p id="result"></p>

</div>

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

</body>

</html>

CSS

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

padding: 0;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

}

.container {

background-color: #ffffff;

padding: 20px;

border-radius: 5px;

box-shadow: 0 2px 5px rgba(0, 0, 0, 0.1);

}

h1 {

font-size: 24px;

margin-bottom: 10px;

}

input {

padding: 10px;

width: 100%;

border: 1px solid #ccc;

border-radius: 5px;

margin-bottom: 10px;

}

button {

padding: 10px 20px;

background-color: #007bff;

color: #ffffff;

border: none;

border-radius: 5px;

cursor: pointer;

}

button:hover {

background-color: #0056b3;

}

JS

function checkPalindrome() {

const inputText = document.getElementById("inputText").value.toLowerCase().replace(/\s/g, "");

const reversedText = inputText.split("").reverse().join("");

if (inputText === reversedText) {

document.getElementById("result").textContent = "It's a palindrome!";

} else {

document.getElementById("result").textContent = "It's not a palindrome.";

}

}

**item selector in dropdown using HTML,CSS and JS**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<link rel="stylesheet" href="item2.css">

<title>Item Selector</title>

</head>

<body>

<div class="dropdown">

<button class="dropdown-btn">Select an Item</button>

<ul class="dropdown-list">

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

<li>Item 4</li>

</ul>

</div>

<script src="item3.js"></script>

</body>

</html>

CSS

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.dropdown {

position: relative;

display: inline-block;

}

.dropdown-btn {

padding: 10px 20px;

background-color: #3498db;

color: #fff;

border: none;

cursor: pointer;

}

.dropdown-list {

display: none;

position: absolute;

list-style: none;

margin: 0;

padding: 0;

background-color: #f1f1f1;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.dropdown-list li {

padding: 10px;

border-bottom: 1px solid #ccc;

cursor: pointer;

}

.dropdown-list li:last-child {

border-bottom: none;

}

JS

document.addEventListener('DOMContentLoaded', function() {

const dropdownBtn = document.querySelector('.dropdown-btn');

const dropdownList = document.querySelector('.dropdown-list');

dropdownBtn.addEventListener('click', function() {

dropdownList.style.display = dropdownList.style.display === 'block' ? 'none' : 'block';

});

dropdownList.addEventListener('click', function(event) {

if (event.target.tagName === 'LI') {

dropdownBtn.textContent = event.target.textContent;

dropdownList.style.display = 'none';

}

});

document.addEventListener('click', function(event) {

if (!dropdownBtn.contains(event.target)) {

dropdownList.style.display = 'none';

}

});

});

**Captcha generation and Validation using HTML,CSS and JS**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>CAPTCHA Example</title>

</head>

<body>

<div class="captcha-container">

<p>Prove you're not a robot:</p>

<div class="captcha-box">

<span id="captcha-operation"></span>

<input type="text" id="captcha-input" placeholder="Enter result">

<button id="captcha-submit">Submit</button>

</div>

<p id="captcha-feedback"></p>

</div>

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

</body>

</html>

CSS

body {

font-family: Arial, sans-serif;

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

}

.captcha-container {

text-align: center;

}

.captcha-box {

margin-top: 10px;

}

#captcha-feedback {

margin-top: 10px;

}

JS

document.addEventListener("DOMContentLoaded", function () {

generateCaptcha();

const captchaInput = document.getElementById("captcha-input");

const captchaSubmit = document.getElementById("captcha-submit");

const captchaFeedback = document.getElementById("captcha-feedback");

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

const userInput = captchaInput.value.trim();

const operationResult = eval(document.getElementById("captcha-operation").textContent);

if (userInput === operationResult.toString()) {

captchaFeedback.textContent = "CAPTCHA passed!";

captchaFeedback.style.color = "green";

} else {

captchaFeedback.textContent = "CAPTCHA failed. Please try again.";

captchaFeedback.style.color = "red";

generateCaptcha();

}

});

});

function generateCaptcha() {

const operators = ["+", "-", "\*"];

const num1 = Math.floor(Math.random() \* 10);

const num2 = Math.floor(Math.random() \* 10);

const operator = operators[Math.floor(Math.random() \* operators.length)];

const captchaOperation = `${num1} ${operator} ${num2}`;

document.getElementById("captcha-operation").textContent = captchaOperation;

document.getElementById("captcha-input").value = "";

document.getElementById("captcha-feedback").textContent = "";

}

**traffic signal creation using HTML,CSS and JS**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Traffic Signal Simulation</title>

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

</head>

<body>

<div class="traffic-signal">

<div class="light red"></div>

<div class="light yellow"></div>

<div class="light green"></div>

</div>

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

</body>

</html>

CSS

body {

display: flex;

justify-content: center;

align-items: center;

height: 100vh;

margin: 0;

background-color: #f0f0f0;

}

.traffic-signal {

display: flex;

flex-direction: column;

align-items: center;

background-color: #333;

padding: 10px;

border-radius: 8px;

}

.light {

width: 80px;

height: 80px;

border-radius: 50%;

margin: 5px;

transition: background-color 0.3s;

}

.red {

background-color: red;

}

.yellow {

background-color: yellow;

}

.green {

background-color: green;

}

JS

const redLight = document.querySelector('.red');

const yellowLight = document.querySelector('.yellow');

const greenLight = document.querySelector('.green');

function switchLights() {

setTimeout(() => {

redLight.style.backgroundColor = 'red';

yellowLight.style.backgroundColor = 'gray';

greenLight.style.backgroundColor = 'gray';

setTimeout(() => {

redLight.style.backgroundColor = 'gray';

yellowLight.style.backgroundColor = 'yellow';

greenLight.style.backgroundColor = 'gray';

setTimeout(() => {

redLight.style.backgroundColor = 'gray';

yellowLight.style.backgroundColor = 'gray';

greenLight.style.backgroundColor = 'green';

switchLights(); // Repeat the cycle

}, 2000);

}, 1000);

}, 2000);

}

switchLights();

**Random Quotes Generator using HTML,CSS and JS**

HTML

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

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

<title>Random Quotes Generator</title>

</head>

<body>

<div class="container">

<h1>Random Quotes Generator</h1>

<div class="quote">

<p id="quoteText">Click below to generate a random quote</p>

</div>

<button id="generateBtn">Generate Quote</button>

</div>

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

</body>

</html>

CSS

body {

font-family: Arial, sans-serif;

background-color: #f4f4f4;

margin: 0;

display: flex;

justify-content: center;

align-items: center;

min-height: 100vh;

}

.container {

text-align: center;

background-color: #fff;

padding: 20px;

border-radius: 5px;

box-shadow: 0px 0px 10px rgba(0, 0, 0, 0.2);

}

h1 {

color: #333;

}

.quote {

margin: 20px 0;

}

#generateBtn {

padding: 10px 20px;

font-size: 16px;

background-color: #007bff;

border: none;

color: #fff;

border-radius: 5px;

cursor: pointer;

}

JS

const quotes = [

"The only way to do great work is to love what you do. - Steve Jobs",

"Success is not final, failure is not fatal: It is the courage to continue that counts. - Winston Churchill",

"Believe you can and you're halfway there. - Theodore Roosevelt",

"The future belongs to those who believe in the beauty of their dreams. - Eleanor Roosevelt",

"In the middle of every difficulty lies opportunity. - Albert Einstein",

"The only limit to our realization of tomorrow will be our doubts of today. - Franklin D. Roosevelt",

"Life is what happens when you're busy making other plans. - John Lennon"

];

function generateRandomQuote() {

const randomIndex = Math.floor(Math.random() \* quotes.length);

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

quoteText.textContent = quotes[randomIndex];

}

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

generateBtn.addEventListener("click", generateRandomQuote);

**13 . Rest api**

const express = require("express") const app = express()

const port = 5000

app.use(express.json()) app.get("/data", (req, res)=>{

res.json({message : "Get Method"})

})

app.post("/data", (req, res)=>{ res.json({message : "Post Method"})

})

app.put("/data", (req, res)=>{ res.json({message : "put Method"})

})

app.delete("/data", (req, res)=>{ res.json({message : "delete Method"})

})

app.listen( port, (req, res)=>{ console.log("server started !...");

})

**14 . Login using Angular js**

<style>

.main{

width: 100vw; height: 100vh; display: flex;

flex-direction: column; align-items: center; justify-content: center;

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

}

.logcomp{ padding: 30px;

border-radius: 6px; width: 250px; height: 300px; gap: 20px;

background: linear-gradient(to bottom right, #ff2ab7, #7522a1);

/\* background-color: grey; \*/ color: white;

}

.form{ display: flex;

flex-direction: column; gap: 10px;

}

input{

padding: 10px; border-radius: 2px; border: none;

}

.btn{

width: 70px; padding: 10px; border-radius: 6px; border: none; margin-top: 25px; margin-left: 80px;

}

.login{

color: orange;

}

.in{

color: yellow;

}

.Angular{ color: red;

}

</style>

<div class="main" >

<h3><span class="login" >Log<span class="in">in</span></span> Form Using <span class="Angular" >Angular.Js</span> </h3>

<div class="logcomp" >

<h4>Login</h4>

<form class="form" >

<label>Email</label>

<input type="text" />

<label>Password</label>

<input type="text" />

<button class="btn" >Submit</button>

</form>

</div>

</div>

<style>

  .main{

  width: 100vw; height: 100vh; display: flex;

  flex-direction: column; align-items: center; justify-content: center;

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

  }

  .logcomp{ padding: 30px;

  border-radius: 6px; width: 250px; height: 300px; gap: 20px;

  background: linear-gradient(to bottom right, #ff2ab7, #7522a1);

*/\* background-color: grey; \*/* color: white;

  }

  .form{ display: flex;

  flex-direction: column; gap: 10px;

  }

  input{

  padding: 10px; border-radius: 2px; border: none;

  }

  .btn{

  width: 70px; padding: 10px; border-radius: 6px; border: none; margin-top: 25px; margin-left: 80px;

  }

  .login{

  color: orange;

  }

  .in{

  color: yellow;

  }

  .Angular{ color: red;

  }

  </style>

  <div class="main" >

  <h3><span class="login" >Log<span class="in">in</span></span> Form Using <span class="Angular" >Angular.Js</span> </h3>

  <div class="logcomp" >

  <h4>Login</h4>

  <form class="form" >

  <label>Email</label>

  <input type="text" />

  <label>Password</label>

  <input type="text" />

  <button class="btn" >Submit</button>

  </form>

  </div>

  </div>

**15. CRUD using Mongodb, node & express**

const express = require("express") const mongoose = require("mongoose")

const use = require("./model/usermodule")

const app = express() const port = 5000

const con = async() =>{

const connect = await mongoose.connect(`mongodb://127.0.0.1:27017`) console.log({name : connect.connection.name, host : connect.connection.host});

}

app.use(express.json()) app.get("/data", async(req, res)=>{

const data = await use.find() res.status(200).json(data)

})

app.post("/data", async(req ,res)=>{ console.log(req.body);

const data = await use.create({ username : req.body.username, email :req.body.email, password : req.body.password

})

res.json(data)

})

app.put("/data/:id", async(req, res)=>{

const insert = await use.findByIdAndUpdate(id,{ name,

email, password

},{new : true}) res.status(200).json(insert)

})

app.delete("/data/:id", async(req, res)=>{ const delet = await use.deleteOne({\_id : id}) if(delet){

res.status(200).json({message:`The Deleted contact is ${req.params.id}`})

}

})

con() if(con){

app.listen( port, (req, res)=>{ console.log("server started !...");

})

}

**Userschema :**

const mongoose = require("mongoose")

const userschema = mongoose.Schema({

username : {

type : String,

require :[true, "Name is mandatory"]

},

email : {

type :String,

require : [true, "email is mandatory"]

},

password : {

type : String,

require :[true, "Password id Mandatory"]

}

})

module.exports = mongoose.model("use", userschema)

**const** express = require("express")

**const** mongoose = require("mongoose")

**const** use = require("./module/userSchema")

**const** app = express()

**const** port = 5000

**const** con = **async**() **=>**{

**const** connect = await mongoose.connect(`mongodb+srv://ranji:ranji@ranji.e148dlq.mongodb.net/ranji?retryWrites=true&w=majority`)

console.log({name : connect.connection.name, host : connect.connection.host});

}

app.use(express.json())

app.get("/data", **async**(req, res)**=>**{

**const** data = await use.find()

    res.status(200).json(data)

})

app.post("/data", **async**(req ,res)**=>**{ console.log(req.body);

**const** data = await use.create({ username : req.body.username, email :req.body.email, password : req.body.password})

res.json(data)

})

app.put("/data/:id", **async**(req, res)**=>**{

**const** insert = await use.findByIdAndUpdate(id,{ name,

email, password

},{new : true})

res.status(200).json(insert)

})

app.delete("/data/:id", **async**(req, res)**=>**{

**const** delet = await use.deleteOne({\_id : id})

    if(delet){

    res.status(200).json({message:`The Deleted contact is ${req.params.id}`})

    }

})

con()

if(con){

app.listen( port, (req, res)**=>**{ console.log("server started !...");

})

}