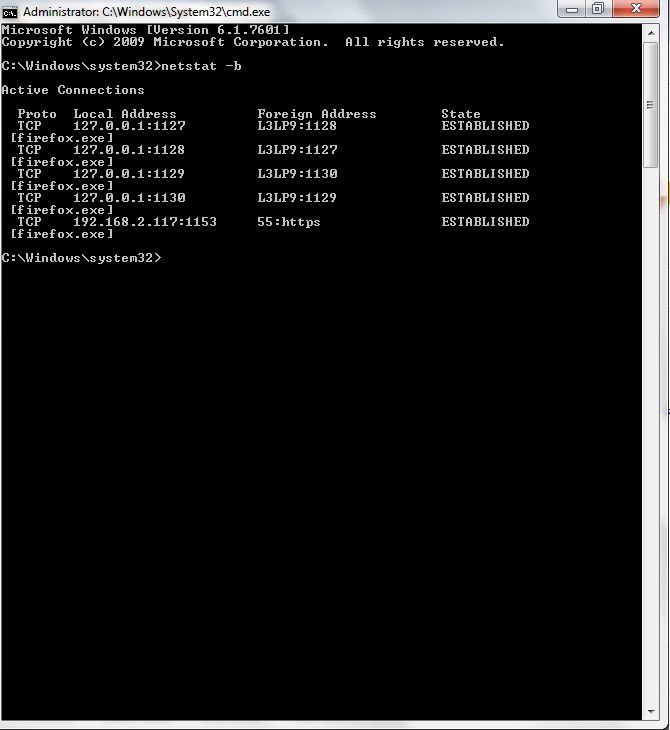
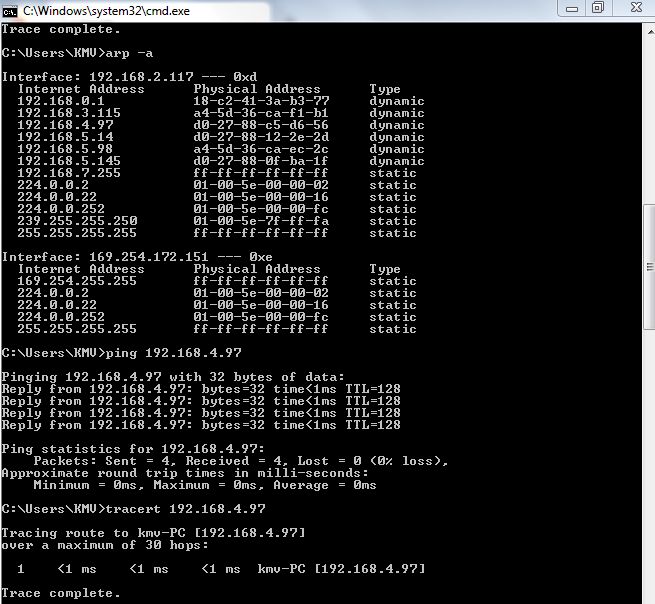
Practicals:

1. Display your systems IP Address, Subnet mask using ipconfig, and find out the network address and the maximum number of systems possible on your network and range of IP addresses available to these systems.
2. With help of ping, check if you are connected to other systems of your network and find the route to connect to that system using tracert. List all the processes which are using ports for TCP protocol.
3. Create an HTML page that shows information about you, your course, hobbies, address, and your plans. Use CSS for styling of HTML page so that looks nice.

**Code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Document</title>

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

</head>

<body>

<div class="container">

<article class="card">

<div class="thumb1"></div>

<div class="infos">

<h2 class="title">V <span class="flag"></span></h2>

<h3 class="date">Bsc hons computer science</h3>

<h3 class="seats">footbaal badminton coding</h3>

<p class="txt">

a/56 Night Cityy<br>

Phantom Liberty <br>

Cyberpunk 2077

</p>

<h3 class="details">paeceful life</h3>

</div>

</article>

<article class="card">

<div class="thumb2"></div>

<div class="infos">

<h2 class="title">Aman Raj<span class="flag"></span></h2>

<h3 class="date">Bsc hons computer science</h3>

<h3 class="seats">Founder Game Tests.co </h3>

<p class="txt">

Gulag,<br>

BRZ,<br>

Arkham City

</p>

<h3 class="details">ruthless vigilante</h3>

</div>

</article>

</div>

</body>

</html>



1. Create an HTML page with the sole purpose to show multiplication tables of 2 to 10 (row-wise) created by JavaScript. Initially, the page is blank. With help of setInterval function print a row every 5 seconds in different colors and increasing font size.

**Code**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Multiplication Tables</title>

<script src="https://code.jquery.com/jquery-3.6.4.min.js"></script>

<style>

.multiplication-table {

margin: 10px;

transition: font-size 0.5s, background-color 0.5s;

}

.multiplication-table th, .multiplication-table td {

border: 1px solid #dddddd;

text-align: center;

padding: 10px;

width: 40px;

}

</style>

</head>

<body>

<div id="multiplicationTablesContainer"></div>

<script>

function generateRandomColor() {

const letters = '0123456789ABCDEF';

let color = '#';

for (let i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

return color;

}

function generateMultiplicationTable(factor, columns, fontSize, bgColor) {

let tableHtml = `<table class="multiplication-table" style="font-size: ${fontSize}px; background-color: ${bgColor};"><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr><tr><th></th></tr>`;

for (let i = 1; i <= 10; i++) {

tableHtml += `<tr><td>${factor} x ${i} = ${factor \* i}</td></tr>`;

}

tableHtml += `</table>`;

return tableHtml;

}

const totalTables = 10;

let currentTable = 2;

let fontSize = 16; // Initial font size

function displayNextTable() {

if (currentTable > totalTables) {

clearInterval(intervalId); // Stop the interval when all tables are displayed

} else {

const bgColor = generateRandomColor();

const tableHtml = generateMultiplicationTable(currentTable, 10, fontSize, bgColor);

$('#multiplicationTablesContainer').append(tableHtml);

currentTable++;

fontSize += 2; // Increase font size for the next table

}

}

// Set interval to append tables every 5000 milliseconds (5 seconds)

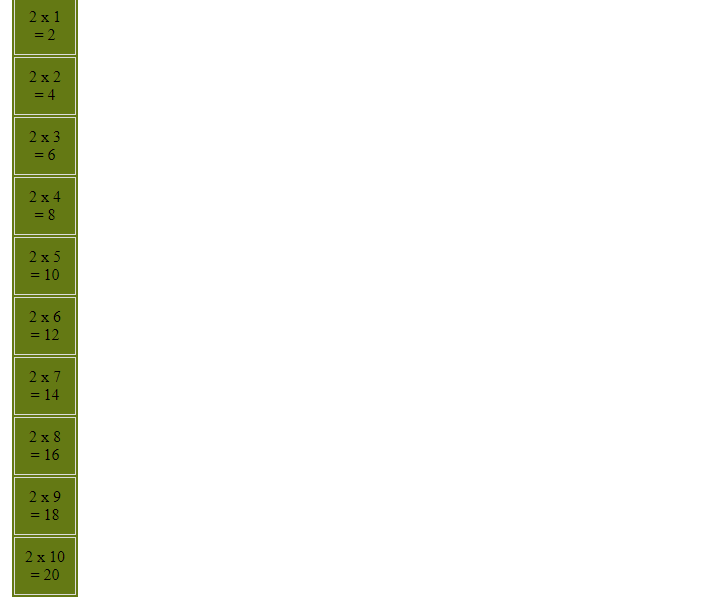
const intervalId = setInterval(displayNextTable, 5000);

</script>

</body>

</html>

**Output**

****

1. Create an HTML page with a paragraph written on it and under which 9 buttons are placed in a 3X3 grid. The first row is for buttons labeled with colors names Red, Green, and Blue, the second row with numbers 10, 20, 30, and the third row with different font names. Click event of each of the buttons should make the appropriate change in the style of paragraph.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Button Grid</title>

<style>

body {

font-family: Arial, sans-serif;

text-align: center;

}

#button-grid {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 10px;

margin-top: 20px;

}

.button {

padding: 10px 20px;

font-size: 16px;

background-color: #3498db;

color: #fff;

border: none;

cursor: pointer;

}

/\* Add initial styles for the paragraph \*/

#paragraph {

color: black; /\* Default color \*/

font-size: 16px; /\* Default font size \*/

font-family: Arial, sans-serif; /\* Default font family \*/

}

</style>

</head>

<body>

<h1>Button Grid Example</h1>

<!-- Add the paragraph with the id "paragraph" -->

<p id="paragraph">Lorem ipsum dolor sit amet consectetur adipisicing elit...</p>

<div id="button-grid">

<button class="button" onclick="changeColor('red')">Red</button>

<button class="button" onclick="changeColor('green')">Green</button>

<button class="button" onclick="changeColor('blue')">Blue</button>

<button class="button" onclick="changeFontSize(10)">10</button>

<button class="button" onclick="changeFontSize(20)">20</button>

<button class="button" onclick="changeFontSize(30)">30</button>

<button class="button" onclick="changeFont('Arial')">Arial</button>

<button class="button" onclick="changeFont('Times New Roman')">Times New Roman</button>

<button class="button" onclick="changeFont('Courier New')">Courier New</button>

</div>

<script>

function changeColor(color) {

document.getElementById('paragraph').style.color = color;

}

function changeFontSize(fontSize) {

document.getElementById('paragraph').style.fontSize = fontSize + 'px';

}

function changeFont(font) {

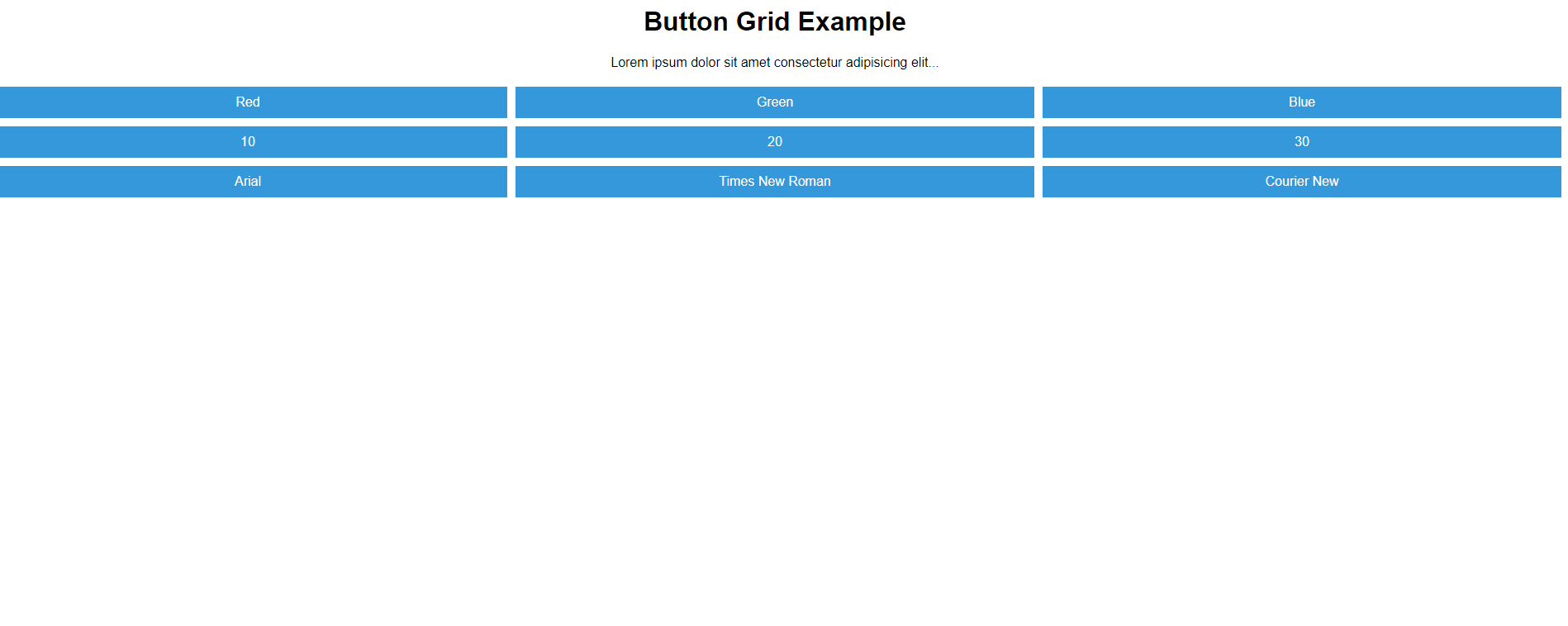
document.getElementById('paragraph').style.fontFamily = font;

}

</script>

</body>

</html>



1. Create a form that takes data about a pet. The form must be well designed and should accept the Pet’s Name, Age, Weight, Type, and what it likes most. At the submission of this form create a Pet object in JavaScript filled with these values and log that object and equivalent JSON on the console.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Pet Data Form</title>

<style>

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f2f2f2;

}

.container {

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #fff;

border-radius: 5px;

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

}

.form-group {

margin-bottom: 15px;

}

label {

font-weight: bold;

display: block;

}

input[type="text"],

input[type="number"] {

width: 100%;

padding: 8px;

border: 1px solid #ccc;

border-radius: 3px;

}

button {

background-color: #007bff;

color: #fff;

border: none;

padding: 10px 20px;

border-radius: 3px;

cursor: pointer;

}

</style>

</head>

<body>

<div class="container">

<h2>Pet Data Form</h2>

<form id="petForm">

<div class="form-group">

<label for="name">Name:</label>

<input type="text" id="name" name="name" required>

</div>

<div class="form-group">

<label for="age">Age:</label>

<input type="number" id="age" name="age" required>

</div>

<div class="form-group">

<label for="weight">Weight (kg):</label>

<input type="number" id="weight" name="weight" required>

</div>

<div class="form-group">

<label for="type">Type:</label>

<input type="text" id="type" name="type" required>

</div>

<div class="form-group">

<label for="likes">What it likes most:</label>

<input type="text" id="likes" name="likes" required>

</div>

<button type="button" onclick="submitform()">Submit</button>

</form>

</div>

<script>

/\* document.getElementById("petForm").addEventListener("submit", function (event) {

event.preventDefault(); // Prevent the form from actually submitting.

const name = document.getElementById("name").value;

const age = document.getElementById("age").value;

const weight = document.getElementById("weight").value;

const type = document.getElementById("type").value;

const likes = document.getElementById("likes").value;

// Create a JavaScript object

const pet = {

name: name,

age: parseInt(age),

weight: parseFloat(weight),

type: type,

likes: likes

};

// Log the object and equivalent JSON to the console

console.log("Pet Object:", pet);

console.log("JSON Equivalent:", JSON.stringify(pet));

// You can further process the 'pet' object as needed.

});\*/

/\*function formSubmit(){

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

const formData = new formData(form);

var pet={}

formData.array.forEach((value,key) => {

pet[key]=value;

});

console.log(pet)

}\*/

function submitform() {

event.preventDefault();

var form = document.getElementById("petForm");

// console.log(form);

var formchild = form.firstElementChild;

// console.log(formchild);

var childinput = formchild.lastElementChild;

// console.log(childinput);

var id = childinput.id;

var value = document.getElementById(id).value;

var pet = {};

pet[0] = value;

//console.log(pet);

for (var i = 1; i < form.childElementCount - 1; i++) {

formchild = formchild.nextElementSibling;

childinput = formchild.lastElementChild;

var id = childinput.id;

var value = document.getElementById(id).value;

pet[i] = value;

}

console.log(pet);

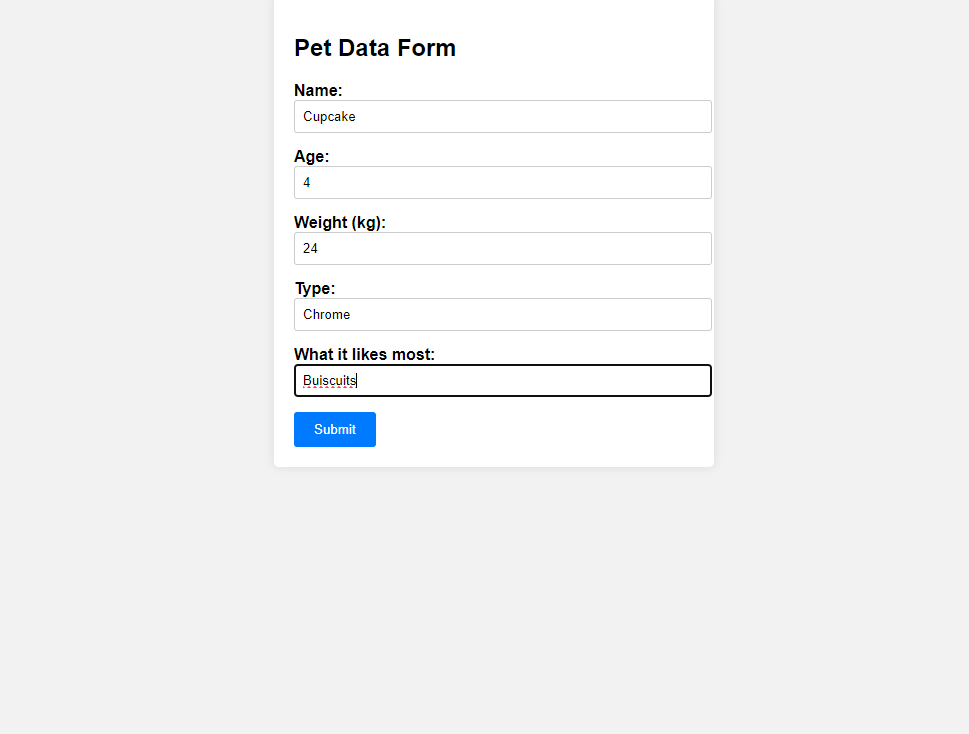
console.log(JSON.stringify(pet))

}

</script>

</body>

</html>



1. Store JSON data of few pets that you created in previous practical in a JSON file (copy from console output of previous program to a .json file). Using AJAX, load data from the file and display it in a presentable way using HTML and CSS.

**Code**

Index

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

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

<title>Pets Information</title>

<style>

body {

font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;

background-color: #f2f2f2;

margin: 0;

padding: 20px;

text-align: center;

}

h1 {

color: #333;

}

.pet-container {

display: flex;

justify-content: center;

flex-wrap: wrap;

gap: 20px;

margin-top: 20px;

}

.pet-card {

border: 1px solid #ddd;

border-radius: 8px;

overflow: hidden;

box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);

width: 250px;

transition: transform 0.3s ease-in-out, box-shadow 0.3s ease-in-out;

}

.pet-card:hover {

transform: scale(1.05);

box-shadow: 0 8px 16px rgba(0, 0, 0, 0.2);

}

.pet-card h2 {

color: #fff; /\* White text color for contrast \*/

margin-bottom: 0;

}

.pet-card p {

color: #eee; /\* Light gray text color for contrast \*/

margin: 5px 0;

}

</style>

</head>

<body>

<h1>Pets Information</h1>

<div id="petData" class="pet-container">

<!-- Pets data will be dynamically added here -->

</div>

<script>

// Function to generate a random vibrant color

function getRandomColor() {

var letters = '0123456789ABCDEF';

var color = '#';

for (var i = 0; i < 6; i++) {

color += letters[Math.floor(Math.random() \* 16)];

}

return color;

}

// Function to load JSON data using AJAX

function loadPetsData() {

var xhr = new XMLHttpRequest();

xhr.onreadystatechange = function() {

if (xhr.readyState === 4 && xhr.status === 200) {

var pets = JSON.parse(xhr.responseText);

displayPets(pets);

}

};

xhr.open("GET", "data.json", true);

xhr.send();

}

// Function to display pets data

function displayPets(pets) {

var petContainer = document.getElementById("petData");

pets.forEach(function(pet) {

var petCard = document.createElement("div");

petCard.className = "pet-card";

petCard.innerHTML = `

<h2>${pet.name}</h2>

<p>Age: ${pet.age}</p>

<p>Weight: ${pet.weight}</p>

<p>Type: ${pet.type}</p>

<p>Likes: ${pet.likes}</p>

`;

petCard.style.backgroundColor = getRandomColor();

petContainer.appendChild(petCard);

});

}

// Load pets data when the page is loaded

window.onload = function() {

loadPetsData();

};

</script>

</body>

</html>

**Data**

[

{

"name": "Cupcake",

"age": "4",

"weight": "24",

"type": "Chrome",

"likes": "Buiscuits"

},

{

"name": "tp2",

"age": "10",

"weight": "20",

"type": "gs",

"likes": "sleeping"

},

{

"name": "tp3",

"age": "10",

"weight": "20",

"type": "gs",

"likes": "sleeping"

},

{

"name": "tp4",

"age": "10",

"weight": "20",

"type": "gs",

"likes": "sleeping"

},

{

"name": "tp5",

"age": "10",

"weight": "20",

"type": "gs",

"likes": "sleeping"

},

{

"name": "tp1",

"age": "10",

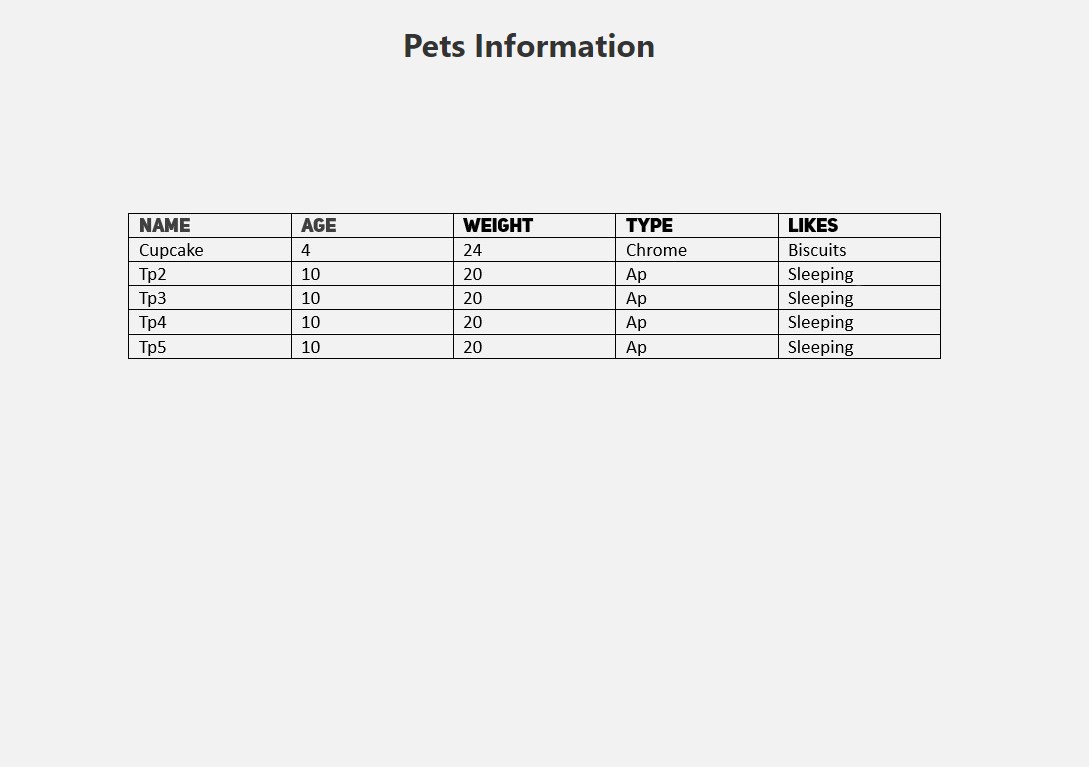
"weight": "20",

"type": "gs",

"likes": "sleeping"

}

]



1. Create a plain HTML page for B.Sc. Hons CS course, mentioning details like fee, eligibility criteria, papers with names and credits, and future possibilities after the course. A button for styling should be there at bottom of the page. On clicking on this button JavaScript should redesign the complete page using jQuery in a nice presentable way.

<!DOCTYPE html>

<html>

<head>

<title>B.Sc. Hons Computer Science</title>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

</head>

<body>

<div class="container">

<h1>Bachelor of Science (B.Sc. Hons) in Computer Science</h1>

<h2>Course Details</h2>

<p><strong>Fee:</strong> &#8377 30000 per year</p>

<p><strong>Eligibility Criteria:</strong> Minimum 12th-grade education with a focus on mathematics and computer

science subjects.</p>

<p><strong>Papers with Names and Credits:</strong></p>

<ul>

<li>Introduction to Computer Science (4 credits)</li>

<li>Data Structures and Algorithms (4 credits)</li>

<li>Database Management (4 credits)</li>

<li>Web Development (4 credits)</li>

<li>Artificial Intelligence (4 credits)</li>

</ul>

<h2>Future Possibilities</h2>

<p>After completing this course, you can pursue various career options such as software development, data

analysis, network administration, and more.</p>

<p>Click the button below to style the page:</p>

<button id="styleButton">Style Page</button>

</div>

<script>

// jQuery function to style the page on button click

$(document).ready(function () {

$("#styleButton").click(function () {

$("body").css({

"font-family": "Arial , sans-serif",

"margin": "0",

"padding": "0",

"background-color": "#f2f2f2",

});

$(".container").css({

"max-width": "900px",

"margin": "0 auto",

"padding": "20px",

"background-color": "#fff",

"border-radius": "5px",

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

});

$("button").css({

"background-color": "#00ffc3",

"color": "#fff",

"border": "none",

"padding": "10px 20px",

"border-radius": "3px",

"cursor": "pointer",

});

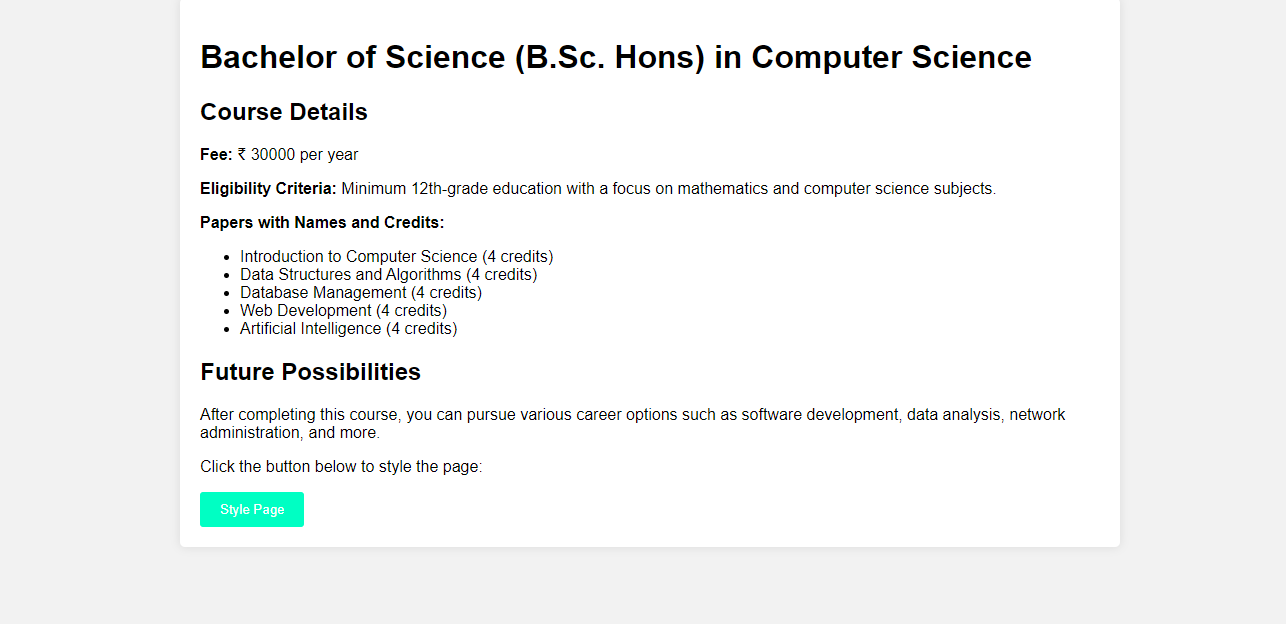
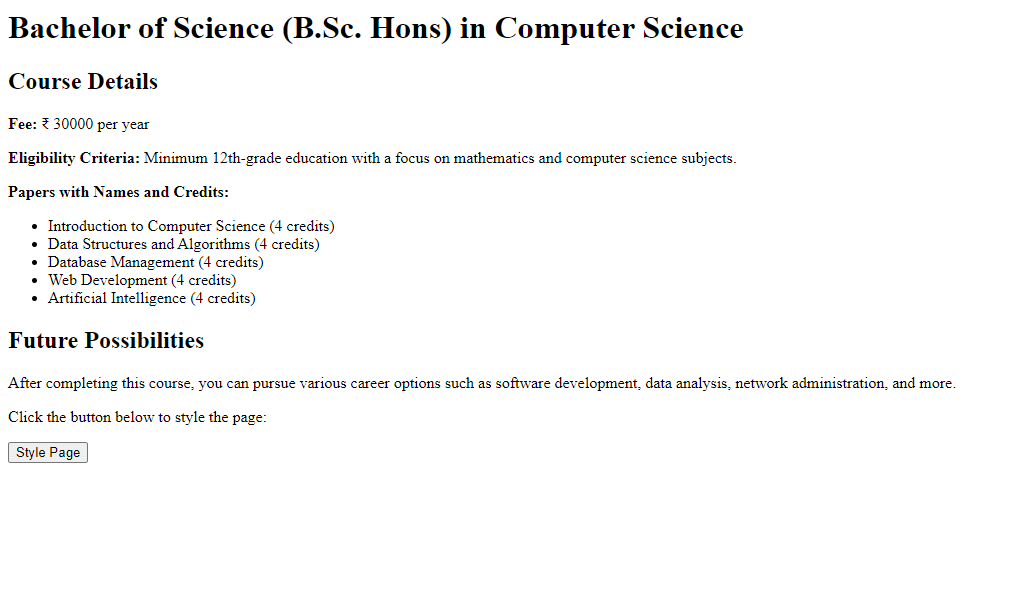
});

});

</script>

</body>

</html>



1. Create an HTML page for an image gallery which shows the use of BOOTSTRAP to rearrange and resize its contents on resizing the browser.

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

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

<title>Document</title>

<style>

.container1 {

max-width: 400px;

margin: 0 auto;

padding: 20px;

background-color: #fff;

border-radius: 5px;

box-shadow: 0 0 10px rgba(112, 37, 37, 0.1);

}

.image-container {

display: flex;

flex-wrap: wrap;

justify-content: space-evenly;

gap: 10px;

max-width: 100%;

margin: 0 auto;

padding: 20px;

border-radius: 5px;

background-color: rgb(0, 0, 0);

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

transition: background-color 0.5s;

}

.image-container img {

max-width: 100%;

height: auto;

transition: box-dow 0.5s;

}

</style>

<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>

<link

href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css"

rel="stylesheet"

integrity="sha384-T3c6CoIi6uLrA9TneNEoa7RxnatzjcDSCmG1MXxSR1GAsXEV/Dwwykc2MPK8M2HN"

crossorigin="anonymous"

/>

</head>

<body>

<div class="container1 container">

<input type="number" placeholder="enter number of imaages" id="input" />

</div>

<div class="image-container container" id="imagecontainer"></div>

<script>

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

var inele = document.getElementById("input");

var imgcon = document.getElementById("imagecontainer");

inele.addEventListener("input", function () {

event.preventDefault();

var inval = inele.value;

console.log(inval);

imgcon.innerHTML = "";

for (var i = 0; i < inval; i++) {

var image = document.createElement("img");

image.src = "https://picsum.photos/200?random=" + i;

image.alt = "Image " + (i + 1);

imgcon.appendChild(image);

image.addEventListener("mouseover", () => {

$('.image-container').css({ "background-color": "rgb(89, 91, 93)" });

$("img").css({

transform: "scale(0.9,0.9)",

"box-shadow": "5px 5px 10px rgba(0, 0, 0, 0.5)",

});

});

image.addEventListener("mouseout", () => {

$('.image-container').css({ "background-color": "rgb(0, 0, 0)" });

$("img").css({

transform: "scale(0.9,0.9)",

"box-shadow": "5px 5px 10px rgba(255, 255, 255, 0.5)",

});

});

}

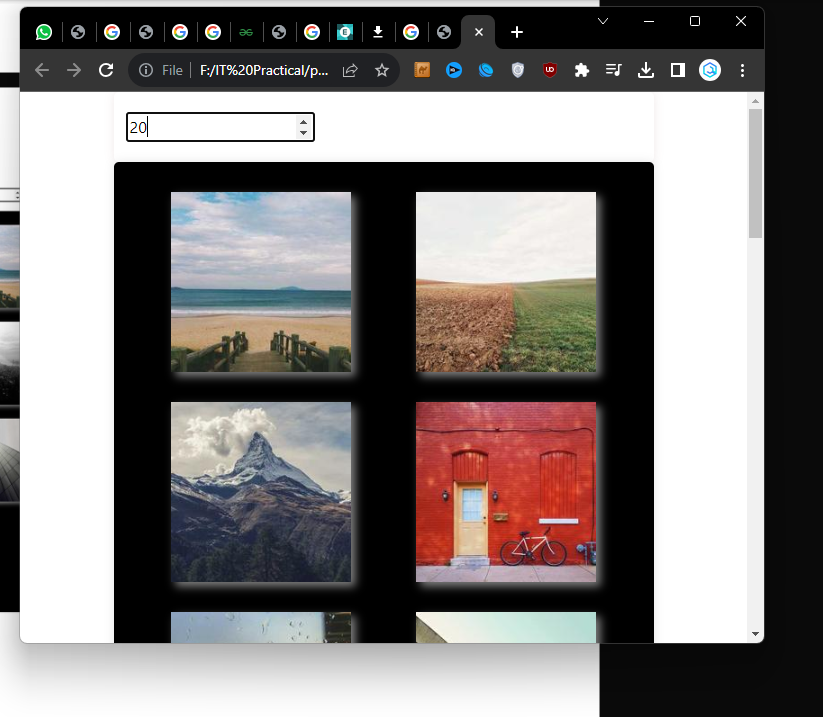
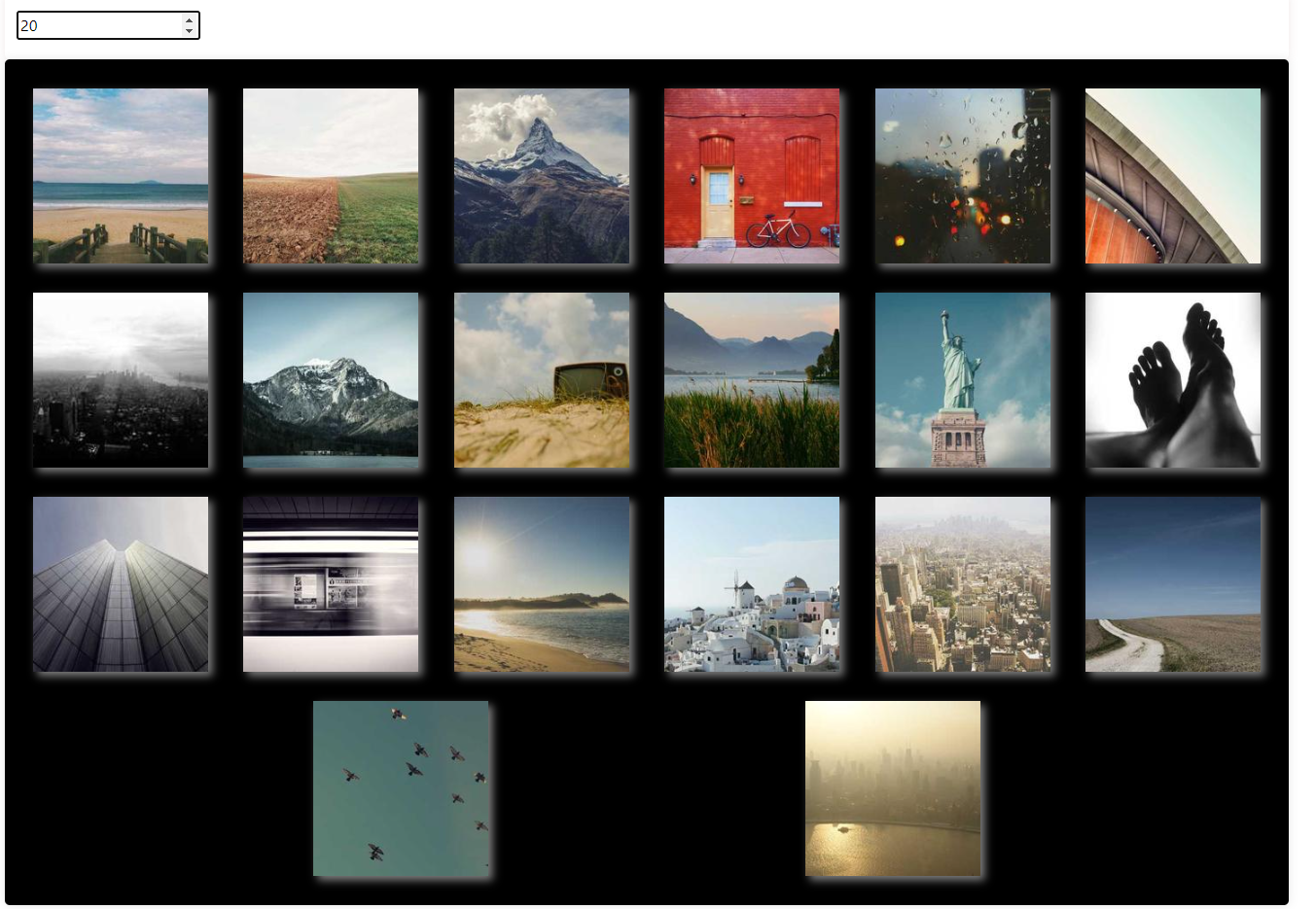
});

});

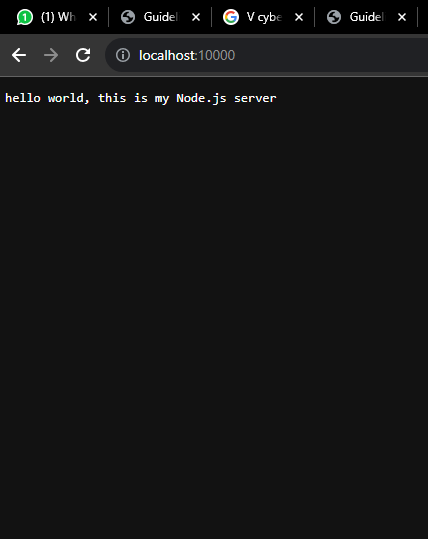
</script>

</body>

</html>



10. Create an HTTP server using Node.js which handles requests on port 10000 or a free port beyond 10000. Modify the server in such a way that opening localhost:10000 will display “Hello World, this is my Node.js server” on browser.



11. Create index.html file containing two forms for SignIn and SignUp. Submitting SignIn form should search for credentials in mysql database using server created in previous practical. On successful signin, a welcome page should be displayed. Submitting SignUp form should insert new entry for credentials in mysql database using server created in previous practical. On successful signup, user should be returned back to index.html.

**Index**

**<!DOCTYPE html>**

**<html lang="en">**

**<head>**

**<meta charset="UTF-8">**

**<meta http-equiv="X-UA-Compatible" content="IE=edge">**

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

**<title>SignInUp</title>**

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

**</head>**

**<body>**

**<main class="container" id="container">**

**<div class="form-container sign-up-container">**

**<form action="http://localhost:3000/signup" method="POST">**

**<h1>Sign Up</h1>**

**<input type="text" name="name" placeholder="Name" />**

**<input type="email" name="email" placeholder="Email" />**

**<input type="password" name="password" placeholder="Password" />**

**<a href="#">You agree to T&C</a>**

**<button type="submit">Sign Up</button>**

**</form>**

**</div>**

**<div class="form-container sign-in-container">**

**<form action="http://localhost:3000/signin" method="POST">**

**<h1>Sign in</h1>**

**<input type="email" name="email" placeholder="Email" />**

**<input type="password" name="password" placeholder="Password" />**

**<a href="#">Forgot your password?</a>**

**<button type="submit">Sign In</button>**

**</form>**

**</div>**

**<div class="overlay-container">**

**<div class="overlay">**

**<div class="overlay-panel overlay-left">**

**<h1>Already Have An Account? 👀</h1>**

**<p>Sign in to access your dashboard</p>**

**<button class="ghost" id="signIn">Sign In</button>**

**</div>**

**<div class="overlay-panel overlay-right">**

**<h1>Hey There 👋</h1>**

**<p>Create an account with us and get access to your dashboard</p>**

**<button class="ghost" id="signUp">Sign Up</button>**

**</div>**

**</div>**

**</div>**

**</div>**

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

**</body>**

**</html>**

**Script**

const signUpButton = document.getElementById('signUp');

const signInButton = document.getElementById('signIn');

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

signUpButton.addEventListener('click', () => {

  container.classList.add("right-panel-active");

});

signInButton.addEventListener('click', () => {

  container.classList.remove("right-panel-active");

});

**Server**

const http = require('http');

const url = require('url');

const fs = require('fs');

const path = require('path');

const querystring = require('querystring');

const mysql = require('mysql');

const cors = require('cors');

// Replace these details with your MySQL database configuration

const dbConfig = {

    host: 'localhost',

    user: 'root',

    password: '2004',

    database: 'details',

    insecureAuth: true,

  };

const connection = mysql.createConnection(dbConfig);

// Create a simple in-memory array to store user data

const users = [];

// Create a table for users in the database

connection.query(`

  CREATE TABLE IF NOT EXISTS users (

    id INT AUTO\_INCREMENT PRIMARY KEY,

    name VARCHAR(255) NOT NULL,

    email VARCHAR(255) NOT NULL,

    password VARCHAR(255) NOT NULL

  )

`);

const server = http.createServer((req, res) => {

  const parsedUrl = url.parse(req.url);

  const pathname = parsedUrl.pathname;

  if (pathname === '/' && req.method === 'GET') {

    const indexPath = path.join(\_\_dirname, 'public', 'index.html');

    fs.readFile(indexPath, 'utf8', (err, data) => {

      if (err) {

        res.writeHead(500, { 'Content-Type': 'text/plain' });

        res.end('Internal Server Error');

      } else {

        res.writeHead(200, { 'Content-Type': 'text/html' });

        res.end(data);

      }

    });

  } else if (pathname === '/signup' && req.method === 'POST') {

    let data = '';

    req.on('data', chunk => {

      data += chunk;

    });

    req.on('end', () => {

      const { name, email, password } = querystring.parse(data);

      // Check if the email is already registered

      if (users.some(user => user.email === email)) {

        res.writeHead(409, { 'Content-Type': 'application/json' });

        res.end(JSON.stringify({ message: 'Email is already registered' }));

      } else {

        // Insert the user into the in-memory array (and database in a real application)

        users.push({ name, email, password });

        connection.query('INSERT INTO users (name, email, password) VALUES (?, ?, ?)', [name, email, password], (error, results) => {

          if (error) {

            res.writeHead(500, { 'Content-Type': 'application/json' });

            res.end(JSON.stringify({ message: 'Internal Server Error' }));

          } else {

            res.writeHead(201, { 'Content-Type': 'application/json' });

            res.end(JSON.stringify({ message: 'User created successfully' }));

          }

        });

      }

    });

  } else if (pathname === '/signin' && req.method === 'POST') {

    let data = '';

    req.on('data', chunk => {

      data += chunk;

    });

    req.on('end', () => {

      const { email, password } = querystring.parse(data);

      // Check if the user exists

      const user = users.find(user => user.email === email && user.password === password);

      if (user) {

        res.writeHead(200, { 'Content-Type': 'application/json' });

        res.end(JSON.stringify({ message: 'Sign in successful', user }));

      } else {

        res.writeHead(401, { 'Content-Type': 'application/json' });

        res.end(JSON.stringify({ message: 'Invalid email or password' }));

      }

    });

  } else {

    res.writeHead(404, { 'Content-Type': 'application/json' });

    res.end(JSON.stringify({ message: 'Not Found' }));

  }

});

const port = 3000;

server.listen(port, () => {

  console.log(`Server is running on port ${port}`);

});

**Style**

@import url('https://fonts.googleapis.com/css?family=Montserrat:400,800');

\* {

    box-sizing: border-box;

}

body {

    background: #f6f5f7;

    display: flex;

    justify-content: center;

    align-items: center;

    flex-direction: column;

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

    height: 100vh;

    margin: 0;

    padding: 0;

}

h1 {

    font-weight: bold;

    margin: 0;

}

h2 {

    text-align: center;

}

p {

    font-size: 14px;

    font-weight: 100;

    line-height: 20px;

    letter-spacing: 0.5px;

    margin: 20px 0 30px;

}

span {

    font-size: 12px;

}

a {

    color: #333;

    font-size: 14px;

    text-decoration: none;

    margin: 15px 0;

}

button {

    border-radius: 20px;

    border: 1px solid #07070A;

    background-color: #07070A;

    color: #ffffff;

    font-size: 12px;

    font-weight: bold;

    padding: 12px 45px;

    letter-spacing: 1px;

    text-transform: uppercase;

    transition: transform 80ms ease-in;

}

button:active {

    transform: scale(0.95);

}

button:focus {

    outline: none;

}

button.ghost {

    background-color: transparent;

    border-color: #ffffff;

}

form {

    background-color: #ffffff;

    display: flex;

    align-items: center;

    justify-content: center;

    flex-direction: column;

    padding: 0 50px;

    height: 100%;

    text-align: center;

}

form h1 {

    margin-bottom: 0.33em;

}

input {

    background-color: #eee;

    border: none;

    padding: 12px 15px;

    margin: 8px 0;

    width: 100%;

}

.container {

    background-color: #fff;

    box-shadow: 0 14px 28px rgba(0,0,0,0.22), 0 10px 10px rgba(0,0,0,0.22);

    position: relative;

    overflow: hidden;

    width: 768px;

    max-width: 100%;

    min-height: 480px;

    border-radius: 100px;

}

.form-container {

    position: absolute;

    top: 0;

    height: 100%;

    transition: all 0.6s ease-in-out;

}

.sign-in-container {

    left: 0;

    width: 50%;

    z-index: 2;

}

.container.right-panel-active .sign-in-container {

    transform: translateX(100%);

}

.sign-up-container {

    left: 0;

    width: 50%;

    opacity: 0;

    z-index: 1;

}

.container.right-panel-active .sign-up-container {

    transform: translateX(100%);

    opacity: 1;

    z-index: 5;

    animation: show 0.6s;

}

@keyframes show {

    0%, 49.99% {

        opacity: 0;

        z-index: 1;

    }

    50%, 100% {

        opacity: 1;

        z-index: 5;

    }

}

.overlay-container {

    position: absolute;

    top: 0;

    left: 50%;

    width: 50%;

    height: 100%;

    overflow: hidden;

    transition: transform 0.6s ease-in-out;

    z-index: 100;

}

.container.right-panel-active .overlay-container{

    transform: translateX(-100%);

}

.overlay {

    background: #24272B;

    background: linear-gradient(to right, #07070A, #24272B);

    background-repeat: no-repeat;

    background-size: cover;

    background-position: 0 0;

    color: #ffffff;

    position: relative;

    left: -100%;

    height: 100%;

    width: 200%;

    transform: translateX(0);

    transition: transform 0.6s ease-in-out;

}

.container.right-panel-active .overlay {

    transform: translateX(50%);

}

.overlay-panel {

    position: absolute;

    display: flex;

    align-items: center;

    justify-content: center;

    flex-direction: column;

    padding: 0 40px;

    text-align: center;

    top: 0;

    height: 100%;

    width: 50%;

    transform: translateX(0);

    transition: transform 0.6s ease-in-out;

}

.overlay-left {

    transform: translateX(-20%);

}

.container.right-panel-active .overlay-left {

    transform: translateX(0);

}

.overlay-right {

    right: 0;

    transform: translateX(0);

}

.container.right-panel-active .overlay-right {

    transform: translateX(20%);

}

.social-container {

    margin: 20px 0;

}

.social-container a {

    border: 1px solid #DDDDDD;

    border-radius: 50%;

    display: inline-flex;

    justify-content: center;

    align-items: center;

    margin: 0 5px;

    height: 40px;

    width: 40px;

}

footer {

    background-color: #222;

    color: #fff;

    font-size: 14px;

    bottom: 0;

    position: fixed;

    left: 0;

    right: 0;

    text-align: center;

    z-index: 999;

}

footer p {

    margin: 10px 0;

}

footer i {

    color: red;

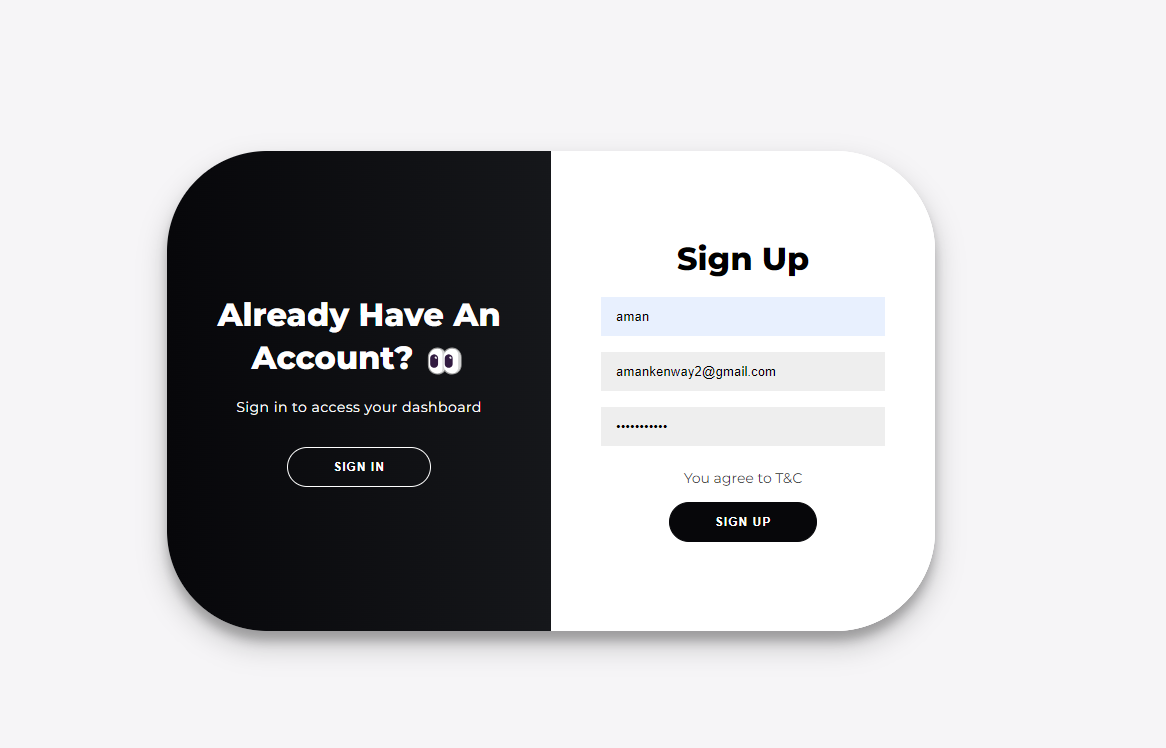
}

footer a {

    color: #3c97bf;

    text-decoration: none;

}

****