|  |  |
| --- | --- |
| Practical: 11 | *// Write a program that demonstrates asynchronous behavior using a callback*  *// function. For example, create a function that simulates fetching data from an*  *// API and invokes a callback with the fetched data.*  *// Simulate an asynchronous API request*  function fetchDataFromAPI(callback) {      setTimeout(function () {        const data = {          userId: 114,          id: '21CE114',          title: 'Practical 11',          body: 'This is some sample data fetched from an API.',        };        callback(data);      }, 2000); *// Simulate a 2-second delay*    }    *// Callback function to handle the fetched data*    function handleData(data) {      console.log('Data received:', data);    }    *// Calling the fetchDataFromAPI function with the callback*    console.log('Fetching data...');    fetchDataFromAPI(handleData);    console.log('Request sent asynchronously.'); |
| Practical: 12 | *// Create a program that reads a file asynchronously using callbacks and displays*  *// its contents.*  const fs = require('fs');  *// Function to read a file asynchronously and display its contents*  function readFileAsync(filePath, callback) {    fs.readFile(filePath, 'utf8', (err, data) => {      if (err) {        callback(err);      } else {        callback(null, data);      }    });  }  const filePath = 'Data.txt';  readFileAsync(filePath, (err, data) => {    if (err) {      console.error('Error reading file:', err);    } else {      console.log('File contents:');      console.log(data);    }  }); |
| Practical: 13 | *// Write a program that uses Promises to handle asynchronous operations. For*  *// example, create a function that returns a Promise to fetch data from an API and*  *// resolve it with the fetched data.*  *// Implement error handling using Promises by rejecting a Promise with an error*  *// message in case of failure.*  *// Function that simulates fetching data from an API*  function fetchDataFromSimulatedAPI() {      return new Promise((resolve, reject) => {  *// Simulate a delay like the time it takes to fetch data in real api*        setTimeout(() => {          const Data = {            id: 114,            name: '21CE114',            description: 'Practical 13',          };          resolve(Data);        }, 2000); *// Simulated delay of 2 seconds*      });    }      fetchDataFromSimulatedAPI().then(data => {        console.log('Data fetched successfully:', data);      })      .catch(error => {        console.error('Error in fetching the Data:', error);      }); |
| Practical: 14 | *// Convert a Promise-based asynchronous function into an async/await style*  *// function. For example, rewrite a function that fetches data from an API using*  *// async/await.*  *// Write a program that utilizes multiple async/await functions to fetch data from*  *// different APIs sequentially and display the combined results.*  *// Simulate fetching data from API1*  async function fetchDataFromAPI1() {      try {  *// Simulated data*        const data = { message: 'ID: 21CE114' };        return data;      } catch (error) {        throw new Error('Error fetching data from API1: ' + error.message);      }    }    *// Simulate fetching data from API2*    async function fetchDataFromAPI2() {      try {  *// Simulated data*        const data = { message: 'Practical 14' };        return data;      } catch (error) {        throw new Error('Error fetching data from API2: ' + error.message);      }    }    *//fetch data from different APIs sequentially*    async function fetchAndDisplayCombinedData() {      try {        const data1 = await fetchDataFromAPI1();        const data2 = await fetchDataFromAPI2();  *// Combine and displaying the result*        const combinedData = { data1, data2 };        console.log('Combined Data:', combinedData);      } catch (error) {        console.error(error.message);      }    }      fetchAndDisplayCombinedData(); |