**Javascript Assignments**

**Javascript Fundamentals**

**1: Reverse a String**

**Problem: Write a function to reverse a given string.**

**2: Find the Largest Number in an Array**

**Problem: Write a function to find the largest number in an array.**

**3: Check if a String is a Palindrome**

**Problem: Write a function to check if a given string is a palindrome.**

**4: Filter Array of Objects**

**Problem: Write a function to filter an array of objects based on a specified condition.**

**People array with name and age and where age is greater than 18.**

**5: Sum of All Numbers in an Array**

**Problem: Write a function to calculate the sum of all numbers in an array.**

**6: Remove Duplicates from an Array**

**Problem: Write a function to remove duplicates from an array.**

**7: Flatten a Nested Array**

**Problem: Write a function to flatten a nested array.**

**// Example usage:**

**var nestedArray = [1, [2, 3], [4, [5, 6]]].**

**var result = flattenArray(nestedArray).**

**console.log(result); // Output: [1, 2, 3, 4, 5, 6]**

**8: Find Intersection of Two Arrays**

**Problem: Write a function to find the intersection of two arrays.**

**// Example usage:**

**var array1 = [1, 2, 3, 4];**

**var array2 = [3, 4, 5, 6];**

**var result = findIntersection(array1, array2);**

**console.log(result); // Output: [3, 4]**

**9: Convert an Object to an Array of Key-Value Pairs**

**Problem: Write a function to convert an object into an array of key-value pairs.**

**// Example usage:**

**var obj = {name: 'John', age: 30};**

**var result = objectToArray(obj);**

**console.log(result); // Output: [['name', 'John'], ['age', 30]]**

**10: Sort an Array of Objects by a Property**

**Problem: Write a function to sort an array of objects by a specified property.**

**Consider People array with object having properties like name and age,**

**Javascript Object Oriented Programming**

**1: Creating and Using Objects**

**Create a Simple Object:**

**- Create an object `car` with properties `make`, `model`, and `year`.**

**- Add a method `getCarInfo` that returns a string with the car's information.**

**2: Prototypal Inheritance**

**Animal and Dog Classes:**

**- Create a constructor function `Animal` with properties `name` and `sound`.**

**- Add a method `makeSound` to the `Animal` prototype that logs the sound.**

**- Create a constructor function `Dog` that inherits from `Animal`.**

**- Add a method `fetch` to the `Dog` prototype.**

**3: Encapsulation and Private Variables**

**Bank Account:**

**- Create a constructor function `BankAccount` with a private variable `balance`.**

**- Add methods `deposit`, `withdraw`, and `getBalance` to manage the balance.**

**4: Polymorphism**

**Shapes:**

**- Create a base constructor function `Shape` with a method `getArea` that returns 0.**

**- Create derived constructor functions `Circle` and `Rectangle` that override the `getArea` method to calculate the area of the shape.**

**5: Inheritance with Method Overriding**

**Vehicle and Car:**

**- Create a constructor function `Vehicle` with properties `make`, `model`, and `year`.**

**- Add a method `getDetails` to the `Vehicle` prototype that returns a string with the vehicle's details.**

**- Create a constructor function `Car` that inherits from `Vehicle` and adds a property `doors`.**

**- Override the `getDetails` method in the `Car` prototype to include the number of doors.**

**6: Creating and Using Static Methods**

**Math Utilities:**

**- Create a constructor function `MathUtil`.**

**- Add a static method `add` that takes two numbers and returns their sum.**

**- Add a static method `subtract` that takes two numbers and returns their difference.**

**- Add a static method `multiply` that takes two numbers and returns their product.**

**- Add a static method `divide` that takes two numbers and returns their quotient.**

**7: Using Getters and Setters**

**Book:**

**- Create a constructor function `Book` with properties `title`, `author`, and `year`.**

**- Add a getter method `getSummary` that returns a summary string of the book.**

**- Add a setter method `setYear` that updates the year and logs a message when the year is set.**

**8: Composition Over Inheritance**

**User with Address:**

**- Create a constructor function `User` with properties `username` and `password`.**

**- Create a constructor function `Address` with properties `street`, `city`, and `country`.**

**- Add an `address` property to the `User` constructor that holds an `Address` instance.**

**- Add a method `getAddress` to the `User` prototype that returns the user's address as a formatted string.**

**9: Implementing Polymorphism**

**Shape and Specific Shapes:**

**- Create a constructor function `Shape` with a method `getArea` that returns 0.**

**- Create a constructor function `Square` that inherits from `Shape` and has a property `sideLength`.**

**- Override the `getArea` method in `Square` to return the area of the square.**

**- Create a constructor function `Triangle` that inherits from `Shape` and has properties `base` and `height`.**

**- Override the `getArea` method in `Triangle` to return the area of the triangle.**

**10: Using Mixins**

**Flyable and Swimmable Mixins :**

**- Create a mixin `Flyable` that adds a method `fly` to an object.**

**- Create a mixin `Swimmable` that adds a method `swim` to an object.**

**- Create a constructor function `Duck` that uses both mixin**

**Function Overloading**

1. **Write an overloaded function in javascript that would take care of**

**Addition of 2 numbers**

**Addition of 3 numbers and**

**Addition of 4 numbers.**

1. **Write an overloaded function in javascript that would take care of**

* **Function with two strings**
* **Function with two numbers**
* **Function with three numbers**
* **Function with one String and one number**
* **And common implementation for the rest of all combinations.**

**Dunder Proto, Object.create and Object.setPrototypeOf and Object.getPrototypeOf**

1. **Create Pseudo classical Inheritance by using Object.create and Object.setPrototypeOf for**

**Human – Base / Super Class**

**name – string**

**Prototype Level Method**

**Speak**

**Introduction**

**Student**

**College – string**

**Courses – string []**

**Prototype Level Method**

**Introduction (overridden)**

**takeExams**

1. **Create a Class Hierarchy between**

**Person – Base / Super**

**Developer - Derived / Sub**

**Display Both objects / classes prototype, \_\_proto\_\_ and getPrototypeOf**

**Module Pattern**

1. **Create a Shopping Cart Module, which would allow user to deal with cart data,**

**Product in Shopping Cart would be with following details.**

**productId, name, price, quantity**

**Methods provided by Module are,**

**addItem – Add Item would add one product item, if again called for the same product**

**id, it would increase the quantity.**

**removeItem – Remove product (all quantities by product id)**

**getAllItems – Get all products details.**

**getTotalPrice - Total Price of all items in the cart**

**Runtime Type Information**

1. **Go for Base class Vehicle.**

**function Vehicle (make, model) {**

**this.make = make;**

**this.model = model;**

**}**

**And 2 derived classes**

**function Car (make, model) {**

**this.wheels = 4;**

**}**

**function Bike (make, model) {**

**this.wheels = 2;**

**}**

**Do the inheritance with Vehicle and Car as well as on Vehicle and Bike**

**Instantiate Car and Bike**

* **Use type of to check the type of the Car and Bike Object**
* **Use instance of to check Car Object is of type Vehicle, Car, or Bike**
* **Use instance of to check Bike Object is of type Vehicle, Car, or Bike**

**Use constructor property to see whether,**

* **CarObject.constructor property pointing to also check whether,**
* **BikeObject.constructor property pointing to**

**Also check,**

* **getPrototypeOf(carObj) is equal to Car.Prototype and**
* **getPrototypeOf(bikeObj) is equal to Bike.Prototype**

**DOM Manipulation**

**Assignment 1: Changing Text Content**

**Write a JavaScript function that changes the text content of a specific HTML element when a button is clicked.**

**HTML**

<!DOCTYPE html>

<html>

<head>

    <title>Change Text Content</title>

</head>

<body>

    <p id="text">Original text</p>

    <button onclick="changeText()">Change Text</button>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 2: Adding New Elements**

**Write a JavaScript function that adds a new list item to an unordered list when a button is clicked.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Add List Item</title>

</head>

<body>

    <ul id="myList">

        <li>Item 1</li>

        <li>Item 2</li>

    </ul>

    <button onclick="addListItem()">Add Item</button>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 3: Removing Elements**

**Write a JavaScript function that removes the last list item from an unordered list when a button is clicked.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Remove List Item</title>

</head>

<body>

    <ul id="myList">

        <li>Item 1</li>

        <li>Item 2</li>

        <li>Item 3</li>

    </ul>

    <button onclick="removeListItem()">Remove Last Item</button>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 4: Changing Styles**

**Write a JavaScript function that changes the background color of a div element when a button is clicked.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Change Background Color</title>

    <style>

        #colorBox {

            width: 100px;

            height: 100px;

            background-color: lightgray;

        }

    </style>

</head>

<body>

    <div id="colorBox"></div>

    <button onclick="changeColor()">Change Color</button>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 5: Form Validation**

**Write a JavaScript function to validate a form. The form should contain an input field for an email address. The function should check if the input is a valid email address and display an appropriate message.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Form Validation</title>

</head>

<body>

    <form id="myForm" onsubmit="return validateForm()">

        <label for="email">Email:</label>

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

        <input type="submit" value="Submit">

    </form>

    <p id="message"></p>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 6: Toggling Visibility**

**Write a JavaScript function that toggles the visibility of a div element when a button is clicked.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Toggle Visibility</title>

    <style>

        #toggleDiv {

            width: 100px;

            height: 100px;

            background-color: lightblue;

        }

    </style>

</head>

<body>

    <div id="toggleDiv"></div>

    <button onclick="toggleVisibility()">Toggle Visibility</button>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 7: Updating Attributes**

**Write a JavaScript function that updates the `src` attribute of an image when a button is clicked.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Update Image Source</title>

</head>

<body>

    <img id="myImage" src="image1.jpg" alt="Image">

    <button onclick="changeImage()">Change Image</button>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 8: Event Listener**

**Write a JavaScript function that changes the text of a paragraph when a mouseover event occurs on a div element.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Event Listener</title>

</head>

<body>

    <div id="hoverDiv" style="width:100px; height:100px; background-color: lightgreen;"></div>

    <p id="hoverText">Hover over the box to change this text.</p>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 9: Creating a Modal**

**Write a JavaScript function that displays a modal when a button is clicked. The modal should contain some text and a close button.**

HTML

<!DOCTYPE html>

<html>

<head>

    <title>Create Modal</title>

    <style>

        #myModal {

            display: none;

            position: fixed;

            top: 50%;

            left: 50%;

            transform: translate(-50%, -50%);

            width: 200px;

            padding: 20px;

            background-color: white;

            border: 1px solid black;

        }

    </style>

</head>

<body>

    <button onclick="openModal()">Open Modal</button>

    <div id="myModal">

        <p>This is a modal.</p>

        <button onclick="closeModal()">Close</button>

    </div>

    <script src="script.js"></script>

</body>

</html>

**JavaScript (script.js):**

**Assignment 10: Create a New Element**

**Create a new `div` element, set its text content to "Hello, World!", and append it to the body.**

**Hint: Use `document.createElement`, `textContent`, and `appendChild`.**

**Assignment 11: Change an Element's Style**

**Create a button that, when clicked, changes the background color of the page to a random color.**

**Hint: Use `document.body.style.backgroundColor` and `Math.random`.**

**Assignment 12: Add and Remove Classes**

**Create an input field and a button. When the button is clicked, add a class to the input field that changes its border color. Clicking the button again should remove the class.**

**Hint: Use `classList.add` and `classList.remove`.**

**Assignment 13: Create a List and Add Items**

**Create an unordered list and a button. When the button is clicked, add a new list item with text "Item X" (where X is the number of the item).**

**Hint: Use `document.createElement`, `textContent`, and `appendChild`.**

**Assignment 14: Create a Table**

**Create a table with 3 rows and 3 columns and fill it with numbers 1 through 9.**

**Hint: Use nested loops and `document.createElement`.**

**Assignment 15: Clone and Modify an Element**

**Create a button and a paragraph. When the button is clicked, clone the paragraph, change its text, and append the clone to the body.**

**Hint: Use `cloneNode`.**

**AJAX Assignments**

**Assignment 1: Simple GET Request**

**Write a JavaScript function to make a GET request to a public API and display the response in an HTML element.**

**//** [**https://jsonplaceholder.typicode.com/posts/1**](https://secure-web.cisco.com/1vrBWB84FqUHKKy-gpMeOnB6HWGg6UWtoR6TCjFt5WtBdjKC4g1u10CfmbtkX7ZrebhXkI4-KcN6kVF4NsePcK8ZJvGpDYcqR8SC0fmHFtEoo5srWzYc7dbDdJCoTEpddt0_ArdDLQ5vHgzm-jtqJElRIEQxOyfsy-Dnhua7SmoLQ2zlnSvq0C21_iuxcTLZ4wTI7hTrz4hWTxlI3t9cwA7N_gWFfOq8Fp2YtfgLCey3UT8e4rzFunpYeM7R2jMnCtSoxzQTakPs_xepRSlGJ-LSh7Y66XPv10DM6vOKHBCWp2x85V1geYZGAjLs6TMJchDBVqtthnfz_bUmR7K7rIPOqx8pA1cP7Gg7-jig74YtpwjwTOGaFfanpNcGFuumArxQslnFf4MgVqm-dLpwICA/https%3A%2F%2Fjsonplaceholder.typicode.com%2Fposts%2F1)

**\*\*HTML\*\*:**

**```html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Simple GET Request</title>**

**</head>**

**<body>**

**<button onclick="makeGetRequest()">Fetch Data</button>**

**<p id="response"></p>**

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

**</body>**

**</html>**

**```**

**\*\*JavaScript\*\* (script.js):**

**Assignment 2: Handling JSON Response**

**Write a JavaScript function to make a GET request to a public API, parse the JSON response, and display specific data in HTML elements.**

**//** [**https://jsonplaceholder.typicode.com/posts/1**](https://secure-web.cisco.com/1vrBWB84FqUHKKy-gpMeOnB6HWGg6UWtoR6TCjFt5WtBdjKC4g1u10CfmbtkX7ZrebhXkI4-KcN6kVF4NsePcK8ZJvGpDYcqR8SC0fmHFtEoo5srWzYc7dbDdJCoTEpddt0_ArdDLQ5vHgzm-jtqJElRIEQxOyfsy-Dnhua7SmoLQ2zlnSvq0C21_iuxcTLZ4wTI7hTrz4hWTxlI3t9cwA7N_gWFfOq8Fp2YtfgLCey3UT8e4rzFunpYeM7R2jMnCtSoxzQTakPs_xepRSlGJ-LSh7Y66XPv10DM6vOKHBCWp2x85V1geYZGAjLs6TMJchDBVqtthnfz_bUmR7K7rIPOqx8pA1cP7Gg7-jig74YtpwjwTOGaFfanpNcGFuumArxQslnFf4MgVqm-dLpwICA/https%3A%2F%2Fjsonplaceholder.typicode.com%2Fposts%2F1)

**\*\*HTML\*\*:**

**```html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Handling JSON Response</title>**

**</head>**

**<body>**

**<button onclick="fetchJsonData()">Fetch JSON Data</button>**

**<p id="title"></p>**

**<p id="body"></p>**

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

**</body>**

**</html>**

**```**

**\*\*JavaScript\*\* (script.js):**

**Assignment 3: Error Handling**

**Write a JavaScript function to make a GET request and handle potential errors by displaying an error message.**

**//**[https://jsonplaceholder.typicode.com/invalid-url](https://secure-web.cisco.com/1YPJhmnZ3Z2AD-N5unxJMdJgFUAU-el3RClumrpW3XLEK_88zbB0hQsH10__2lx82u0wm-UN2xNQUQJdJztWnLfJFaMfnjS1nf44u21qdeDV8CZTUQQ2nTMiE1yYJqkCX6rvUkS4zPq0PkKWRB6zLLvaI7v5sEnwODtUAUwezhDtdz5cl8cEOOkHWPqvzqgyZRcZyYeuwb2mkBYY7nX5teqejsFkM3fJt6jXfb2hr7WYMFUe56t_4v66hghTnccQNTDt3yMqES0hxbqdlMk-kkpSFztdyKPJK0L2WMv3ev1sS-XB6GvNAQUHDA8o65FthUJFJtAkicU6cuCc-p-o2inDy1H0fUFIIT1LNScLYT5WzC5FVgSz5r5S3JXHZ-4rcp9QxnsC8scH_AkHKs-wG5w/https%3A%2F%2Fjsonplaceholder.typicode.com%2Finvalid-url)

**\*\*HTML\*\*:**

**```html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Error Handling</title>**

**</head>**

**<body>**

**<button onclick="fetchWithErrorHandling()">Fetch Data</button>**

**<p id="message"></p>**

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

**</body>**

**</html>**

**```**

**\*\*JavaScript\*\* (script.js):**

**Assignment 4: POST Request**

**Write a JavaScript function to make a POST request to a public API and display the response.**

**//**[https://jsonplaceholder.typicode.com/posts](https://secure-web.cisco.com/15hu8RDanzyicZ3WvXpn6o1_3dcQheXwBEDUu0cAw0MsZ0gt7L8x632tInOLdJJvSE_JQm1jGPd6bpyM1w-eYYajkyQvwIGvuuKPuIh63E3-rq8DPm3RzJX9odsGcerxi62vUZO9mWWPLJeHNTHDvgo04yxxy-r9J039ZlR7p_B9b6oSO3pOkobclEqpQfqlvg3t6kCJu2cW45Jng2PP4KBgoGZUSKQfnMTxeruCJyb7ZK92S1aDoI_d8BVHCDT44lNk1YE8v0DHWS34S056zcEPJ4vuTDz2SbSolWE8-ZgBHBNjjRWdPRmTyr8EQHfJi4gWnaJAir98rui0scjJswQcv1punVri7ZMLVxGTnfedeI5TIm-ABOVRxGEdshf4yNAtpZKcAJG5soWxO3uge6A/https%3A%2F%2Fjsonplaceholder.typicode.com%2Fposts)

**\*\*HTML\*\*:**

**```html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>POST Request</title>**

**</head>**

**<body>**

**<button onclick="makePostRequest()">Send Data</button>**

**<p id="postResponse"></p>**

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

**</body>**

**</html>**

**```**

**\*\*JavaScript\*\* (script.js):**

**Assignment 5: Updating the DOM with Multiple Data Items**

**Write a JavaScript function to make a GET request to fetch a list of items and display them in an HTML list.**

**//**[**https://jsonplaceholder.typicode.com/posts**](https://jsonplaceholder.typicode.com/posts)

**\*\*HTML\*\*:**

**```html**

**<!DOCTYPE html>**

**<html>**

**<head>**

**<title>Fetch List of Items</title>**

**</head>**

**<body>**

**<button onclick="fetchListOfItems()">Fetch Items</button>**

**<ul id="itemList"></ul>**

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

**</body>**

**</html>**

**```**

**\*\*JavaScript\*\* (script.js):**

**Assignment 6:** [**https://jsonplaceholder.typicode.com/posts**](https://jsonplaceholder.typicode.com/posts)

**Create CRUD requests to the above mentioned url**

**4 buttons**

**Get Posts**

**Create Post**

**Delete Post and**

**Update Post**

**Display the responses on console window.**

**Promise Assignments**

**1 Basic Promise**

**Write a function that returns a promise which resolves after a specified time with a success message.**

**Instructions:**

**1. Create a function `wait(ms)` that returns a promise.**

**2. The promise should resolve with a message "Completed after X milliseconds" where `X` is the input to the function.**

**3. Test the function by calling it with different time values and logging the result.**

**2: Promise Chaining**

**Create a series of functions that return promises and chain them together to perform a sequence of async operations.**

**Instructions:**

**1. Create three functions `task1() `, `task2()`, and `task3()` that return promises.**

**2. Each function should resolve after a certain time with a specific message (e.g., "Task 1 complete").**

**3. Chain these functions together so they execute in sequence.**

**4. Log each task’s completion message.**

**3: Handling Errors**

**Write a promise that simulates an error condition and handle it using `catch()`.**

**Instructions:**

**1. Create a function `errorProneTask()` that returns a promise.**

**2. The promise should randomly either resolve with a success message or reject with an error message.**

**3. Handle both the success and error cases using `then()` and `catch()`.**

**Assignment 4: Using `Promise.all()`**

**Create multiple promises and use `Promise.all()` to wait for all of them to complete.**

**Instructions:**

**1. Create three functions `taskA()`, `taskB()`, and `taskC()` that return promises.**

**2. Each function should resolve after a certain time with a specific message.**

**3. Use `Promise.all()` to execute these tasks concurrently and log the results once all tasks are complete.**

**Async Await Assignments**

**Assignment 1: Basic Async/Await**

**Understand how to use async/await with simple functions.**

**Instructions:**

**1. Create an async function named `fetchData`.**

**2. Inside the function, use `setTimeout` to simulate fetching data from a server. Use `await` to wait for 2 seconds.**

**3. After 2 seconds, log a message to the console saying, "Data fetched!".**

**4. Call the `fetchData` function and observe the output.**

**Assignment 2: Handling API Requests**

**Use async/await to handle real API requests.**

**Instructions:**

**1. Create an async function named `getUserData`.**

**2. Use the `fetch` API to make a GET request to `**[**https://jsonplaceholder.typicode.com/users/1`**](https://secure-web.cisco.com/1d12ZQ_iXZWsWwR6qzaHKfYAyaCVT9WrizwEjEJ-f9sNgUhZ1nGhrKAhWWRTXIJNw8YmkLsWuGGwNNaQd5UAQeH9GOERdMpAyPHW6XgK2x77_xdTgX30aNhkqSMDorxpTHVSZpStNCd3QJq1jp4aCAB0IIOgCbo3rR169E4IXySTzy65A4V9fWGMTm8IacA93nosmez_AW9l9JI_1LIn3JLXgjs--oyz4ElYAiyREi0sqQpca9rnBUNoojkg58ew2i1MRhVOmrAkdrR6dbWKxEPSivLL_CyuxfyqI_p_bebh3za3tMsgDJtTSiXls7keZCMLkfBejJmVdX8k1hZuXX5jouYeqBDXTLryHzu90KKn9_q3TRJAPH5W33nA4cfOSCf50bPCxgYaU-vnVEt8RBg/https%3A%2F%2Fjsonplaceholder.typicode.com%2Fusers%2F1%60)**.**

**3. Use `await` to wait for the response and convert it to JSON.**

**4. Log the user data to the console.**

**5. Call the `getUserData` function and observe the output.**

**Assignment 3: Error Handling**

**Handle errors in async functions using try/catch.**

**Instructions:**

**1. Modify the `getUserData` function from Assignment 2.**

**2. Wrap the `fetch` call and the subsequent code in a try/catch block.**

**3. In the catch block, log an error message to the console if the request fails.**

**4. Test the function by modifying the URL to an invalid one and observing the output.**

**Assignment 4: Chaining Async Functions**

**Chain multiple async functions together.**

**Instructions:**

**1. Create two async functions, `fetchPosts` and `fetchComments`.**

**2. In `fetchPosts`, use the `fetch` API to get posts from `**[**https://jsonplaceholder.typicode.com/posts`**](https://secure-web.cisco.com/1AC3_h8uJBhF1pEGRihYkQ92ZYLsFbUhzOK9VWJYv5Qe73D60QpAMD3mnlfxIitEL-yYBTsc-Y0-f63alf49ICinA0272lgjoRCQ0zxrj0wJKH4JMd9DfikUx4_LM6ak-3_wGCGAQqJL58-uXKtaQ62ppnxhBj5ADyofpusiPXGDaLxS7vst0xp5CBm6Y5KyINWTcbs5LaCAuQCYq64FZ-MN_89mRB_LkWit0jwNyRGl41IMoWyIs4G2Y8KQyF2n5rW9umqifFGTE888Id4ema_H2qhAgf8TWgewxHW-3ZLulnfmRdxCvjBcJQozPRZ2FWz_C7ui22rSM5HQv3m0iHqZZ8jPeqO3K42MNvuTOv7p6MVLKInP-OHH4l3cP6bXsjrWBAohQxUpMgZcFw2eUDw/https%3A%2F%2Fjsonplaceholder.typicode.com%2Fposts%60)**.**

**3. In `fetchComments`, use the `fetch` API to get comments for a specific post from `**[**https://jsonplaceholder.typicode.com/comments?postId=1`**](https://secure-web.cisco.com/1YKam51r8waAE-MhxGmeV7pRPc4u1egUfsy0WsQtm8YrSWF7S4wR0wPvVTNcYRpHuwg14tvQ26gE3HJVVi643fn12jiwWxWGAicHgrcRMRvB9TfRydUgZn6z6hX6YOBZtyyl6pmWeGx-g_9TzTnuNEQVFUu2c629h7IpHu8OWD2nwKgKUH0lSPA0f-d-qdWXvv7o3TluoUGQAZScnrh7xh4GNnw6y2mM40LFPrQv1iWefApyt-6QtOs7ZPCioA0yFIpvKg1LKgSiECebVtPYUDVqrRsqOcrjtSD-NrJrybTmsmbYqB9Hl2bvXLioYtKhI-CtakxDB7tbJH-6Le0zGcyFbN8XazL5IHl_MKN8qAKEUJBT4CGZ9eX1hM0tLumfQRzY-bSHlB_zbPfCsEi-XVg/https%3A%2F%2Fjsonplaceholder.typicode.com%2Fcomments%3FpostId%3D1%60)**.**

**4. Create a third async function named `fetchAllData` that uses `await` to call both `fetchPosts` and `fetchComments`.**

**5. Log the combined data from both functions to the console.**

**6. Call `fetchAllData` and observe the output.**

**Assignment 5: Parallel Execution**

**Execute multiple asynchronous operations in parallel using `Promise.all`.**

**Instructions:**

**1. Modify the `fetchAllData` function from Assignment 4.**

**2. Instead of awaiting each function one after another, use `Promise.all` to execute `fetchPosts` and `fetchComments` in parallel.**

**3. Log the combined data to the console once both promises are resolved.**

**4. Call `fetchAllData` and observe the output.**

**Assignment: To-Do application**

**Develop a To-Do application that allows users to manage their tasks efficiently. Application should provide functionality for users to add new to-dos, remove already added to-do. Added to-dos should be displayed on the html page.**

**Required HTML and CSS are given below, you must write javascript code in a separate javascript file and link them together.**

**html**

<!DOCTYPE html>

<html lang="en">

<head>

    <meta charset="UTF-8">

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

    <title>To-Do List</title>

    <style>

        .container {

            width: 300px;

            margin: 0 auto;

            text-align: center;

        }

        ul {

            list-style-type: none;

            padding: 0;

        }

        li {

            display: flex;

            justify-content: space-between;

            padding: 8px;

            background-color: #f3f3f3;

            margin-bottom: 5px;

        }

    </style>

</head>

<body>

    <div class="container">

        <h1>To-Do List</h1>

        <input type="text" id="todokInput" placeholder="Add a new todo">

        <button id="addTodoButton">Add Todo</button>

        <ul id="todoList"></ul>

    </div>

</body>

</html>