SuperSkill n...
REST API (With Postman C...
students
Asynchronous Programming ...
Understanding Promises
Consuming Promises
Queuing Promises
Creating Promises
Iterating with Async / Await
Error Handling in JavaScript
Conclusion
Checkpoint Asynchronous Programming in JavaScript

Overview
Course
Saved
Resources
Report Content
One to One
Checkpoint
Workshop

Objective

In this checkpoint, We will see the topics of iterating with async/await, awaiting a call, handling errors with async/await, chaining async/await, awaiting concurrent requests, and awaiting parallel calls:

Choose at least 3 tasks to solve.

Instructions

Task 01:

Iterating with Async/Await: Write an async function **iterateWithAsyncAwait** that takes an array of values and logs each value with a delay of 1 second between logs.

Task 02:

Awaiting a Call: Create an async function **awaitCall** that simulates fetching data from an API. Use **await** to wait for the API response and then log the data.

Task 03:

Handling Errors with Async/Await: Modify the **awaitCall** function to handle errors gracefully. If the API call fails, catch the error and log a user-friendly error message.

Chaining Async/Await: Write a function **chainedAsyncFunctions** that calls three separate async functions sequentially. Each function should log a message after a delay of 1 second. Chain these functions using **await**.

Task 04:

Awaiting Concurrent Requests: Create an async function **concurrentRequests** that makes two API calls concurrently using **Promise.all()**. Log the combined results after both requests have resolved.

Task 05:

Awaiting Parallel Calls: Write a function **parallelCalls** that takes an array of URLs and fetches data from each URL concurrently using **Promise.all()**. Log the responses once all requests are complete.

Criteria

All questions are rated from 1 to 5

Checkpoint Asessment

Technical mastery

Work quality

Problem resolution

Deadline (Automatic Note 0 Or 5)

Previous Next