Page No :
Date :

1) How promises work in JavaScript

A promise in JavaScript is an object that basically represents the eventual completion or failure of an asynchronous operation and its resulting value. It allows you to write asynchronous code in a more manageable and readable way compared to traditional callback

States

1. Pending
The initial state meaning the asynchronous operation is still in progress

2. Fulfilled
The operation has completed successfully and the promise has a resolved value

3. Rejected
The operation failed and the promise contains the reason for the failure

promise method

then()

used to handle the fulfilled of the promise. It receive a callback function that is executed once the promise is fulfilled

catch()

used to handle any errors or rejection. It receives function that is executed if the promise is rejected.

finally()

This method executes a callback function once promise is either fulfilled or rejected.

2) How Async/await Makes Asynchronous code More Readable

async/await is a syntactic sugar built on top of promise It makes asynchronous code look and behave more like synchronous code, which can improve readability and reduce the complexity of chaining .then() and .catch() methods

async

Declaring a function as async ensures that it returns as Promise and it allows the use of the await keyword inside it

await

Used to pause the execution of the async function until the promise is resolved or rejected it can only be used inside an async function

(3) Error handling Asynchronous code

Error handling is done using the .catch() method which will be executed if the promise is rejected

Error handling with Async/Await

with Async/Await error handling is simplified through the try/catch block making it easier to manage asynchronous error in a manner similar to synchronous code.

(4) Limitations of Async/Await promises

while async/await and promises offer powerful tools for handling asynchronous operations, there are some limitations to be aware of especially in complex error scenarios:

Limitation of promises

chaining Overhead
promises can become difficult to manage when chaining many .then() or .catch() method especially if there are multiple asynchronous

Uncaught Rejection
If you don't properly handle rejection using .catch() it can lead to uncaught promise rejection causing unpredictable behavior.

Limitation of Async / await

Blocking
await makes asynchronous code look synchronous it can also lead to blocking if used incorrectly

Error handling in parallel Operations.

Handling errors in parallel asynchronous operations with async/await can be tricky if one of the promise fails in promise.all the others are aborted. Using try/catch only around the entire await block can make error handling less granular.