**What is a Promise?**

A promise in JavaScript is an object that represents the eventual completion (or failure) of an asynchronous operation and its resulting value. It acts as a placeholder for a value that is not yet available but will be resolved in the future. Promises provide a more manageable way to handle asynchronous operations compared to traditional callbacks.

**Promise States**

A promise can be in one of three states:

1. **Pending**: The initial state. The promise is neither fulfilled nor rejected, and its final value is not yet available.
2. **Fulfilled**: The asynchronous operation completed successfully, and the promise has a resolved value.
3. **Rejected**: The asynchronous operation failed, and the promise has a reason (error) for the failure.

**Promise Chain**

A promise chain allows you to perform a sequence of asynchronous operations. By using .then(), you can specify what should happen once the promise is fulfilled. Each .then() returns a new promise, allowing you to chain multiple operations together.

Example:

fetch('https://api.example.com/data')

.then(response => response.json()) // Handles the response

.then(data => {

console.log(data); // Process the data

})

.catch(error => {

console.error('Error:', error); // Handle any errors

});

**Promise.all()**

Promise.all() is a method that takes an iterable of promises (usually an array) and returns a single promise that resolves when all of the promises in the iterable have resolved, or rejects if any of the promises reject. This is useful when you need to perform multiple asynchronous operations in parallel and wait for all of them to complete.

Example:

const promise1 = fetch('https://api.example.com/data1');

const promise2 = fetch('https://api.example.com/data2');

Promise.all([promise1, promise2])

.then(responses => {

return Promise.all(responses.map(response => response.json()));

})

.then(data => {

console.log(data); // Both data1 and data2

})

.catch(error => {

console.error('Error:', error);

});

**Use of fetch() & .then()**

fetch() is a built-in JavaScript function that initiates an HTTP request and returns a promise that resolves to the response of the request. You can then use .then() to handle the response.

Example:

fetch('https://api.example.com/data')

.then(response => {

if (!response.ok) {

throw new Error('Network response was not ok');

}

return response.json(); // Parse the JSON data from the response

})

.then(data => {

console.log(data); // Use the data

})

.catch(error => {

console.error('Error:', error); // Handle errors

});