**What is then in JavaScript?**

In JavaScript, .then() is a method used with **Promises**. It allows you to **handle the result** of an asynchronous operation once it is completed.

**Explanation**

1. A **Promise** represents a task that takes some time to complete (like fetching data from a server).
2. The .then() method is used to **execute code** once the task is successful.

**Basic Example**

let promise = new Promise((resolve, reject) => {

setTimeout(() => {

resolve("Task completed!"); // Simulating a delayed success

}, 2000);

});

promise.then((message) => {

console.log(message); // Output after 2 seconds: "Task completed!"

});

**Explanation:**

* A **Promise** is created that resolves after 2 seconds.
* .then() runs when the Promise **resolves successfully**.

**Chaining .then()**

You can chain .then() to handle multiple steps sequentially.

let promise = new Promise((resolve) => {

setTimeout(() => resolve(5), 1000); // Resolves with 5 after 1 sec

});

promise

.then((num) => num \* 2) // 5 \* 2 = 10

.then((num) => num + 3) // 10 + 3 = 13

.then((result) => console.log(result)); // Output after 1 sec: 13

**Explanation:**

* Each .then() receives the previous value and processes it further.

**Handling Errors with .catch()**

If an error occurs, .catch() will handle it.

let promise = new Promise((resolve, reject) => {

setTimeout(() => reject("Something went wrong!"), 2000);

});

promise

.then((message) => console.log(message)) // Skipped due to rejection

.catch((error) => console.error(error)); // Output after 2 sec: "Something went wrong!"

**Final Summary**

* .then() runs when a **Promise resolves successfully**.
* It allows **chaining** for sequential operations.
* .catch() handles **errors** if the Promise is rejected.