📘 JavaScript Day 3.5 – Promises (Technical Explanation)

🔹 Definition:

A **Promise** in JavaScript is an object that represents the **eventual completion (resolve)** or **failure (reject)** of an asynchronous operation.

🔄 States of a Promise:

1. **Pending** – Initial state.
2. **Fulfilled** – Operation completed successfully via `resolve()`.
3. **Rejected** – Operation failed via `reject()`.

✅ Creating a Promise:

js
CopyEdit

```
let promise = new Promise(function(resolve, reject) {
  // async task
  if (success) {
    resolve("Success message");
  } else {
    reject("Error message");
  }
});
```

📦 Using (Consuming) a Promise:

- `.then()` → runs on success (fulfilled)
- `.catch()` → runs on error (rejected)

js
CopyEdit

```
promise
  .then(data => console.log(data))
  .catch(err => console.log(err));
```

📌 Example 1 – Simple Logic

js
CopyEdit

```
let promise = new Promise((resolve, reject) => {
  let foodReady = true;

  if (foodReady) {
    resolve("Khana mil gaya!");
  } else {
    reject("Khana nahi mila.");
  }
});

promise
  .then(result => console.log(result))
  .catch(error => console.log(error));
```

⏳ Example 2 – setTimeout (Asynchronous)

js
CopyEdit

```
function getData() {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve("Data received after 2 seconds");
    }, 2000);
  });
}

getData().then(data => console.log(data));
```

🍕 Real-Life Analogy:

Ordering Pizza:

- You place the order → **Pending**
- Pizza arrives → **Resolved**
- Delivery failed → **Rejected**

🧠 Keywords to Remember:

Keyword	Meaning
`resolve()`	Successful completion
`reject()`	Failed attempt (error)
`.then()`	Handle resolved result
`.catch()`	Handle rejection / error

🧪 TASK:

1. Create a promise that resolves in 3 seconds with `"Task Complete"`
2. Create another that rejects in 2 seconds with `"Something went wrong!"`