Types of Promises in JavaScript

Mastering Async Code

SUMANTH M

Frontend developer

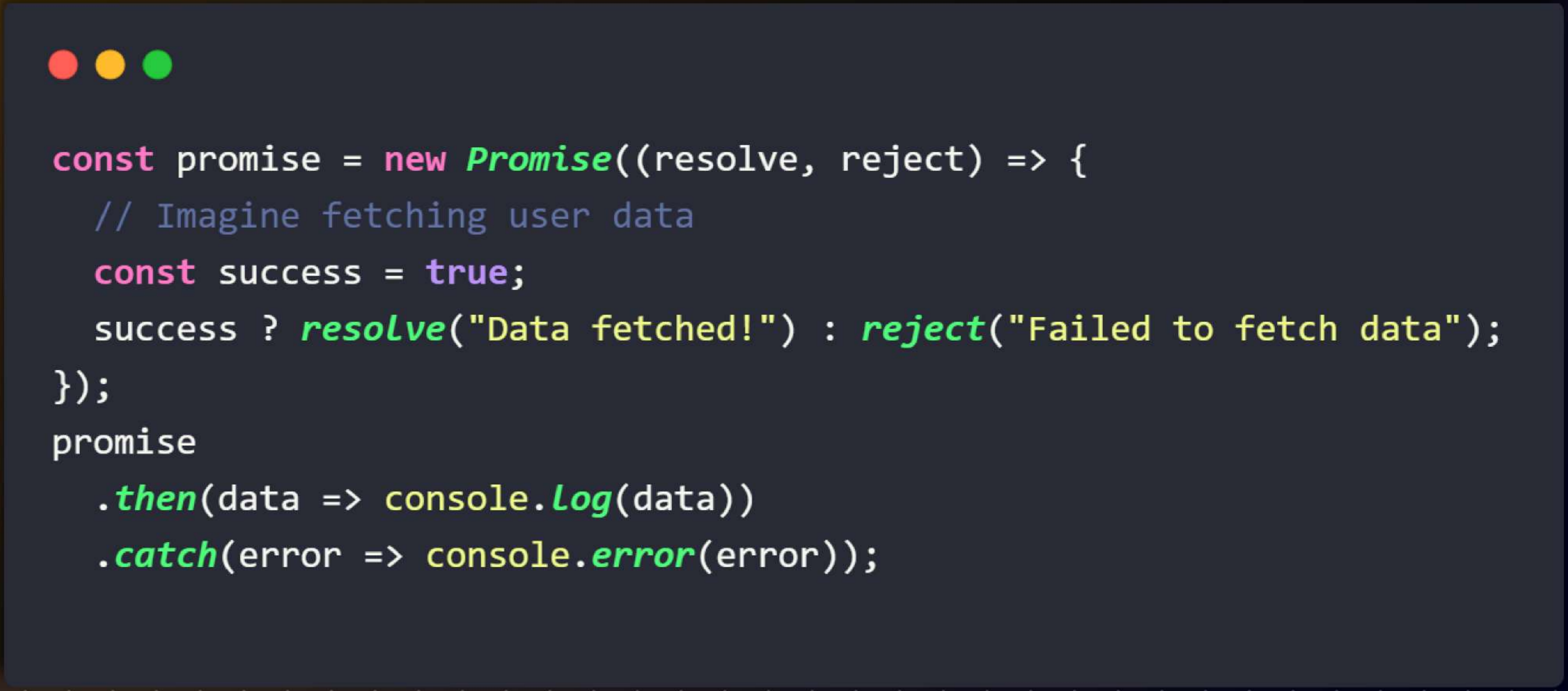
Why Promises Are Important

- Promises help handle asynchronous operations.
- They make code readable and error-proof.
- Whether fetching data, handling multiple tasks, or racing for the fastest result—promises are essential in modern JavaScript development.

SUMANTH M
Frontend developer

2

Simple Promise



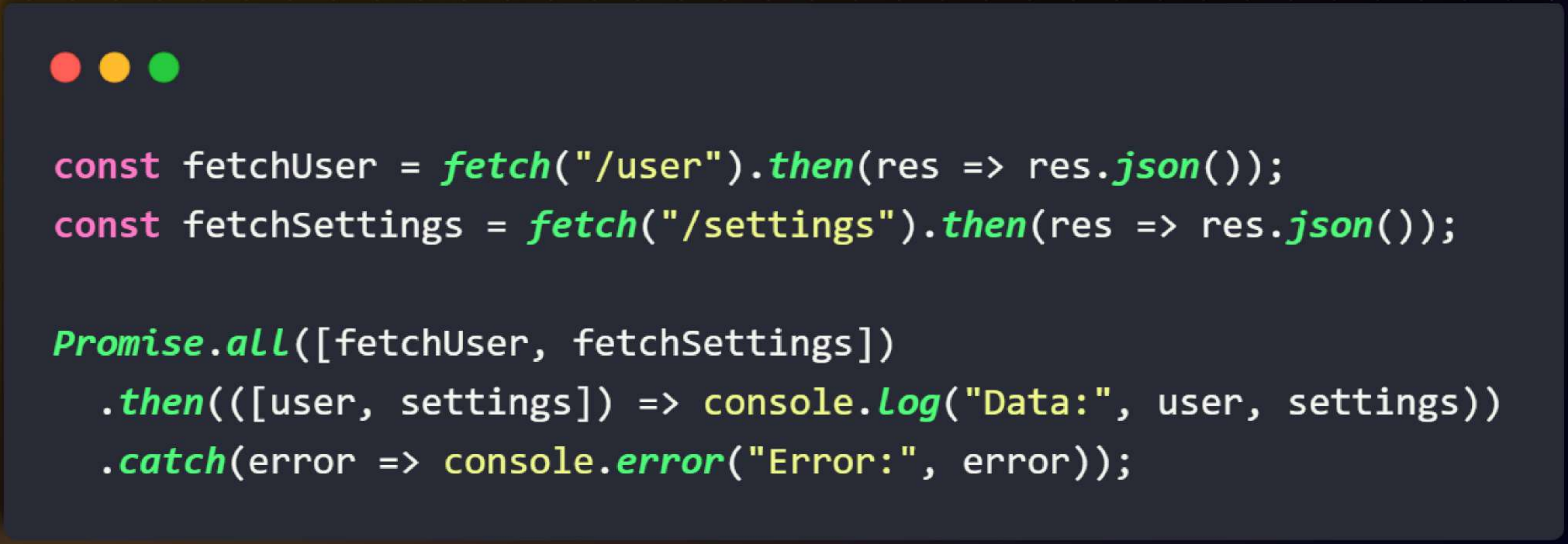
```
const promise = new Promise((resolve, reject) => {
  // Imagine fetching user data
  const success = true;
  success ? resolve("Data fetched!") : reject("Failed to fetch data");
});
promise
  .then(data => console.log(data))
  .catch(error => console.error(error));
```

A promise that either resolves or rejects Often used for API calls or async tasks.

When to Use: For single operations like API requests.

Advantage: Clean handling of success and failure in one block.

Promise.all: Multiple Operations



```
const fetchUser = fetch("/user").then(res => res.json());
const fetchSettings = fetch("/settings").then(res => res.json());

Promise.all([fetchUser, fetchSettings])
  .then(([user, settings]) => console.log("Data:", user, settings))
  .catch(error => console.error("Error:", error));
```

Waits for all promises to resolve.

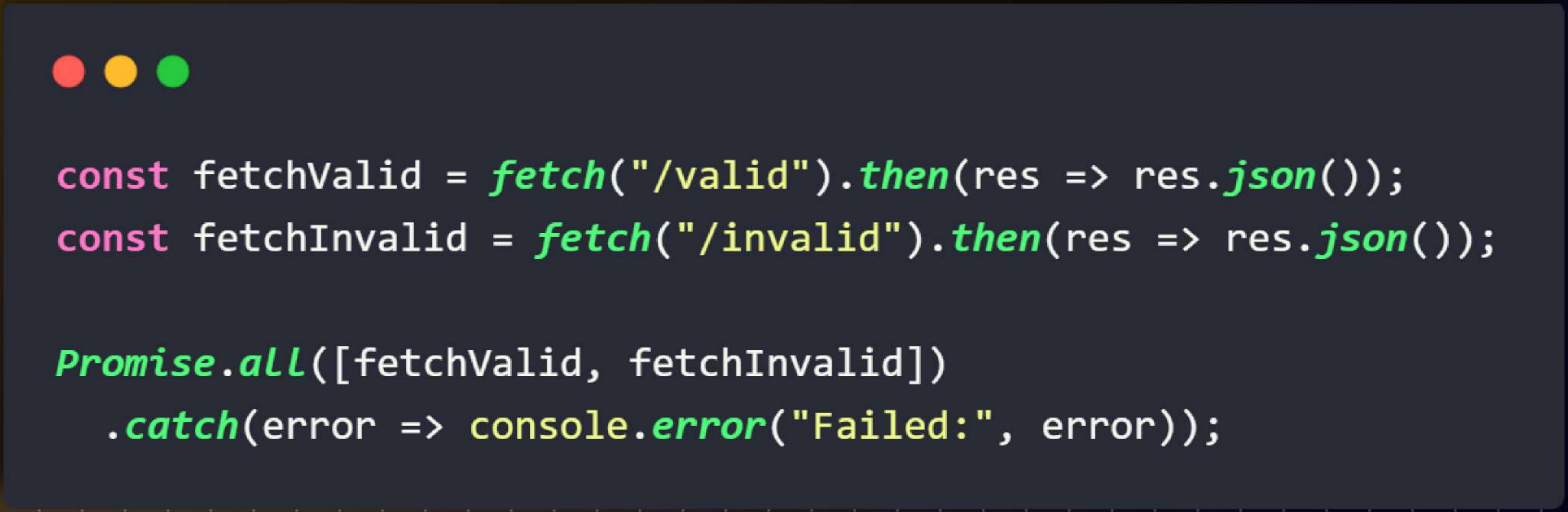
If one fails, everything fails.

Best for multiple async tasks that need to succeed together

When to Use: When you need all results to proceed
Disadvantage: One rejection means everything fails.

Advantage: Clean handling of success and
failure in one block.

4 What Happens if One Promise Fails in Promise.all?



```
const fetchValid = fetch("/valid").then(res => res.json());
const fetchInvalid = fetch("/invalid").then(res => res.json());

Promise.all([fetchValid, fetchInvalid])
  .catch(error => console.error("Failed:", error));
```

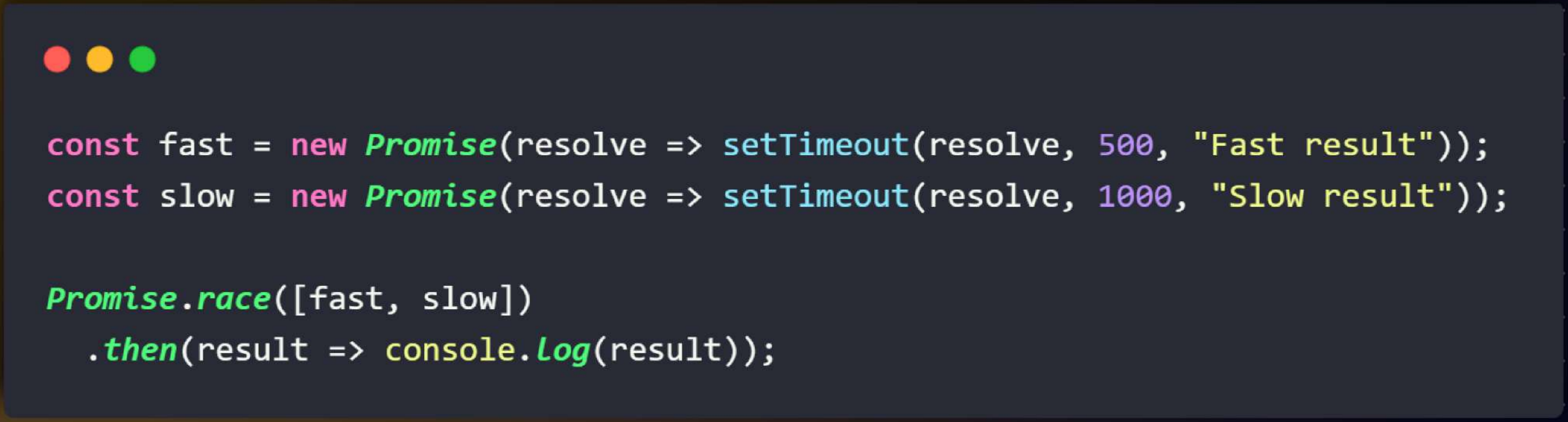
Problem: If even one promise fails, the rest are ignored.

Alternative: If you want partial results, use Promise.allSettled.

SUMANTH M
Frontend developer

5

Promise.race: First to Finish Wins



```
const fast = new Promise(resolve => setTimeout(resolve, 500, "Fast result"));
const slow = new Promise(resolve => setTimeout(resolve, 1000, "Slow result"));

Promise.race([fast, slow])
  .then(result => console.log(result));
```

Returns the result of the first promise to settle, whether it’s resolved or rejected.
Useful when you need speed, such as loading the first available response

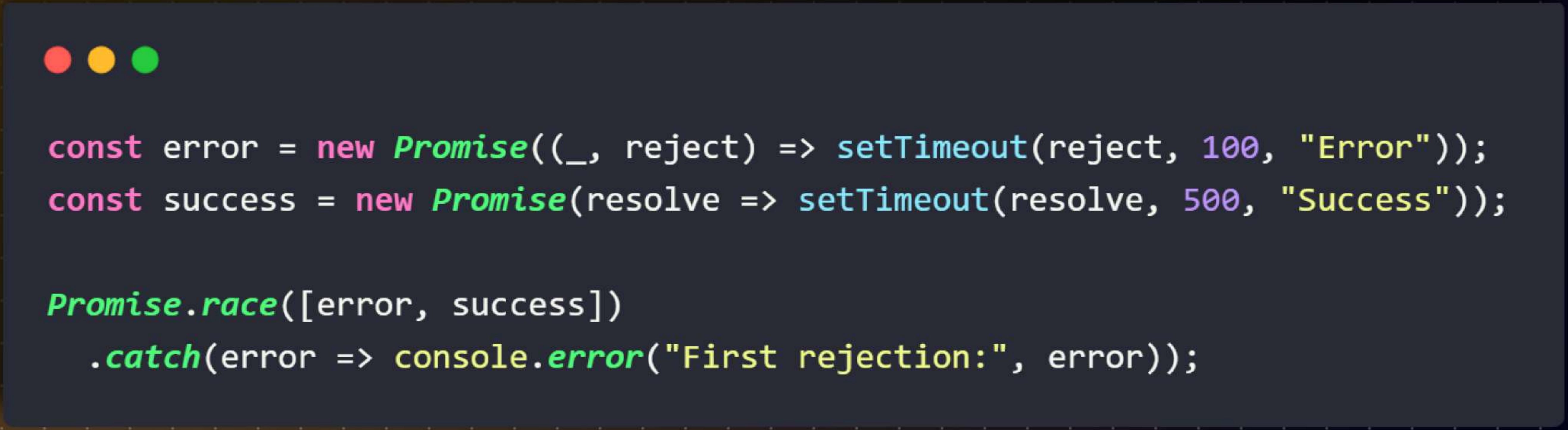
When to Use: When speed matters more than waiting for all tasks.

Limit: You may get an error if the fastest one fails first.

SUMANTH M
Frontend developer

6

What If A Promise In Promise.race Fails?



```
const error = new Promise((_, reject) => setTimeout(reject, 100, "Error"));
const success = new Promise(resolve => setTimeout(resolve, 500, "Success"));

Promise.race([error, success])
  .catch(error => console.error("First rejection:", error));
```

If the first promise rejects, Promise.race will fail immediately.

Disadvantage: Fast rejection stops the race.all tasks.

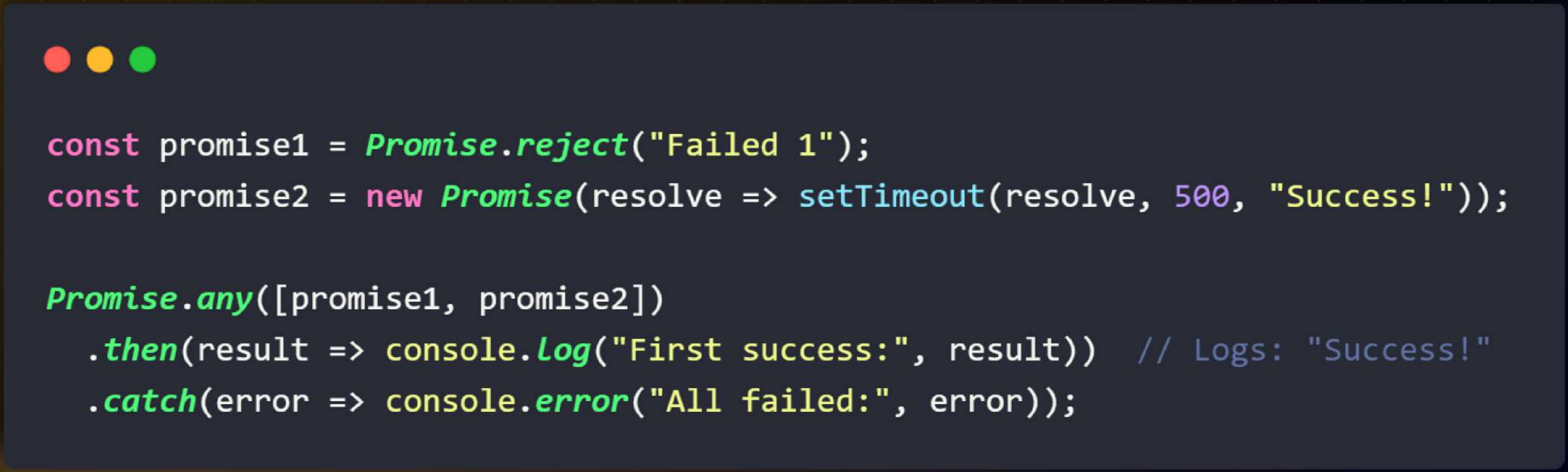
SUMANTH M

Frontend developer

React Js

7

Promise.any: First Success Wins



```
const promise1 = Promise.reject("Failed 1");
const promise2 = new Promise(resolve => setTimeout(resolve, 500, "Success!"));

Promise.any([promise1, promise2])
  .then(result => console.log("First success:", result))  // Logs: "Success!"
  .catch(error => console.error("All failed:", error));
```

Resolves when any one promise resolves. Ignores rejections. Useful when you’re okay with one success, even if others fail

When to Use: When you just need one success, regardless of failures.
Limit: Rejects only if all promises fail.

SUMANTH M
Frontend developer

8

Promise.allSettled: Get All Results

```
const promise1 = fetch("/api1").then(res => res.json());
const promise2 = fetch("/api2").then(res => res.json());

Promise.allSettled([promise1, promise2])
  .then(results => results.forEach(result => {
    console.log(result.status === "fulfilled" ? "Success:" : "Failure:", result);
  }));
```

Waits for all promises to settle, regardless of success or failure. Great when you need both results and errors

When to Use: When you want to know all results, even failures.
Advantage: Avoids immediate failure, unlike Promise.all.

SUMANTH M
Frontend developer

Recap:
Choosing the Right Promise Type

- Promise.all: Wait for everything to resolve; one failure breaks it.
- Promise.race: Fastest result wins, regardless of success/failure.
- Promise.any: Returns the first success; ignores errors.
- Promise.allSettled: Collects all outcomes, good or bad.

SUMANTH M
Frontend developer

10

When To Use Each Promise

- Promise.all: Best for tasks that need to all succeed (e.g., loading multiple data).
- Promise.race: Best for speed, like loading the first available response.
- Promise.any: Great when you only need one success, like trying backup systems.
- Promise.allSettled: Ideal for handling both success and failure gracefully.

SUMANTH M
Frontend developer

Final Thoughts

Choosing the right type of promise is key to efficient async programming.
Use the one that best fits your use case: speed, multiple tasks, or handling failures.
Discussion: Which type of promise has saved you time?

Let’s discuss!

SUMANTH M
Frontend developer