JS

Promise.all() vs Promise.allSettled() in JavaScript

codewithsloba.com

Swipe ⟶

JS

Firstly what are they?

`Promise.all()` and `Promise.allSettled()` are both methods used to work with multiple promises in JavaScript, but they have different behaviors and use cases.

Choose `Promise.all()` when you need all promises to fulfill successfully and want their combined results. Use `Promise.allSettled()` when you need to handle all promise outcomes, including both fulfilled and rejected promises.

Swipe ⟶

JS

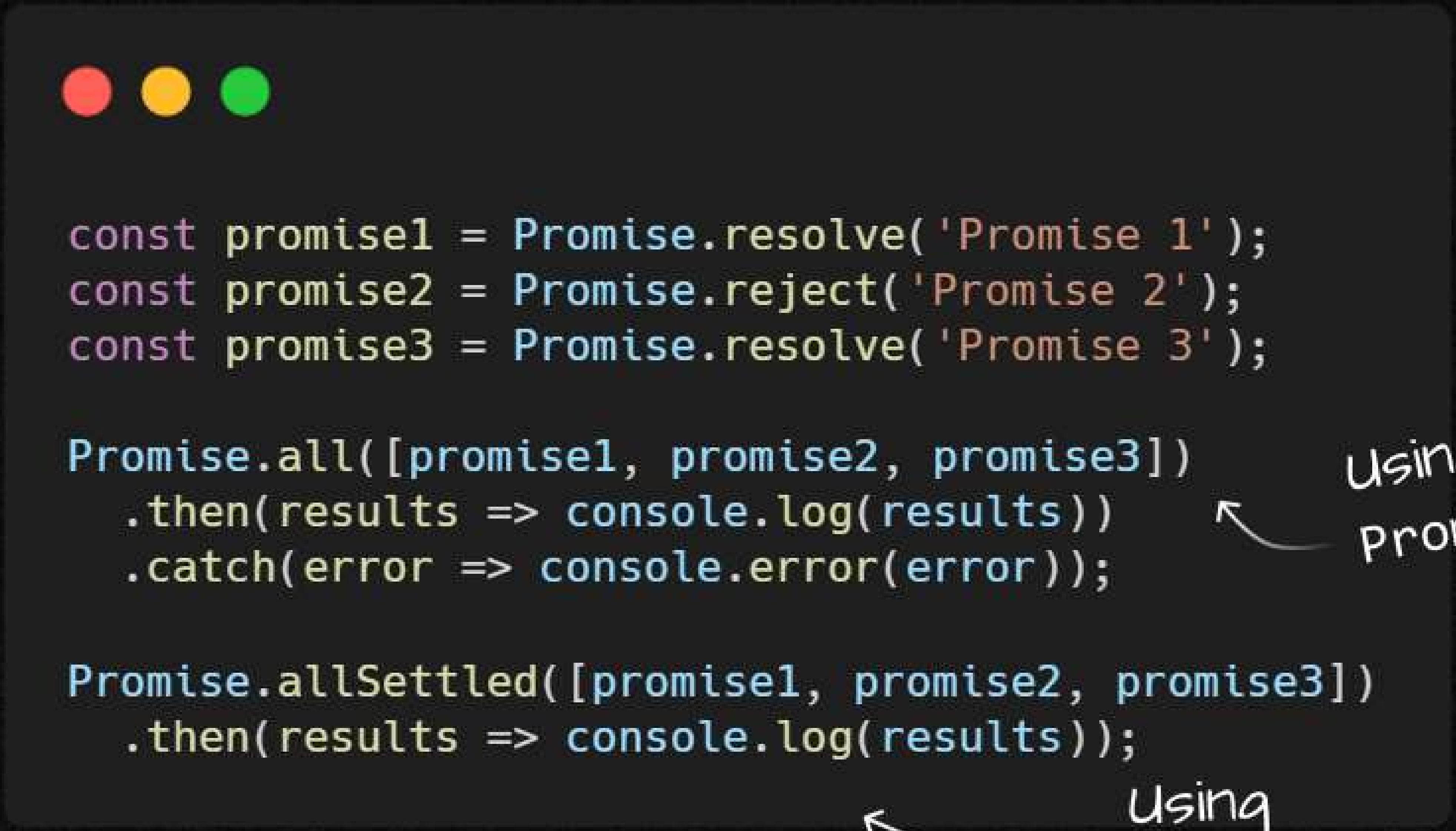
Fulfillment vs Settled Status

`Promise.all()` waits for all the promises to fulfill (successfully complete) or reject (encounter an error) and either returns an array of fulfillment values or rejects with the reason of the first rejected promise.

`Promise.allSettled()` waits for all the promises to either fulfill or reject, and it always returns an array of objects, each representing the outcome of an individual promise, whether it fulfilled or rejected.

Swipe for code

Swipe

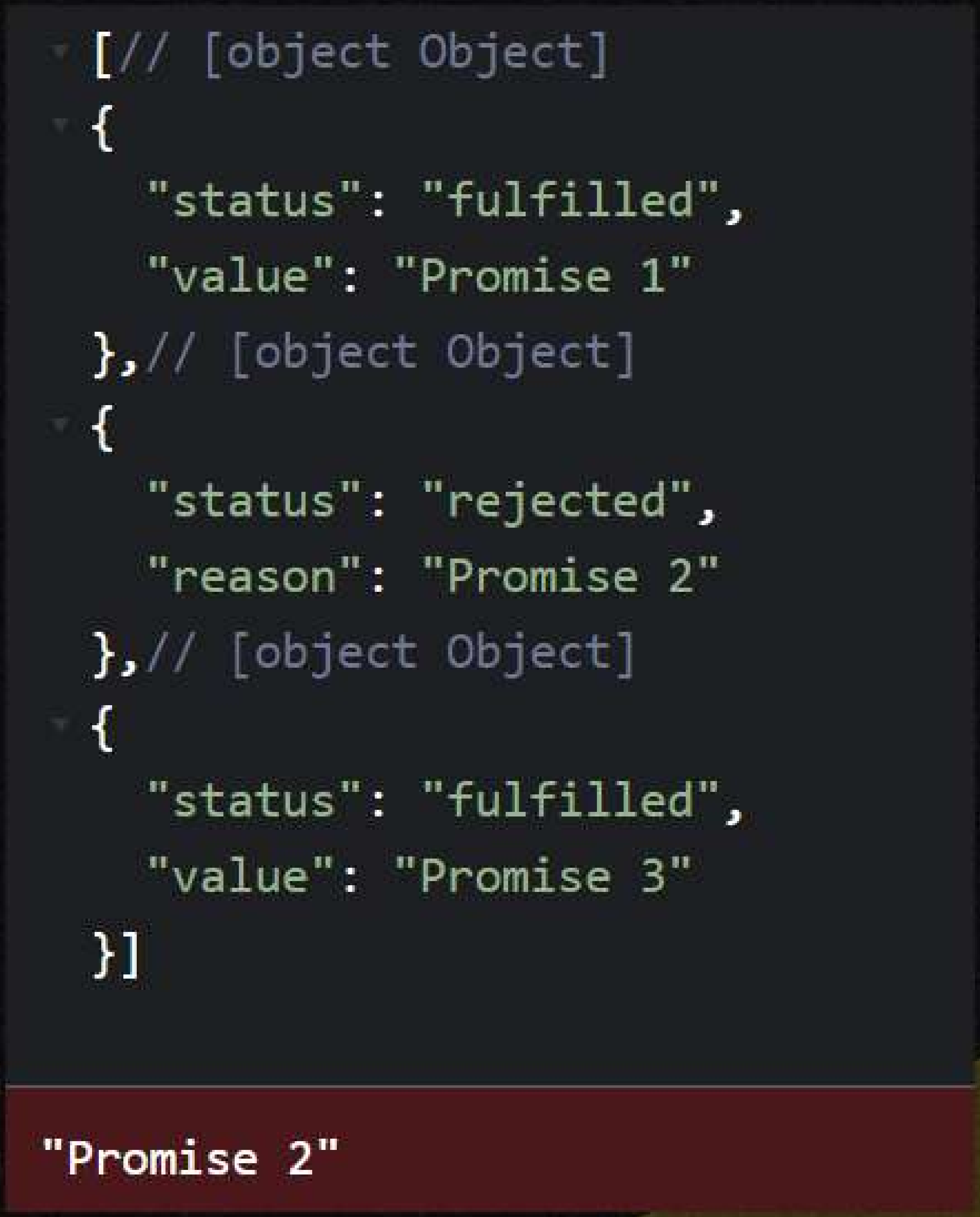


```
const promise1 = Promise.resolve('Promise 1');
const promise2 = Promise.reject('Promise 2');
const promise3 = Promise.resolve('Promise 3');

Promise.all([promise1, promise2, promise3])
  .then(results => console.log(results))
  .catch(error => console.error(error));

Promise.allSettled([promise1, promise2, promise3])
  .then(results => console.log(results));
```

using
Promise.all()

using
Promise.allSettled()

output



```
[// [object Object]
{
  "status": "fulfilled",
  "value": "Promise 1"
},// [object Object]
{
  "status": "rejected",
  "reason": "Promise 2"
},// [object Object]
{
  "status": "fulfilled",
  "value": "Promise 3"
}]
```

```
"Promise 2"
```

Swipe ⟶

JS

Handling Rejections

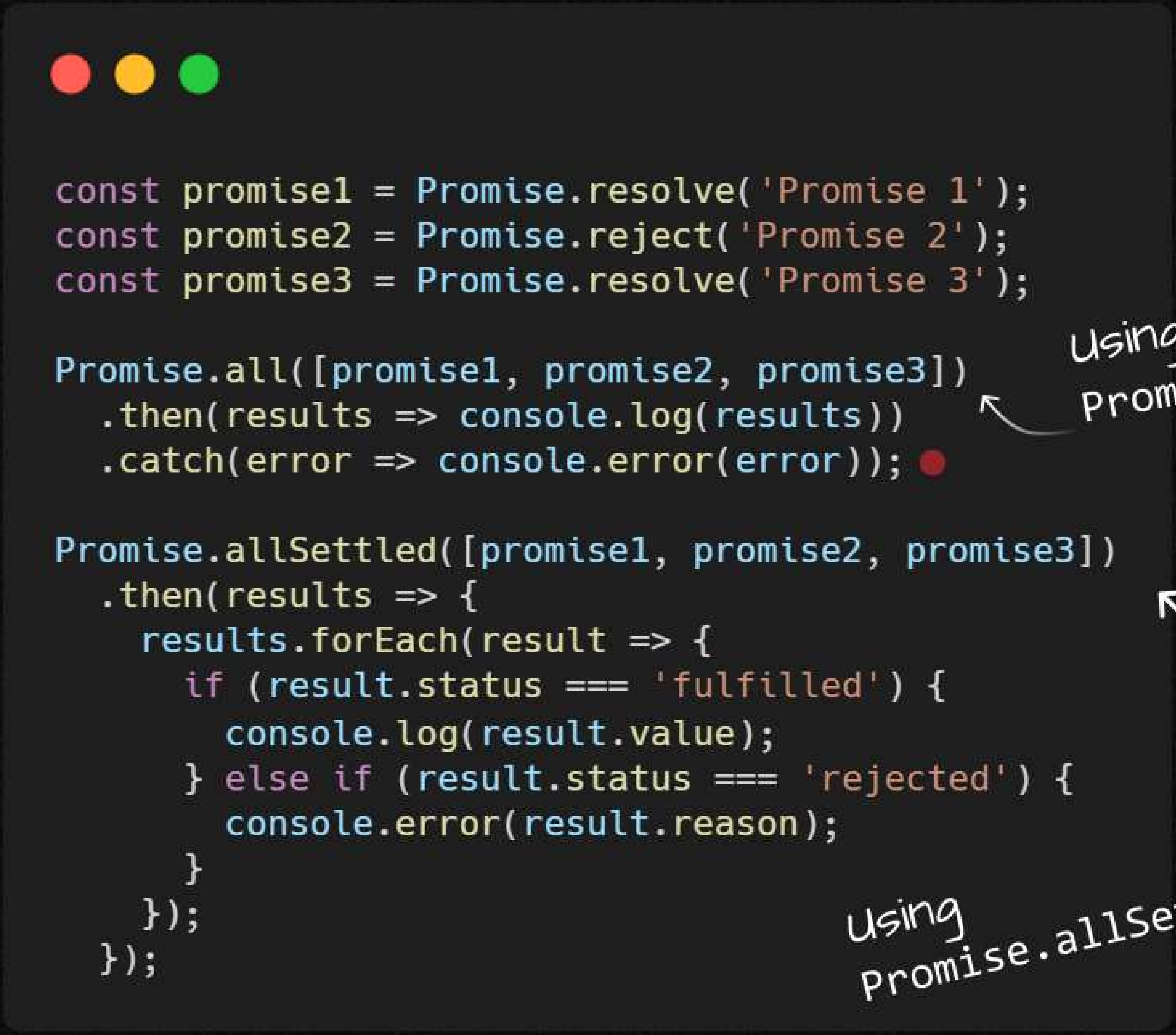
In `Promise.all()`, if any of the promises reject, the whole promise chain immediately rejects with the reason of the first rejected promise, and the remaining promises' results are not accessible.

In `Promise.allSettled()`, even if some promises reject, the resulting array will contain information about all the promises, including both fulfilled and rejected ones.
Use: where you want to process the outcomes of all promises, regardless of whether they succeeded or failed.

Swipe for code

Swipe ⟶

JS



```
const promise1 = Promise.resolve('Promise 1');
const promise2 = Promise.reject('Promise 2');
const promise3 = Promise.resolve('Promise 3');

Promise.all([promise1, promise2, promise3])
  .then(results => console.log(results))
  .catch(error => console.error(error));

Promise.allSettled([promise1, promise2, promise3])
  .then(results => {
    results.forEach(result => {
      if (result.status === 'fulfilled') {
        console.log(result.value);
      } else if (result.status === 'rejected') {
        console.error(result.reason);
      }
    });
  });
```

Using Promise.all()

Using Promise.allSettled()

output

```
"Promise 1"
"Promise 2"
"Promise 3"
"Promise 2"
```

JS

Handling Mixed Results

`Promise.all()` works well when you're interested in the combined results of multiple promises and can tolerate the failure of the entire operation if any promise is rejected.

`Promise.allSettled()` is useful when you want to ensure that all promises are given a chance to complete and you need to process the results of all promises, regardless of whether they succeeded or failed.

Swipe for code

codewithsloba.com

Get a weekly digest of my tips and tutorials by subscribing now.