JavaScript Promise.all()

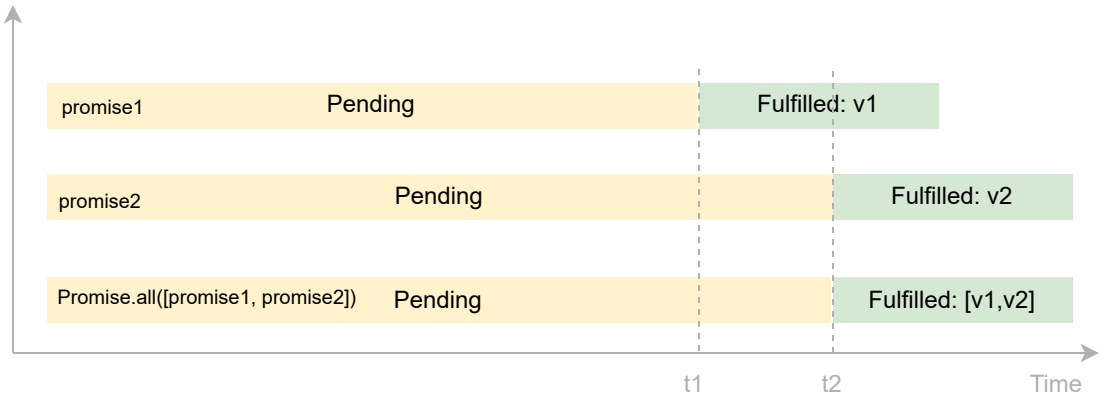
Summary: in this tutorial, you will learn how to use the `Promise.all()` static method to aggregate results from multiple asynchronous operations.

Introduction to the JavaScript Promise.all() method

The `Promise.all()` static method takes an iterable of promises:

```
Promise.all(iterable);
```

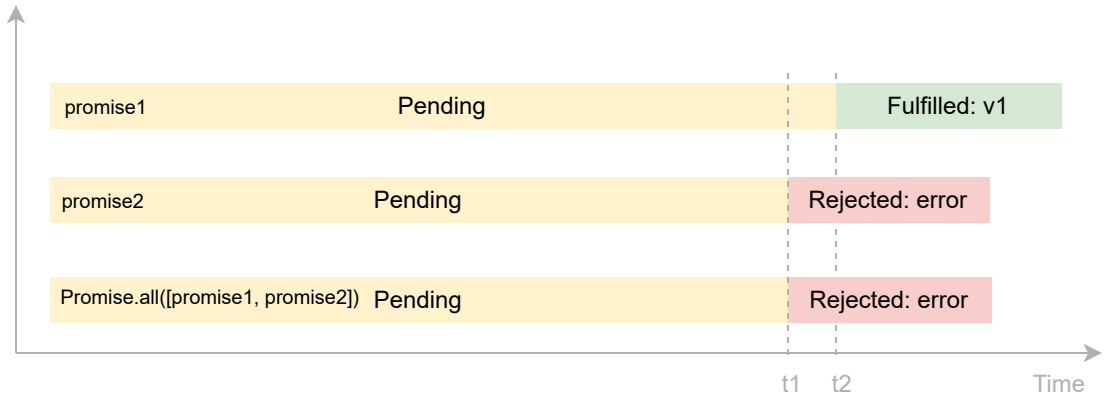
The `Promise.all()` method returns a single promise that resolves when all the input promises have been resolved. The returned promise resolves to an array of the results of the input promises:



In this diagram, the `promise1` resolves to a value `v1` at `t1` and the `promise2` resolves to a value `v2` at `t2`. Hence, the `Promise.all(promise1, promise2)` returns a promise that resolves to an array containing the results of the `promise1` and `promise2` `[v1, v2]` at `t2`.

In other words, the `Promise.all()` waits for all the input promises to resolve and returns a new promise that resolves to an array containing the results of the input promises.

If one of the input promises is rejected, the `Promise.all()` method immediately returns a promise that is rejected with an error of the first rejected promise:



In this diagram, the `promise2` rejects at `t1` with an `error`. Therefore, the `Promise.all()` returns a new promise that is immediately rejected with the same error. Also, the `Promise.all()` doesn't care about other input promises, whether they will be resolved or rejected.

In practice, the `Promise.all()` is useful to aggregate the results from multiple asynchronous operations.

JavaScript Promise.all() method examples

Let's take some examples to understand how the `Promise.all()` method works.

1) Resolved promises example

The following promises resolve to 10, 20, and 30 after 1, 2, and 3 seconds. We use the `setTimeout()` to simulate the asynchronous operations:

```
const p1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('The first promise has resolved');
    resolve(10);
  }, 1 * 1000);
});
const p2 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('The second promise has resolved');
    resolve(20);
  }, 2 * 1000);
});
const p3 = new Promise((resolve, reject) => {
```

```
  setTimeout(() => {
    console.log('The third promise has resolved');
    resolve(30);
  }, 3 * 1000);
});

Promise.all([p1, p2, p3]).then((results) => {
  const total = results.reduce((p, c) => p + c);

  console.log(`Results: ${results}`);
  console.log(`Total: ${total}`);
});
```

Output

```
The first promise has resolved
The second promise has resolved
The third promise has resolved
Results: 10,20,30
Total: 60
```

When all promises have been resolved, the values from these promises are passed into the callback of the `then()` method as an array.

Inside the callback, we use the Array's `reduce()` method to calculate the total value and use the `console.log` to display the array of values as well as the total.

2) Rejected promises example

The `Promise.all()` returns a Promise that is rejected if any of the input promises are rejected.

```
const p1 = new Promise((resolve, reject) => {
    setTimeout(() => {
        console.log('The first promise has resolved');
        resolve(10);
    }, 1 * 1000);

});
```

```
const p2 = new Promise((resolve, reject) => {
    setTimeout(() => {
        console.log('The second promise has rejected');
        reject('Failed');
    }, 2 * 1000);
});
const p3 = new Promise((resolve, reject) => {
    setTimeout(() => {
        console.log('The third promise has resolved');
        resolve(30);
    }, 3 * 1000);
});


Promise.all([p1, p2, p3])
    .then(console.log) // never execute
    .catch(console.log);
```

Output:

```
The first promise has resolved
The second promise has rejected
Failed
The third promise has resolved
```

In this example, we have three promises: the first one is resolved after 1 second, the second is rejected after 2 seconds, and the third one is resolved after 3 seconds.

As a result, the returned promise is rejected because the second promise is rejected. The `catch()` method is executed to display the reason for the rejected promise.

Summary

- The `Promise.all()` method accepts a list of promises and returns a new promise that resolves to an array of results of the input promises if all the input promises are resolved, or rejected with an error of the first rejected promise.

- Use the `Promise.all()` method to aggregate results from multiple asynchronous operations.

Quiz