JavaScript Promise.any()

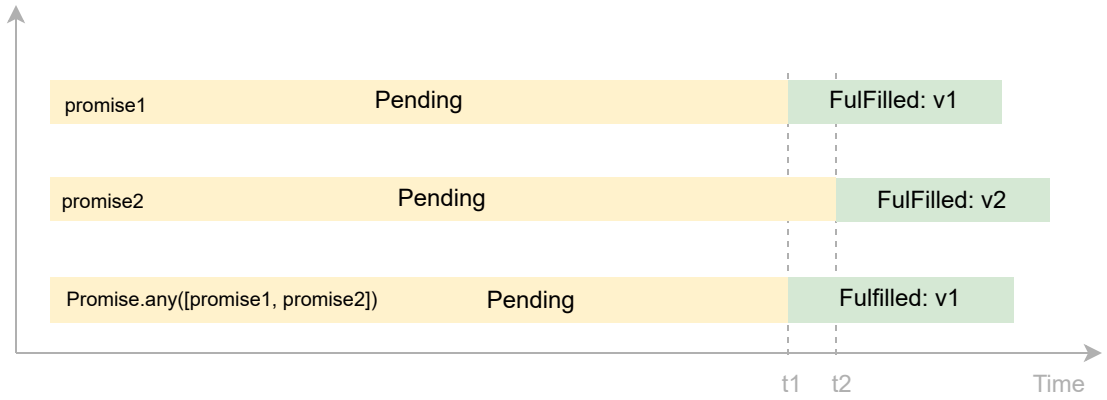
Summary: in this tutorial, you'll learn how to use the JavaScript `Promise.any()` method to compose promises.

Introduction to JavaScript Promise.any() method

The `Promise.any()` method accepts a list of Promise objects as an iterable object:

```
Promise.any(iterable);
```

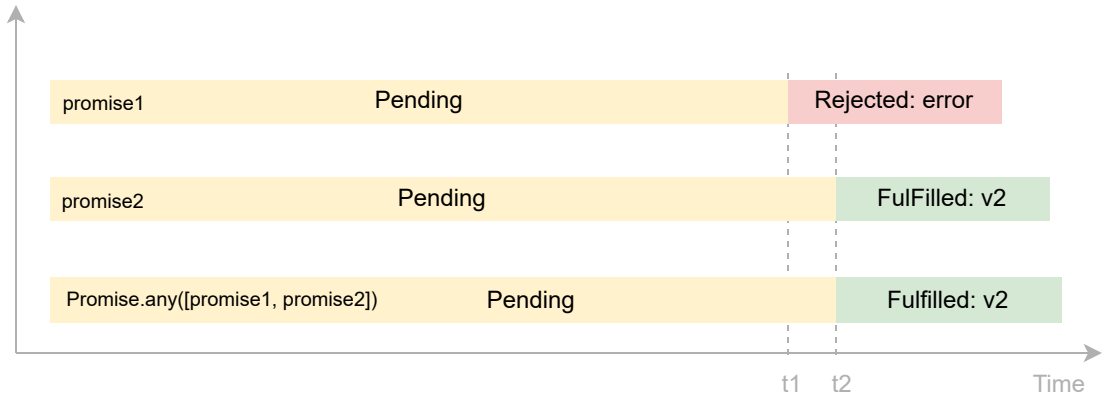
If one of the promises in the iterable object is fulfilled, the `Promise.any()` returns a single promise that resolves to a value which is the result of the fulfilled promise:



In this diagram:

- The `promise1` resolves to a value `v1` at `t1`.
- The `promise2` resolves to a value `v2` at `t2`.
- The `Promise.any()` returns a promise that resolves to a value `v1`, which is the result of the `promise1`, at `t1`

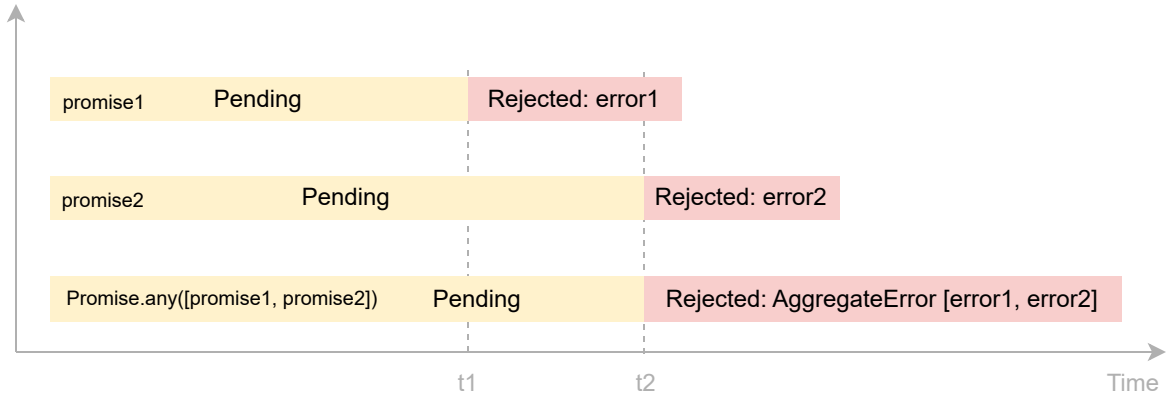
The `Promise.any()` returns a promise that is fulfilled with any first fulfilled promise even if some promises in the iterable object are rejected:



In this diagram:

- The `promise1` is rejected with an `error` at `t1`.
- The `promise2` is fulfilled to value `v2` at `t2`.
- The `Promise.any()` returns the promise that resolves to a value `v2` which is the result of the `promise2`. Note that the `Promise.any()` method ignores the rejected promise (`promise1`).

If all promises in the iterable object are rejected or if the iterable object is empty, the `Promise.any()` return a promise that is rejected with an `AggregateError` containing all the rejection reasons. The `AggregateError` is a subclass of `Error`.



In this diagram:

- The `promise1` is rejected for an `error1` at `t1`.
- The `promise2` is rejected for an `error2` at `t2`.
- The `Promise.any()` returns a promise that is rejected at `t2` with an `AggregateError` containing the `error1` and `error2` of all the rejected promises.

JavaScript Promise.any() examples

Let's take some examples of using the `Promise.any()` method.

1) All promises fulfilled example

The following example demonstrates the `Promise.any()` method with all promises fulfilled:

```
const p1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('Promise 1 fulfilled');
    resolve(1);
  }, 1000);
});

const p2 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('Promise 2 fulfilled');
    resolve(2);
  }, 2000);
});

const p = Promise.any([p1, p2]);
p.then((value) => {
  console.log('Returned Promise');
  console.log(value);
});
```

Output:

```
Promise 1 fulfilled
Returned Promise
1
Promise 2 fulfilled
```

How it works.

- First, create a new promise `p1` that will resolve to a value `1` after one second.
- Second, create a new promise `p2` that will resolve to a value `2` after two seconds.
- Third, use the `Promise.any()` method that uses two promises `p1` and `p2`. The `Promise.any()` returns a promise `p` that will resolve to the value `1` of the first fulfilled promise (`p1`) after one second.

2) One promise rejected example

The following example uses the `Promise.any()` method with a list of promises that have a rejected promise:

```
const p1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('Promise 1 rejected');
    reject('error');
  }, 1000);
});

const p2 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('Promise 2 fulfilled');
    resolve(2);
  }, 2000);
});

const p = Promise.any([p1, p2]);
p.then((value) => {
  console.log('Returned Promise');
  console.log(value);
});
```

Output:

```
Promise 1 rejected
Promise 2 fulfilled
```

```
Returned Promise
2
```

In this example, the `Promise.any()` ignores the rejected promise. When the `p2` resolves with the value `2`, the `Promise.any()` returns a promise that resolves to the same value as the result of the `p2`.

3) All promises rejected example

The following example demonstrates how to use the `Promise.any()` method with all promises rejected:

```
const p1 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('Promise 1 rejected');
    reject('error1');
  }, 1000);
});

const p2 = new Promise((resolve, reject) => {
  setTimeout(() => {
    console.log('Promise 2 rejected');
    reject('error2');
  }, 2000);
});

const p = Promise.any([p1, p2]);
p.catch((e) => {
  console.log('Returned Promise');
  console.log(e, e.errors);
});
```

Output:

```
Promise 1 rejected
Promise 2 rejected
```

```
Returned Promise
[AggregateError: All promises were rejected] [ 'error1', 'error2' ]
```

In this example, both `p1` and `p2` were rejected with the string `error1` and `error2`. Therefore, the `Promise.any()` method was rejected with an `AggregateError` object that has the `errors` property containing all the errors of the rejected promises.

When to use the JavaScript Promise.any() method

In practice, you use the `Promise.any()` to return the first fulfilled promise. Once a promise is fulfilled, the `Promise.any()` method does not wait for other promises to be complete. In other words, the `Promise.any()` short circuits after a promise is fulfilled.

For example, you have a resource served by two or more content delivery networks (CDN). To dynamically load the first available resource, you can use the `Promise.any()` method.

The following example uses the `Promise.any()` method to fetch two images and display the first available image.

The index.html file

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0" />
        <title>JavaScript Promise.any() Demo</title>
    </head>
    <body>
        <script src="js/app.js"></script>
    </body>
</html>
```

The app.js file

```
function getImageBlob(url) {
  return fetch(url).then((response) => {
    if (!response.ok) {
```

```
      throw new Error(`HTTP status: ${response.status}`);
    }
    return response.blob();
  });
}

let cat = getImageBlob(
  'https://upload.wikimedia.org/wikipedia/commons/4/43/Siberian_black_tabby_blotched_cat_03.j
);
let dog = getImageBlob(
  'https://upload.wikimedia.org/wikipedia/commons/a/af/Golden_retriever_eating_pigs_foot.jpg
);

Promise.any([cat, dog])
  .then((data) => {
    let objectURL = URL.createObjectURL(data);
    let image = document.createElement('img');
    image.src = objectURL;
    document.body.appendChild(image);
  })
  .catch((e) => {
    console.log(e.message);
  });
```

How it works.

- First, define the `getImageBlob()` function that uses the fetch API to get the image's blob from a URL. The `getImageBlob()` returns a `Promise` object that resolves to the image blob.
- Second, define two promises that load the images.
- Third, show the first available image by using the `Promise.any()` method.

Summary

- Use the JavaScript `Promise.any()` method to take a list of promises and return a promise that is fulfilled first.

Quiz