Promise API Methods

In Detail

Mallikarjun | @CodeBustler

Promise Methods

```
Promise.all(iterable)

Promise.allSettled(iterable)

Promise.race(iterable)

Promise.any(iterable)

Promise.resolve(value)

Promise.reject(error)
```

Prototype

```
Promise.catch(onRejected)

Promise.finally(onFinally)

Promise.then(Resolve, Reject)
```

@CodeBustler

1. Promise.all(iterable)

The Promise.all() static method takes an iterable of promises as input and **returns a single Promise**.

This returned promise **fulfills when all** of the input's promises fulfill (including when an empty iterable is passed), with an array of the fulfillment values.

It rejects when any of the input's promises rejects, with this first rejection reason(Message).

@CodeBustler

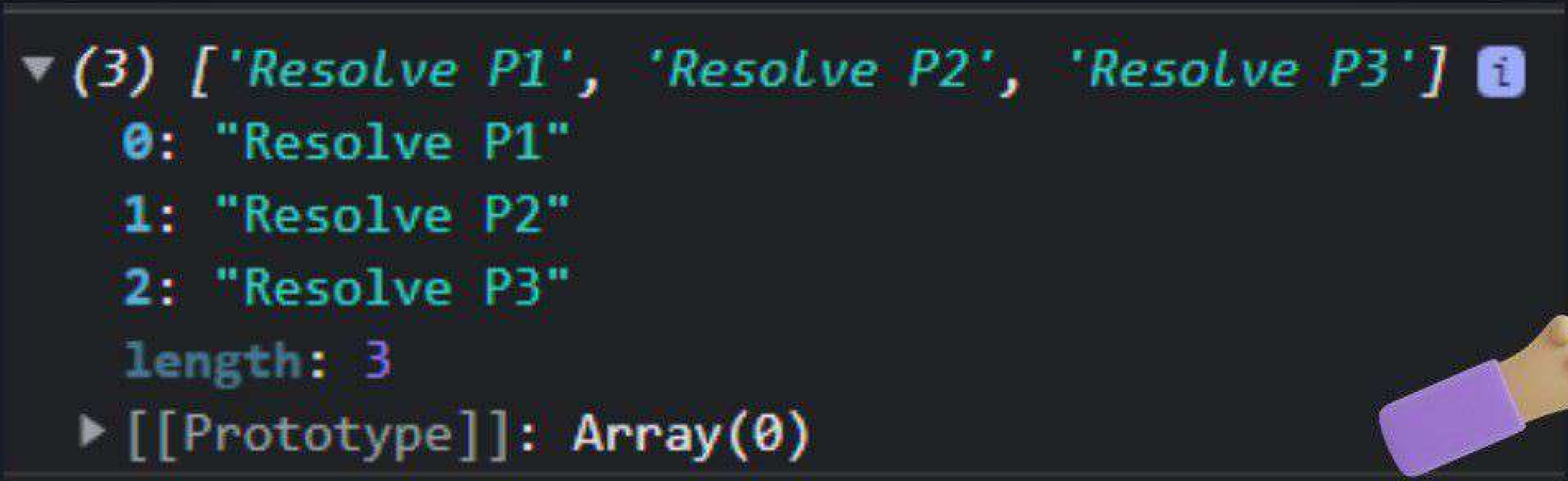
Example Promise.all(iterable)

```javascript
let p1 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P1"), 1000);
});

let p2 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P2"), 2000);
});

let p3 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P3"), 3000);
});

Promise.all([p1, p2, p3])
  .then((values) => console.log(values))
  .catch((error) => console.log(error.message));
```



```
▼ (3) ['Resolve P1', 'Resolve P2', 'Resolve P3'] i
    0: "Resolve P1"
    1: "Resolve P2"
    2: "Resolve P3"
    length: 3
  ▶ [[Prototype]]: Array(0)
```

@CodeBustler

```
// IF REJECT IN ANY PROMISE

let p2 = new Promise((resolve, reject) => {
  setTimeout(() => reject(new Error("P2 Error")), 2000);
}); // p1 & p3 Same

Promise.all([p1, p2, p3])
  .then((values) => console.log(values))
  .catch((error) => console.log(error.message));

// Output : P2 Error
```

2.Promise.allSettled(itrb)

Returns a **new Promise** that resolves after all promises in the iterable have settled (**either resolved or rejected**).

The returned Promise resolves to an array of objects representing the fulfillment **status of each promise**.

@CodeBustler

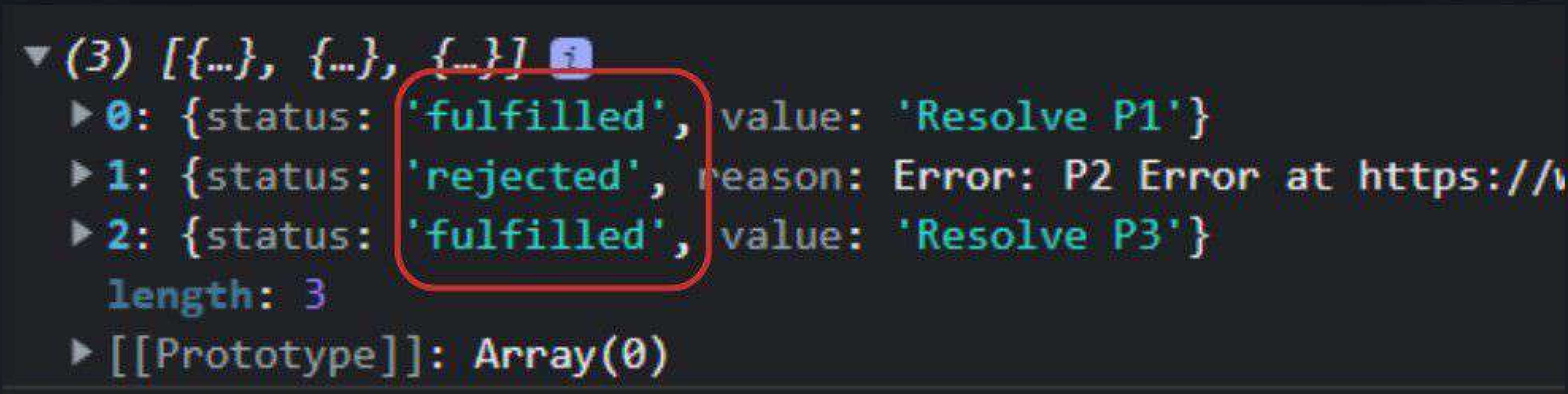
Example Promise.allSettled( )

```javascript
let p1 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P1"), 1000);
});

let p2 = new Promise((resolve, reject) => {
  setTimeout(() => reject(new Error("P2 Error")), 2000);
});

let p3 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P3"), 3000);
});

Promise.allSettled([p1, p2, p3])
  .then((values) => console.log(values))
  .catch((error) => console.log(error.message));
```



```
▼ (3) [{…}, {…}, {…}] i
  ▶ 0: {status: 'fulfilled', value: 'Resolve P1'}
  ▶ 1: {status: 'rejected', reason: Error: P2 Error at https://
  ▶ 2: {status: 'fulfilled', value: 'Resolve P3'}
    length: 3
  ▶ [[Prototype]]: Array(0)
```

@CodeBustler

3.Promise.race(iterable)

Waits for the **first promise** to settle and its **result/error** becomes the outcome

```
let p1 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P1"), 1000);
});

let p2 = new Promise((resolve, reject) => {
  setTimeout(() => reject(new Error("P2 Error")), 2000);
});

let p3 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P3"), 500);
});

Promise.race([p1, p2, p3])
  .then((values) => console.log(values))
  .catch((error) => console.log(error));

// Output : Resolve P3
```

Retutrn First Promise

if First promise reject return error

@CodeBustler

4.Promise.any(iterable)

Returns a new Promise **that resolves** as soon as one of the promises in the iterable resolves. **If all promises reject,** the returned Promise is rejected with an AggregateError containing all the rejection reasons.

5.Promise.resolve(value)

Makes a resolved promise with the given value

6.Promise.reject(error)

Makes a rejected promise with the given error

@CodeBustler

```
// 04.Promise.any()
let p1 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P1"), 2000);
});

let p2 = new Promise((resolve, reject) => {
  setTimeout(() => reject(new Error("P2 Error")), 500);
});

let p3 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("Resolve P3"), 3000);
});

Promise.any([p1, p2, p3])
  .then((values) => console.log(values))
  .catch((error) => console.log(error));
// Output : Resolve P1
```

First Promise

Returns First Fullfiled

```
// 05.Promise.resolve()
let resolve = Promise.resolve("Resolved");
resolve.then((value) => console.log(value));
// Output : Resolved


// 06. Promise.resolve()
let error = Promise.reject(new Error("Fail"));
error.catch((error) => console.log(error.message));
// Output : Fail
```

@CodeBustler