W8D3 - Promises

miguel.garrido@ironhack.com

Promises

A **Promise** is a special class in Javascript

Promises

```
const myPromise = new Promise();
```

Promises

```
const myPromise = new Promise((resolve, reject) => { ... });
```

1. Takes exactly one argument: the executor function
2. The executor receives two functions as parameters:

   resolve(value): Call to fulfill the promise with a value

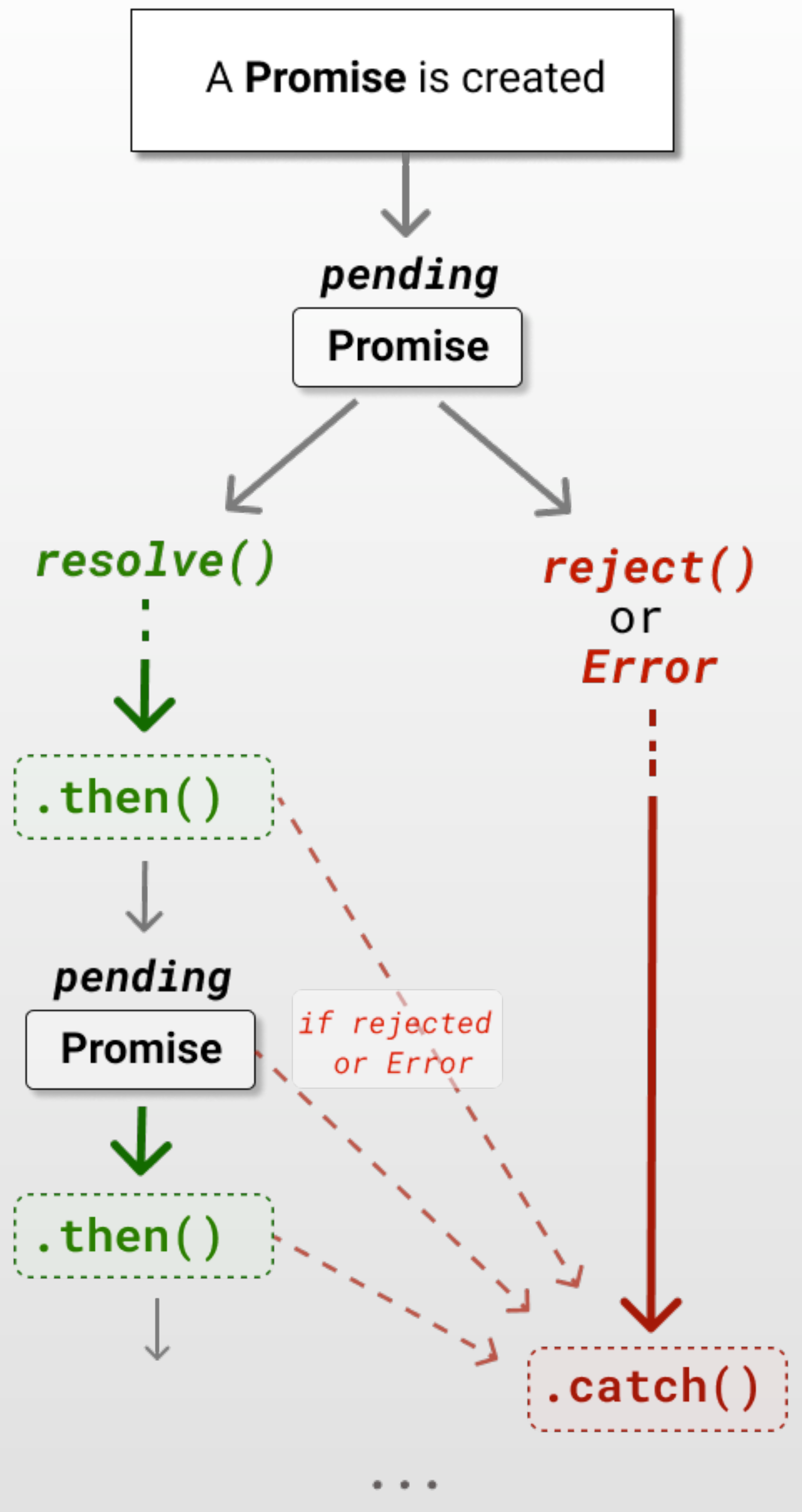
   reject(reason): Call to reject the promise with a reason or error

Promises

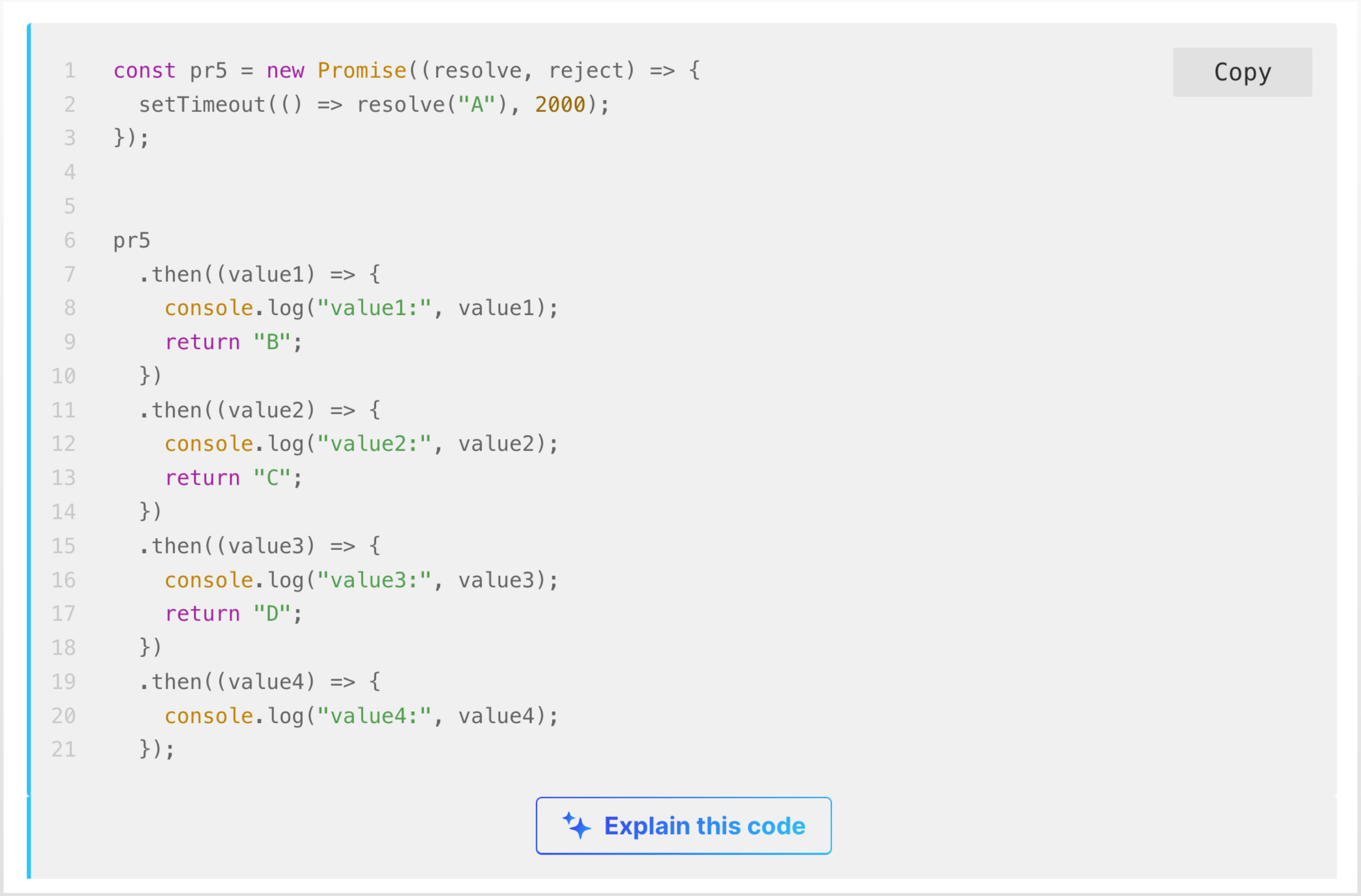
Consuming a Promise:

```
myPromise()
  .then(value => { /* Registers callbacks for when the promise is fulfilled */ })
  .catch(error => { /* Handles promise rejections */ });
```

Promises


A **Promise** is created
pending
Promise
resolve()
reject()
or
Error
.then()
pending
Promise
if rejected or Error
.then()
.catch()
...

Multiple .then() blocks

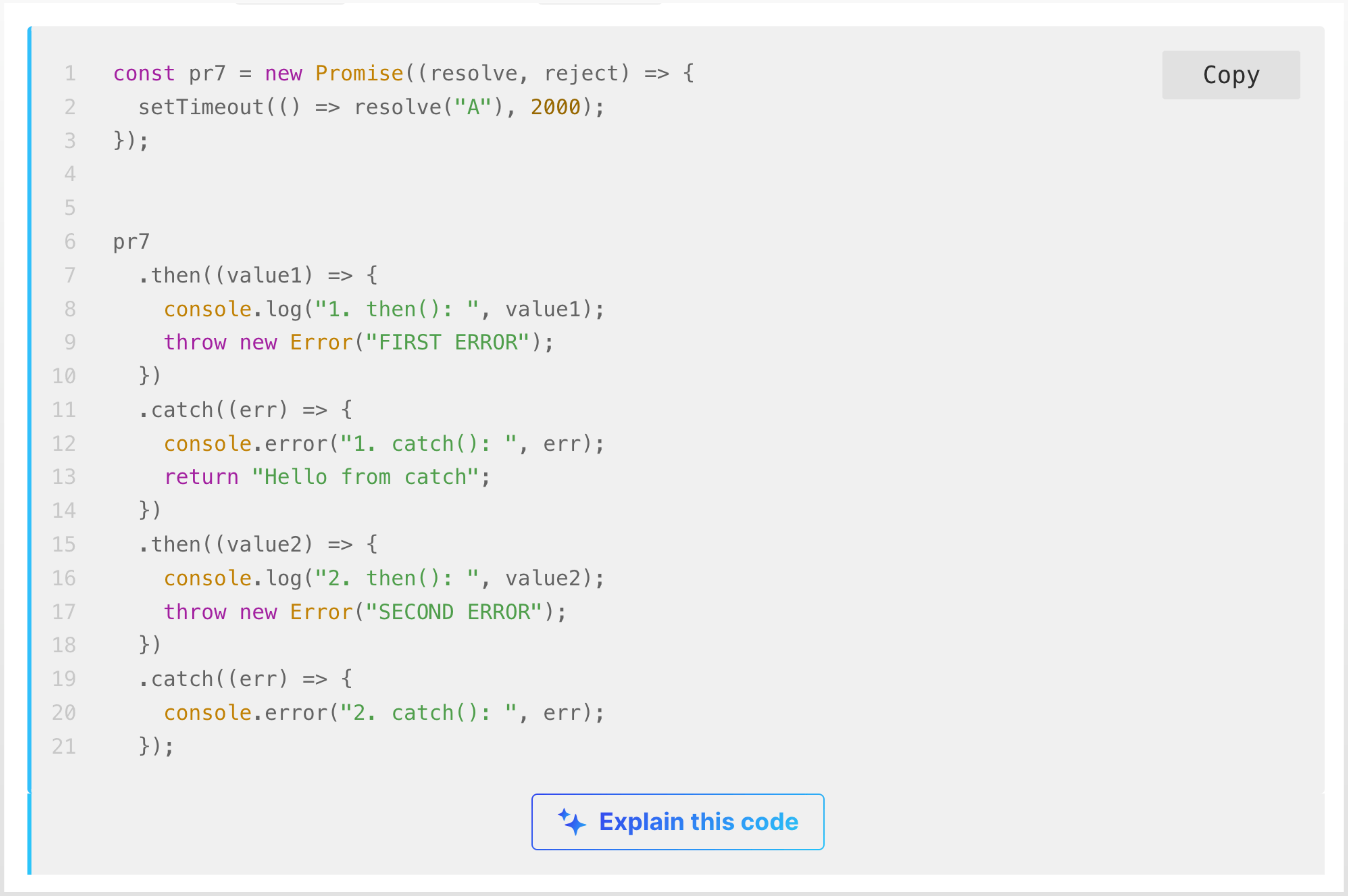


Copy

```
const pr5 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("A"), 2000);
});


pr5
  .then((value1) => {
    console.log("value1:", value1);
    return "B";
  })
  .then((value2) => {
    console.log("value2:", value2);
    return "C";
  })
  .then((value3) => {
    console.log("value3:", value3);
    return "D";
  })
  .then((value4) => {
    console.log("value4:", value4);
  });
```

Multiple .catch() blocks



```javascript
const pr7 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("A"), 2000);
});


pr7
  .then((value1) => {
    console.log("1. then(): ", value1);
    throw new Error("FIRST ERROR");
  })
  .catch((err) => {
    console.error("1. catch(): ", err);
    return "Hello from catch";
  })
  .then((value2) => {
    console.log("2. then(): ", value2);
    throw new Error("SECOND ERROR");
  })
  .catch((err) => {
    console.error("2. catch(): ", err);
  });
```

.finally() method

The **finally()** method is used to do final processing or cleanup once the promise is settled,

```
const pr8 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("A"), 2000);
});


pr8
  .then((value1) => console.log("1. then()"))
  .then((value2) => console.log("2. then()"))
  .finally(() => {
    console.log("finally()");
  });
```

Promise.all()

The **finally()** method is used to do final processing or cleanup once the promise is settled,

```
const p1 = new Promise((resolve, reject) => {
  setTimeout(() => resolve("foo"), 1000);
});

const p2 = new Promise((resolve, reject) => {
  setTimeout(() => resolve(1337), 2000);
});

const p3 = new Promise((resolve, reject) => {
  setTimeout(() => resolve( { name: "Bob" } ), 4000);
});


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