W2D4 - Promises

LIGHTHOUSE LABS

AGENDA

- Async flow with callbacks Recap
- Exception handling
- Error handling with an async flow with callbacks
- Callback Hell
- Promises

Async flow with callbacks Recap

Example: What is the console.log going to print?

```
const upperCaseAsync = (inputStr, callback) => {
  setTimeout(() => {
    callback(inputStr.toUpperCase());
  }, 3000);
};

upperCaseAsync('Sponge Bob', (upperCaseName) =>
console.log(upperCaseName));
```

Exception Handling

- Errors that are thrown will **stop the execution** of our code.
- To better handle errors in our code, we use a **try catch**.

```
try {
  printName('SpongeBob')
} catch(err) {
  console.log("Error:", err.message);
}
```

Try catch allows the code to continue executing.

Handling Errors with Async Code

- Try Catch works well with synchronous code, but it **does not work with asynchronous code**
- How do we handle errors with async code?

Callback Hell

- When we need to have multiple nested async function calls, we get into what we call **callback hell!**

Exercise: https://gist.github.com/DominicTremblay/311014069b5ce616b5ccf4792a362910

Promises

Why Promises?

- Promises suggest a **better syntax** to handle callbacks
- Multiple async calls can be **handled more gracefully**

Promises

What is a promise?

- A promise is an object that represents the **eventual completion (or failure)** of an asynchronous operation and its resulting value

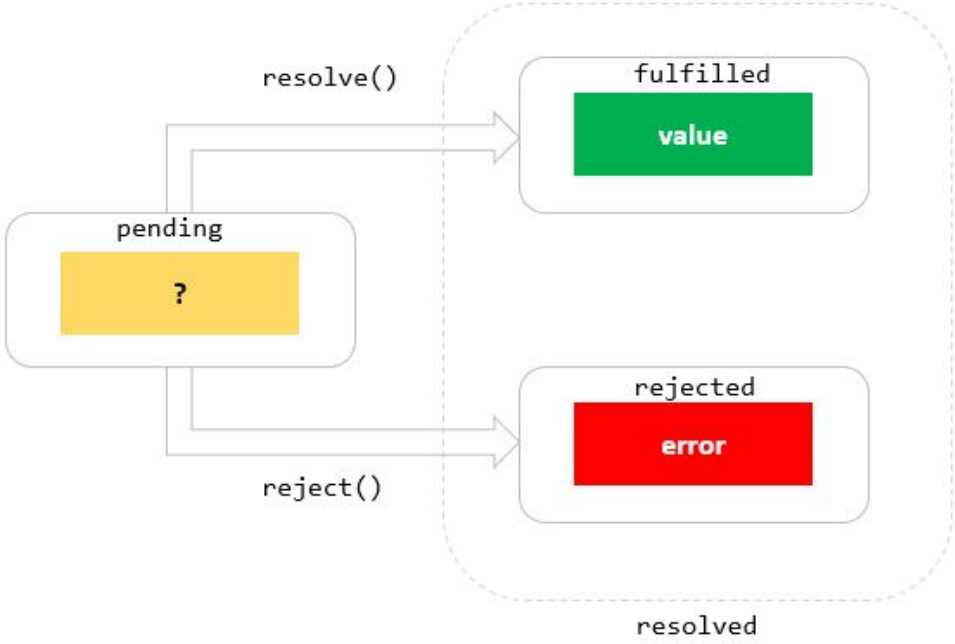
Promises

The Promise Object has 3 potential states:

1

Pending:

The async operation is **ongoing**.



2

Fulfilled:

The async operation is **completed successfully**.

3

Rejected:

The async operation has **failed** and we get an error.

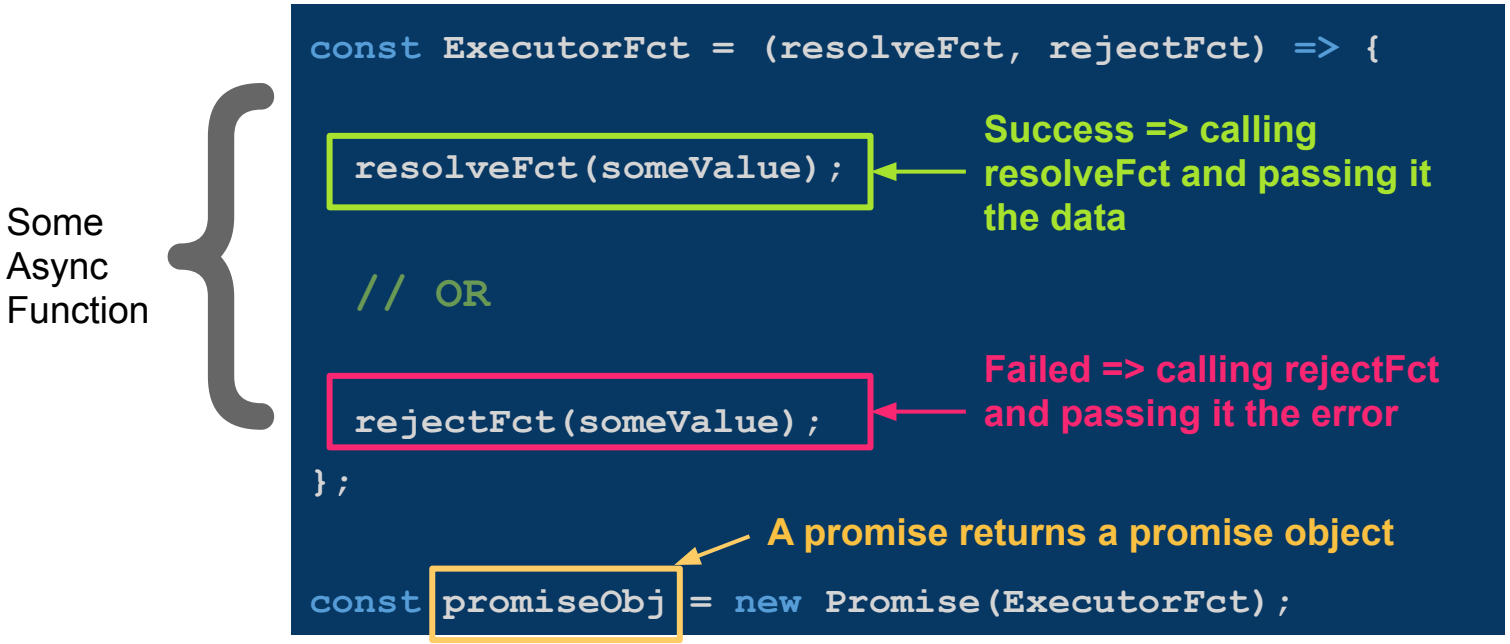
Promises

How to use a promise:

1. Create an **executor function** to create a promise
2. **Consuming** the promise (using it)

Promises

The **Executor** Function:



Promises

Consuming the promise:

```
const ExecutorFct = (resolveFct, rejectFct) => {

  resolveFct(someValue);

  // OR

  rejectFct(someValue);
};

const promiseObj = new Promise(ExecutorFct);
```

Consuming the promise

```
promiseObj
  .then((result) => console.log(result))
  .catch((err) => console.log(err));
```

Promises

Promise.all

Use Promise.all when we need to wait for all of the promises to be resolved.

```
Promise.all([getUser(), getGreeting(), getUser()])
  .then((values) =>
    console.log(`${values[0]} says: ${values[1]} ${values[2]}`)
  )
  .catch((err) => console.log(err));
```

Questions?

LIGHTHOUSE LABS