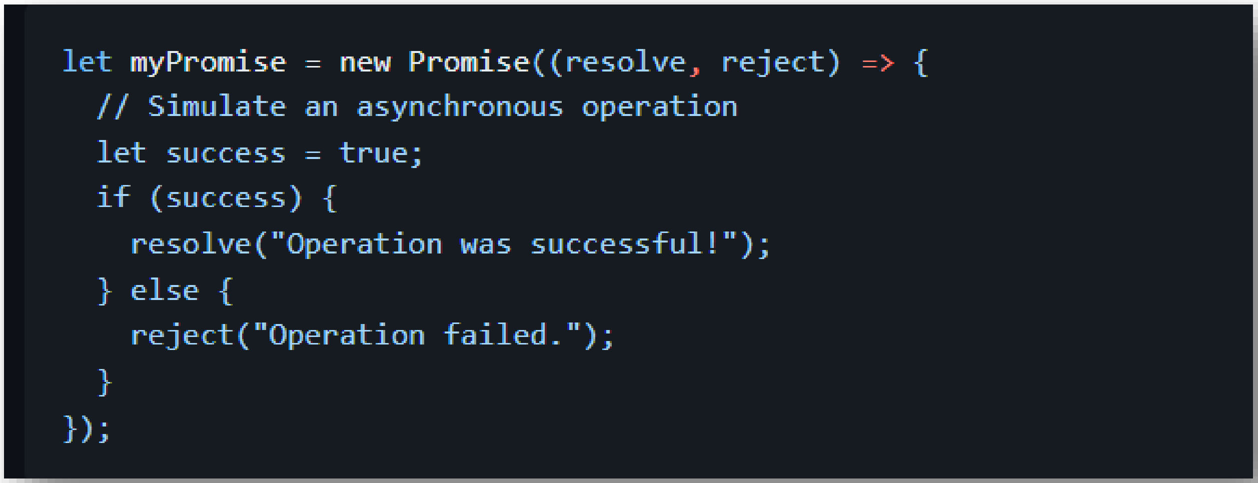
Sure! In JavaScript, a **Promise** is like a placeholder for a value that will be available in the future. It's used to handle asynchronous operations, such as fetching data from a server, without blocking the rest of your code.

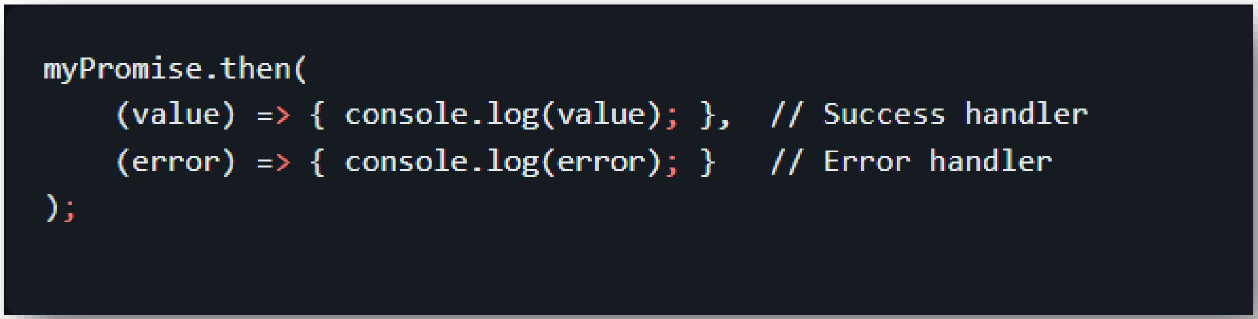
Here's a simple breakdown:

1. **Creating a Promise**: You create a promise using the `Promise` constructor, which takes a function with two arguments: `resolve` and `reject`. These are callbacks that you call when the operation succeeds or fails.



```
let myPromise = new Promise((resolve, reject) => {
  // Simulate an asynchronous operation
  let success = true;
  if (success) {
    resolve("Operation was successful!");
  } else {
    reject("Operation failed.");
  }
});
```

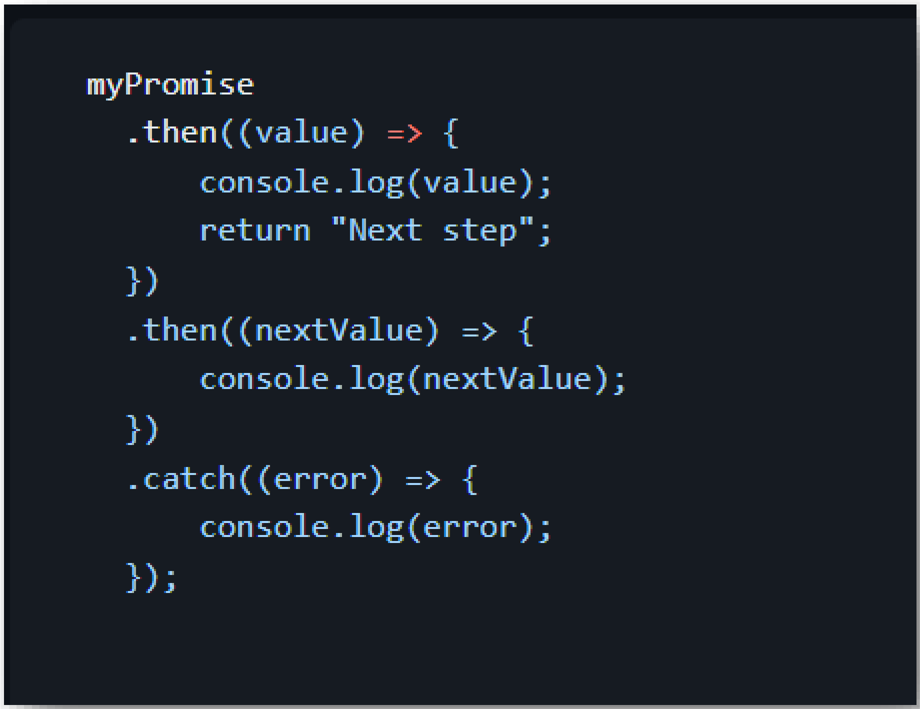
2. **Using a Promise**: You use the `.then()` method to specify what should happen when the promise is resolved (successful) or rejected (failed).



```
myPromise.then(
    (value) => { console.log(value); },  // Success handler
    (error) => { console.log(error); }   // Error handler
);
```

3. **States of a Promise**:
    - **Pending**: The initial state, neither fulfilled nor rejected.

    - **Fulfilled**: The operation completed successfully.
    - **Rejected**: The operation failed.
4. **Chaining Promises**: You can chain multiple `.then()` calls to handle a sequence of asynchronous operations.



```
myPromise
  .then((value) => {
      console.log(value);
      return "Next step";
  })
  .then((nextValue) => {
      console.log(nextValue);
  })
  .catch((error) => {
      console.log(error);
  });
```