MASTER JAVASCRIPT PROMISE

SWIPE LEFT

What is a Promise?

A promise is a special object representing the eventual completion or failure of an asynchronous operation. It acts as a placeholder for a value yet to be determined.

SWIPE LEFT

Promise States

Promises begin in a pending state, awaiting completion. They then transition to fullfilled on success or rejected on failure. This allows for better control and handling of asynchronous tasks.

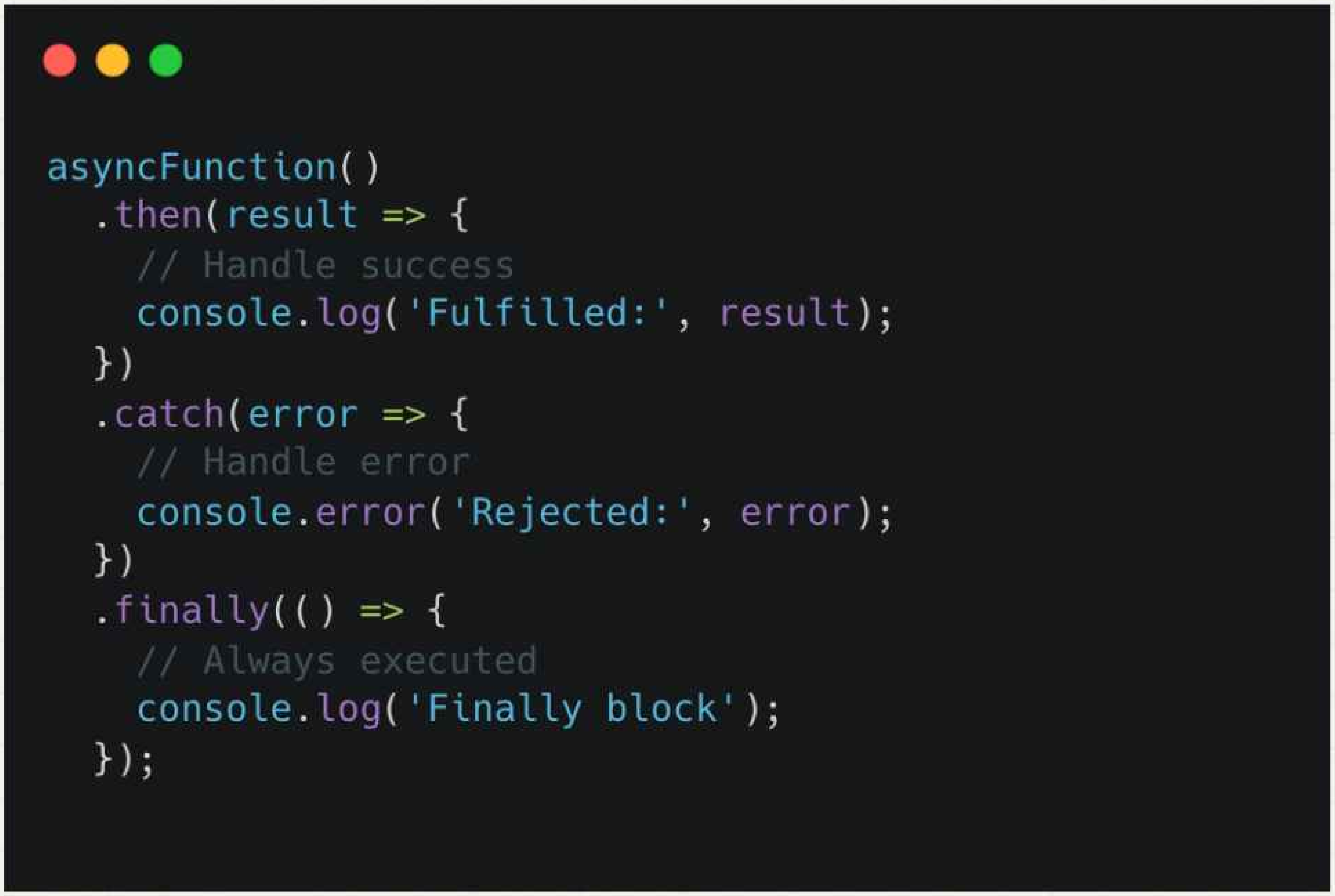
SWIPE LEFT

Promise Handlers

Use then() for success and failure, catch() for handling errors, and finally() for actions that must occur regardless of success or failure. All handlers run as microtasks ensuring efficient execution.

SWIPE LEFT

Promise Handlers



```
asyncFunction()
  .then(result => {
    // Handle success
    console.log('Fulfilled:', result);
  })
  .catch(error => {
    // Handle error
    console.error('Rejected:', error);
  })
  .finally(() => {
    // Always executed
    console.log('Finally block');
  });
```

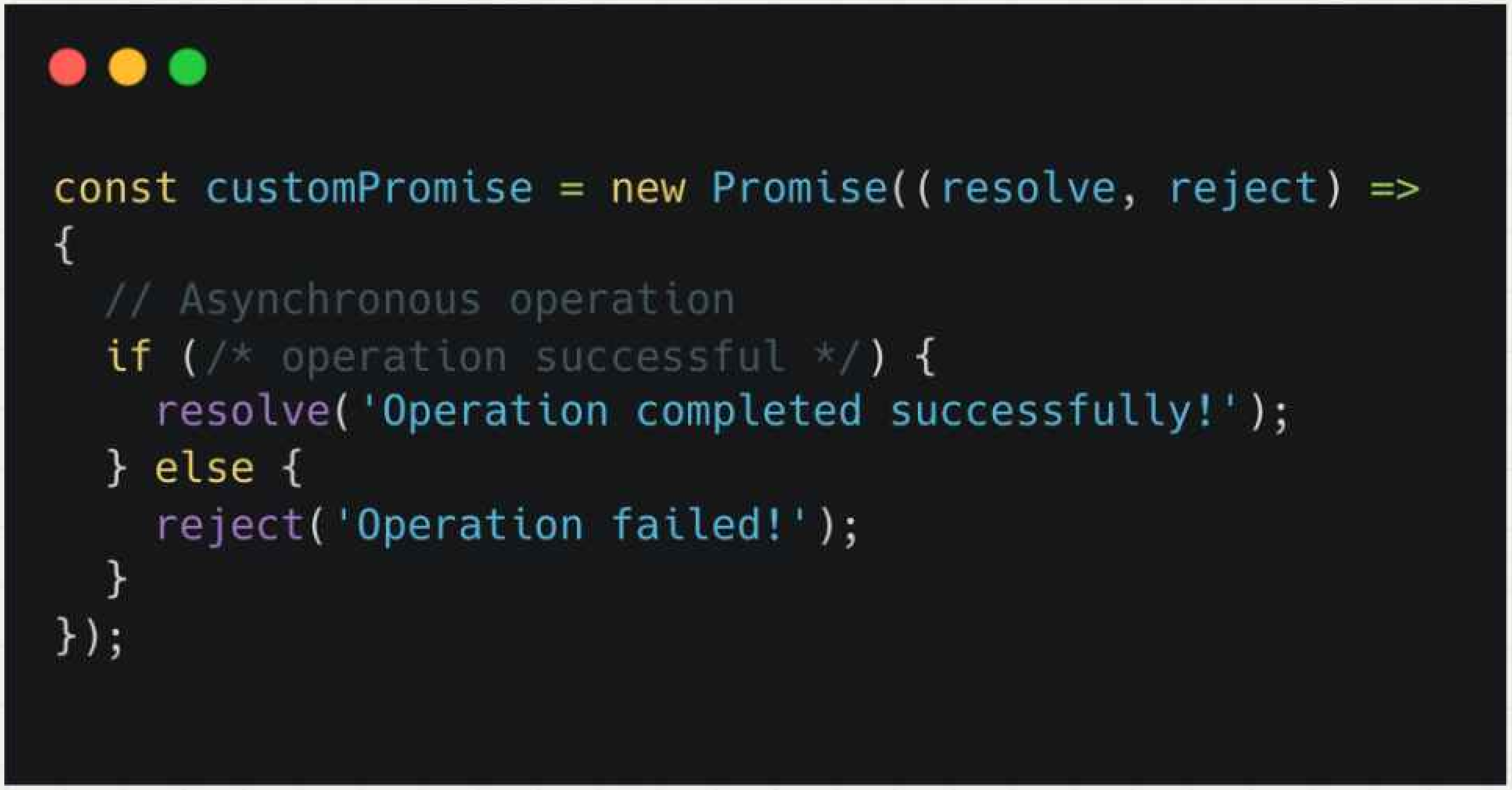
SWIPE LEFT

Creating Promises

Create new promises using the Promise constructor The executor function runs immediately, and any errors thrown within it are automatically caught and passed to reject().

SWIPE LEFT

Creating Promises



```javascript
const customPromise = new Promise((resolve, reject) =>
{
  // Asynchronous operation
  if (/* operation successful */) {
    resolve('Operation completed successfully!');
  } else {
    reject('Operation failed!');
  }
});
```

SWIPE LEFT

Settled Promises

Create settled promises with Promise.resolve() for successful outcomes and Promise.reject() for handling failures. These methods enhance predictability in handling specific cases.

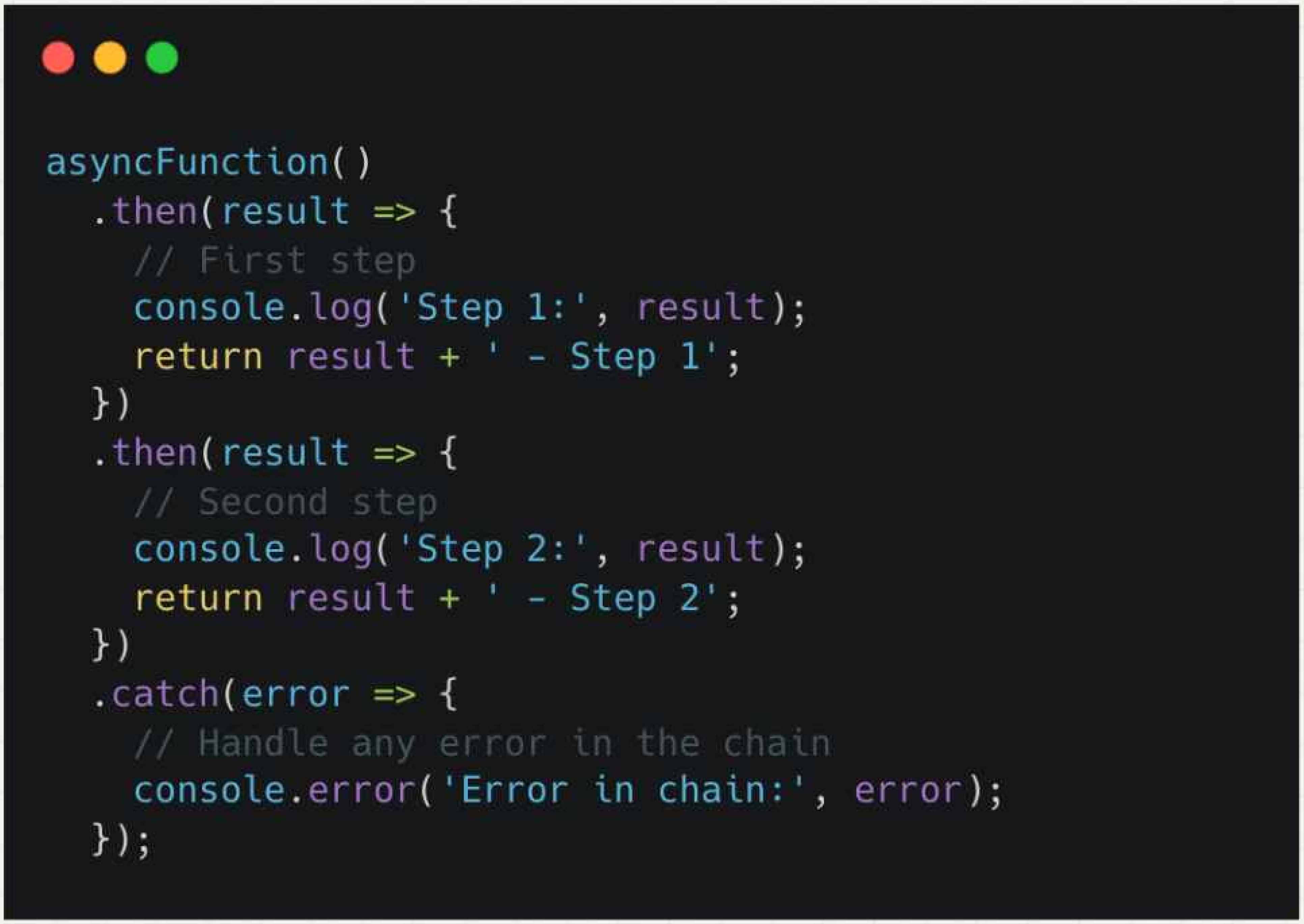
SWIPE LEFT

Promise Chains

Unlock the full potential of promises by chaining them together. Each handler returns a new promise, allowing for a seamless flow of data and operations.

SWIPE LEFT

Promise Chains



```
asyncFunction()
  .then(result => {
    // First step
    console.log('Step 1:', result);
    return result + ' - Step 1';
  })
  .then(result => {
    // Second step
    console.log('Step 2:', result);
    return result + ' - Step 2';
  })
  .catch(error => {
    // Handle any error in the chain
    console.error('Error in chain:', error);
  });
```

SWIPE LEFT

Conclusion

Congratulations! You've leveled up your JavaScript async skills with promises. Keep coding and exploring the endless possibilities of asynchronous programming

SWIPE LEFT

FOLLOW ME

@shivam-dhyani