JS

Async / Await in JavaScript !

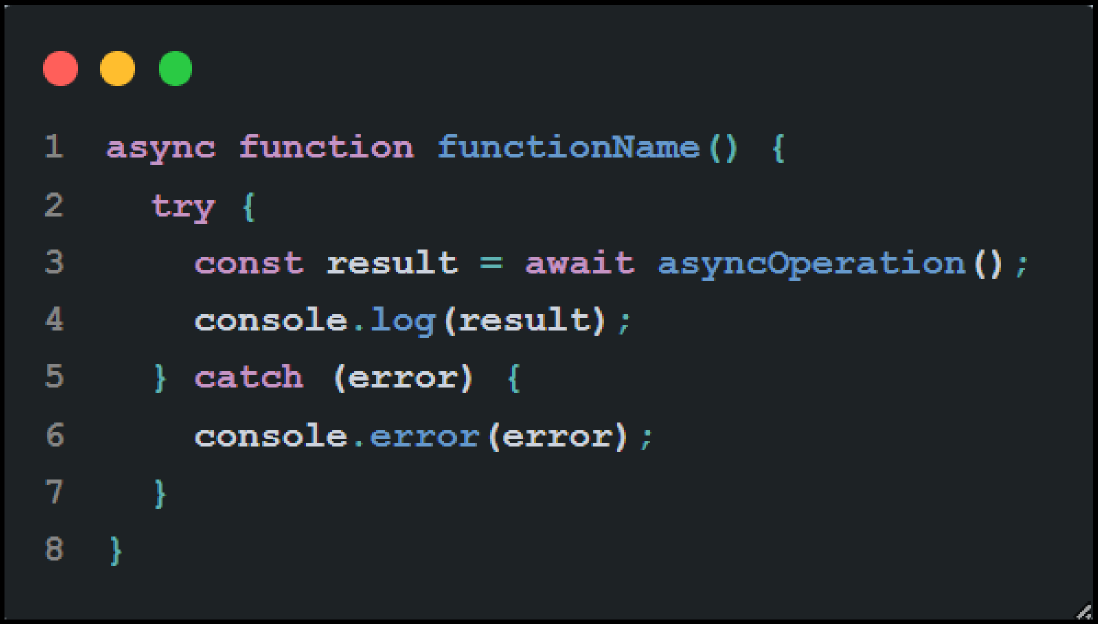
Streamline Your Code! 🚀💡

Jimmy Ramani
@jimmyramani

Async / Await :

Certainly! Asynchronous programming is an important concept in JavaScript, and the async/await syntax provides a more readable and structured way to work with asynchronous code. It's used to manage promises and streamline code that deals with operations that take time to complete, such as network requests or file I/O. Here's an overview of how async/await works and some examples to help you understand and use it effectively.

Basic Syntax :



```
async function functionName() {
  try {
    const result = await asyncOperation();
    console.log(result);
  } catch (error) {
    console.error(error);
  }
}
```

Jimmy Ramani

@jimmyramani

async Function :

The async keyword is used before a function declaration to indicate that the function will always return a promise. This allows you to use the await keyword inside the function.

await Keyword :

The await keyword is used within an async function to pause the execution of the function until the promise is resolved or rejected. It can only be used inside an async function.

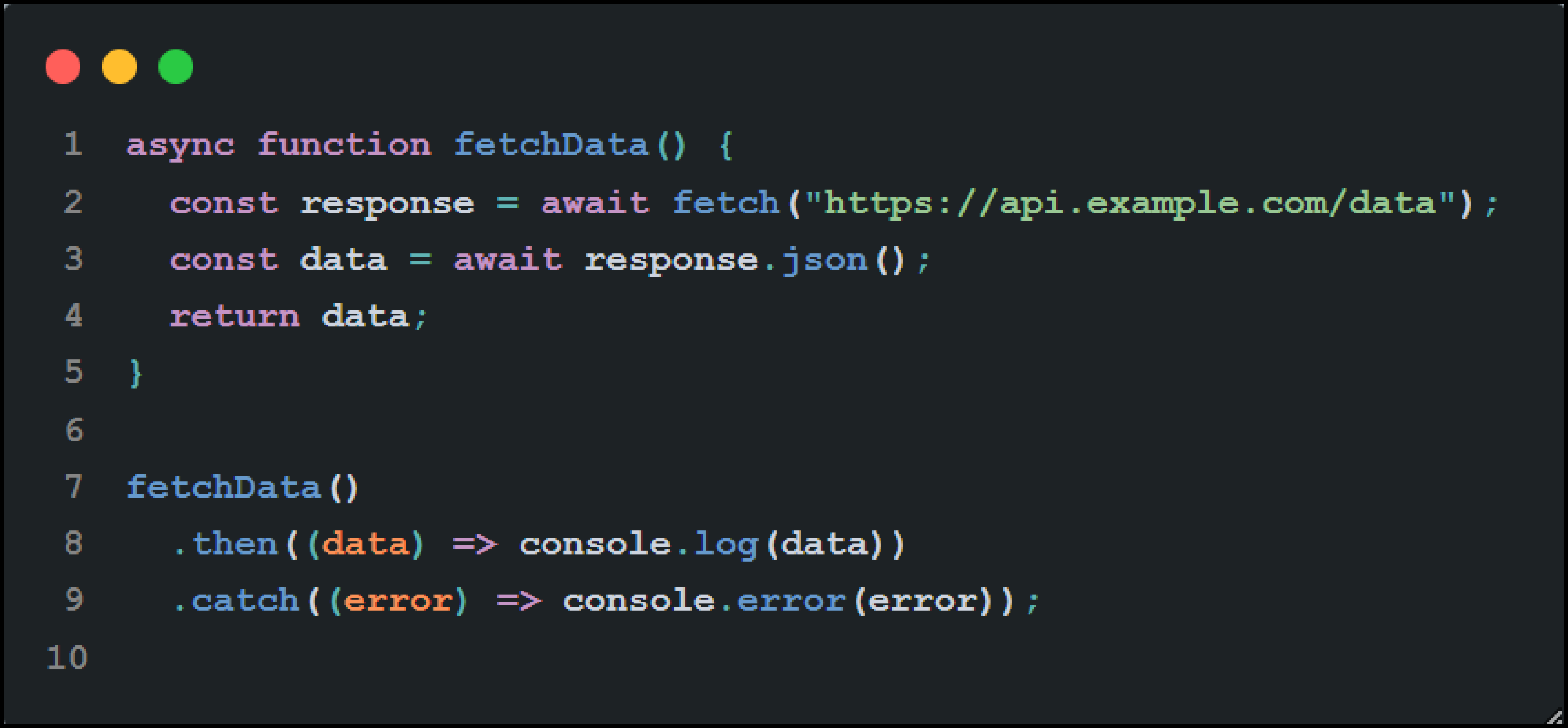
Error Handling :

You can use a try...catch block to handle errors that might occur when using await.

Jimmy Ramani
@jimmyramani

Example

Basic Async Function

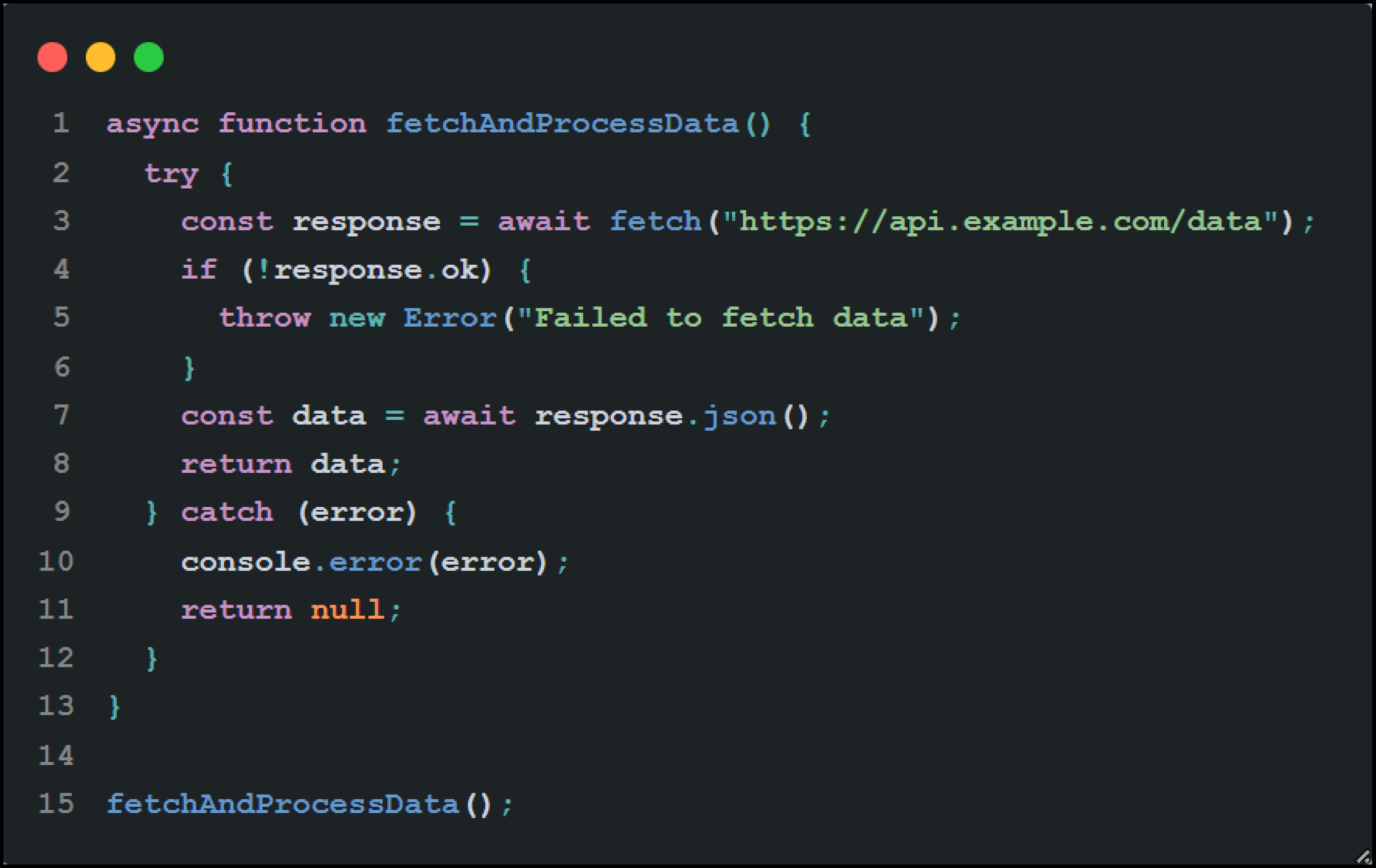


```javascript
async function fetchData() {
  const response = await fetch("https://api.example.com/data");
  const data = await response.json();
  return data;
}

fetchData()
  .then((data) => console.log(data))
  .catch((error) => console.error(error));
```

Jimmy Ramani
@jimmyramani

Example

Error Handling

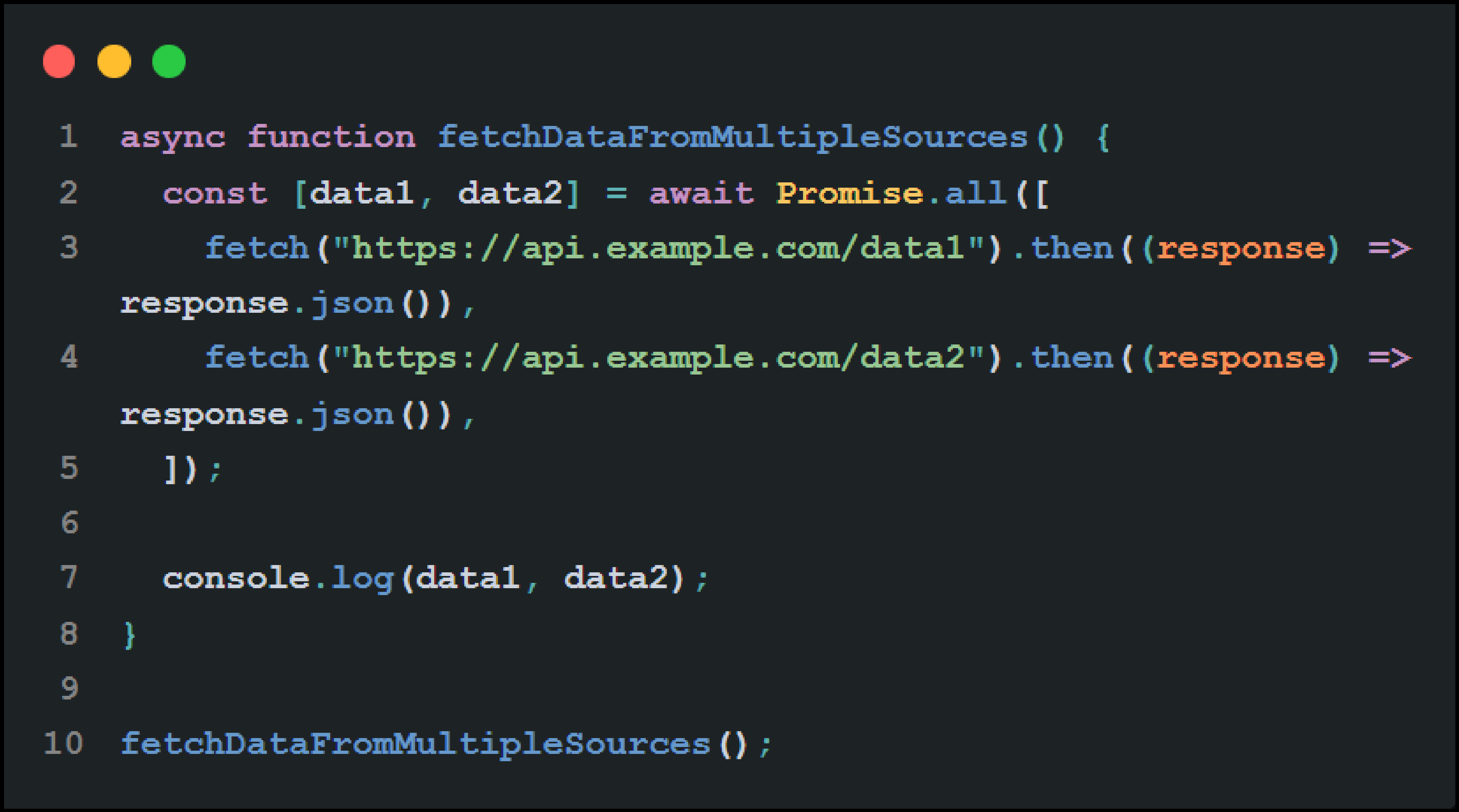


```
async function fetchAndProcessData() {
  try {
    const response = await fetch("https://api.example.com/data");
    if (!response.ok) {
      throw new Error("Failed to fetch data");
    }
    const data = await response.json();
    return data;
  } catch (error) {
    console.error(error);
    return null;
  }
}

fetchAndProcessData();
```

Jimmy Ramani
@jimmyramani

Example

Parallel Async Operations

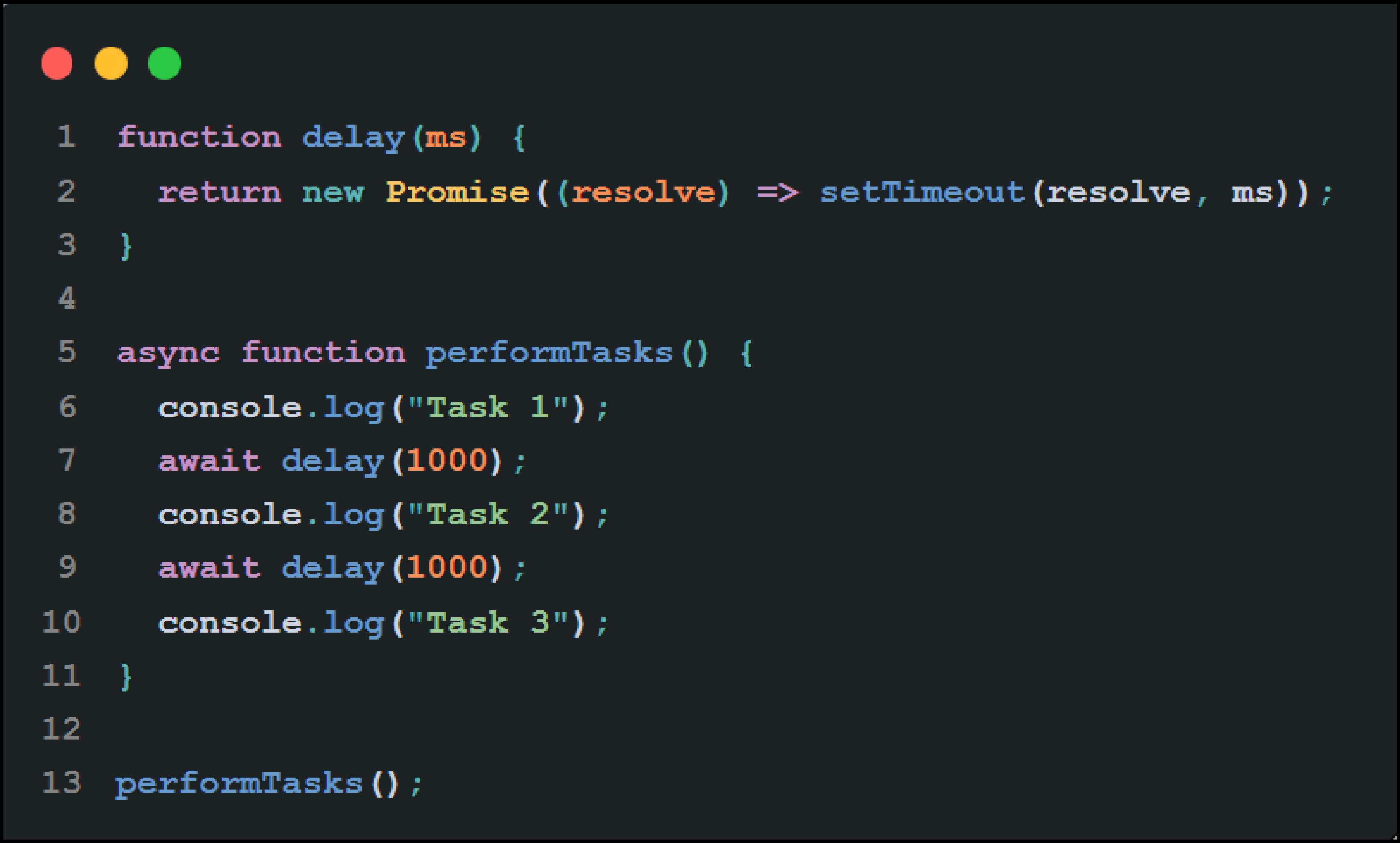


```javascript
async function fetchDataFromMultipleSources() {
  const [data1, data2] = await Promise.all([
    fetch("https://api.example.com/data1").then((response) => response.json()),
    fetch("https://api.example.com/data2").then((response) => response.json()),
  ]);

  console.log(data1, data2);
}

fetchDataFromMultipleSources();
```

Jimmy Ramani
@jimmyramani

Example

Using async/await with Promise



```javascript
function delay(ms) {
  return new Promise((resolve) => setTimeout(resolve, ms));
}

async function performTasks() {
  console.log("Task 1");
  await delay(1000);
  console.log("Task 2");
  await delay(1000);
  console.log("Task 3");
}

performTasks();
```

Jimmy Ramani
@jimmyramani

WAS THIS HELPFUL?

Share with a friend who needs it!

Jimmy Ramani

@jimmyramani