Mehdi Bouchachi

THE COMPLETE FRONT-END DEVELOPMENT

SECTION

JAVASCRIPT REVEIW

LECTURE

ASYNCHRONOUS JAVASCRIPT: PROMISES

@Mehdibouchachi

ASYNCHRONOUS JAVASCRIPT: PROMISES

BEFORE WE START

👉 JavaScript, as a synchronous language, executes its code line by line. However, when dealing with **asynchronous tasks** such as fetching data from a server, traditional synchronous programming approaches can lead to blocking behavior, where subsequent code is halted while waiting for the asynchronous operation to complete. This can result in inefficient code and a poor user experience

👉 For this problem, JavaScript introduces Promises as the solution for handling asynchronous operations

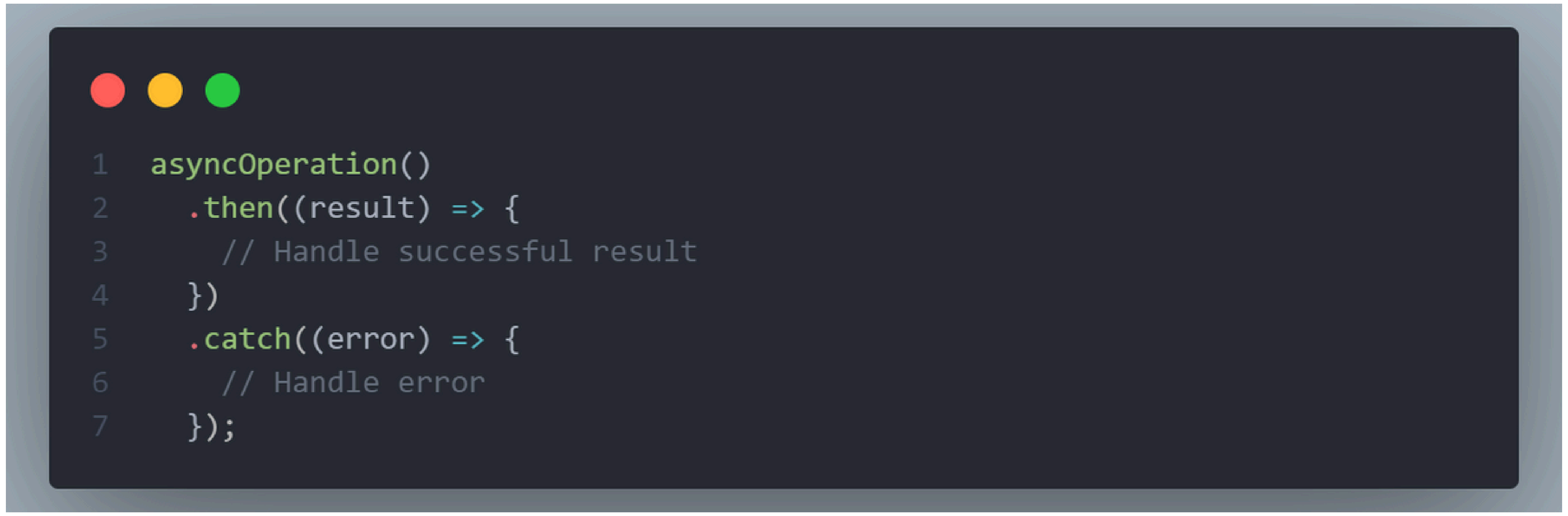
ASYNCHRONOUS JAVASCRIPT: PROMISES

JAVASCRIPT PROMISES

👉 A **promise** represents the eventual **completion** or **failure** of an asynchronous operation and its resulting value

👉 It can be in one of the three states: pending, fulfilled, or rejected.

👉 Promises are utilized with a **.then** method to handle the resolved value, and the **.catch** method is used to handle error cases.



```
asyncOperation()
  .then((result) => {
    // Handle successful result
  })
  .catch((error) => {
    // Handle error
  });
```

ASYNCHRONOUS JAVASCRIPT: PROMISES

👉 In this presentation, we'll use **JSONPlaceholder** as an example to demonstrate how JavaScript promises can be used to fetch data from an external API asynchronously. This practical example will illustrate the real-world application of promises in web development.

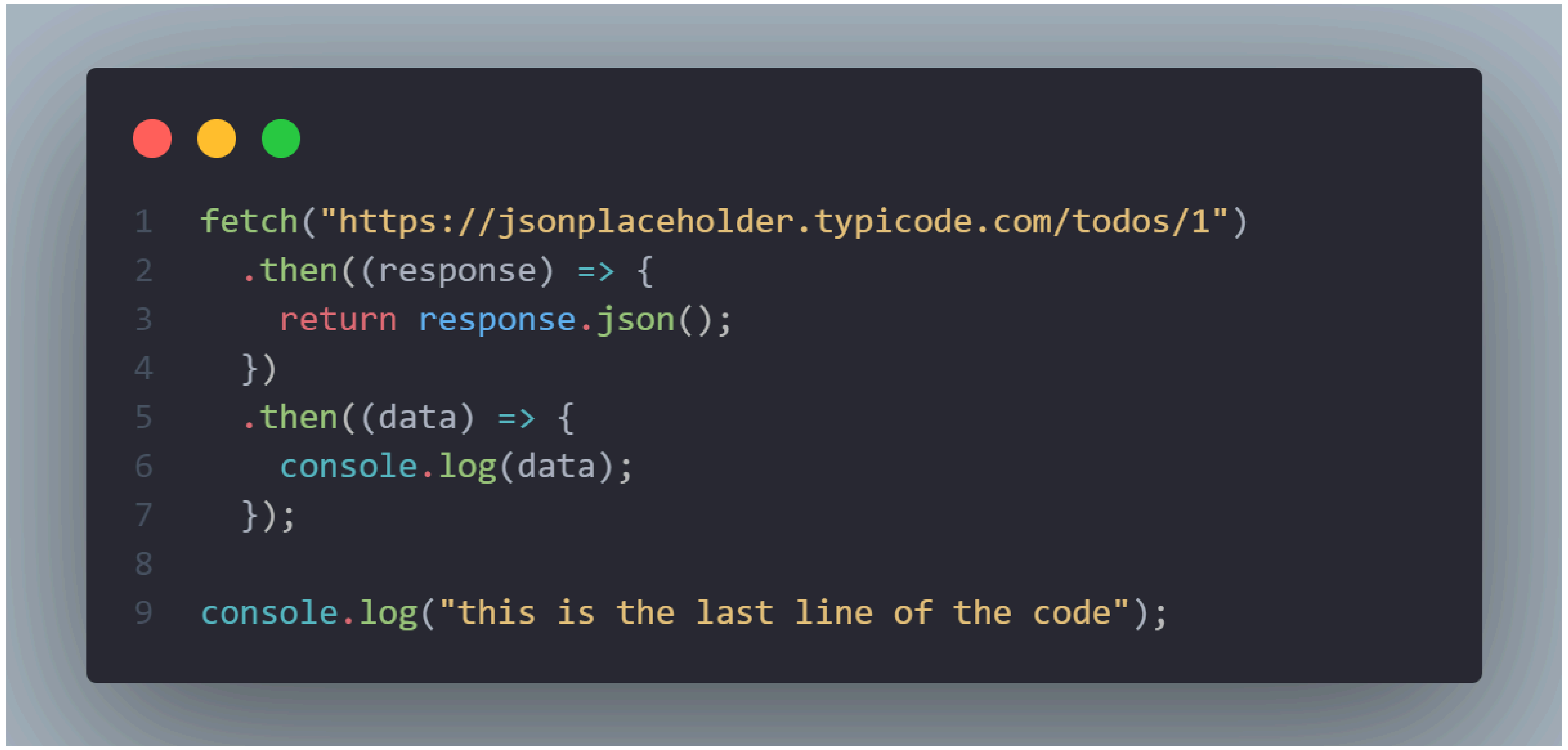
👉 The fetch function is used to make **asynchronous HTTP requests** to fetch resources from a server. It **returns a promise** that resolves to the Response object representing the response to the request.

👉 **JSONPlaceholder** API serves as a useful tool to simulate server responses and test client-side applications without relying on a real backend.

JSONPlaceholder

ASYNCHRONOUS JAVASCRIPT: PROMISES

👉 Below is an example of JavaScript code using promises to fetch data from the JSONPlaceholder API asynchronously:



```javascript
fetch("https://jsonplaceholder.typicode.com/todos/1")
  .then((response) => {
    return response.json();
  })
  .then((data) => {
    console.log(data);
  });

console.log("this is the last line of the code");
```

👉 Due to the asynchronous nature of promises, the console log result at the end may appear before the result of the fetch function.

ASYNCHRONOUS JAVASCRIPT: ASYNC/AWAIT

PROBLEM

👉 Writing asynchronous code with promises using traditional promise chaining can sometimes lead to **complex** and **hard-to-maintain** code. Asynchronous operations often involve nesting multiple **.then()** callbacks, resulting in what's commonly known as 'callback hell'. This can make code difficult to read, understand, and debug.

👉 Async/await provides a modern solution to this problem by offering a cleaner and more readable alternative to promise chaining

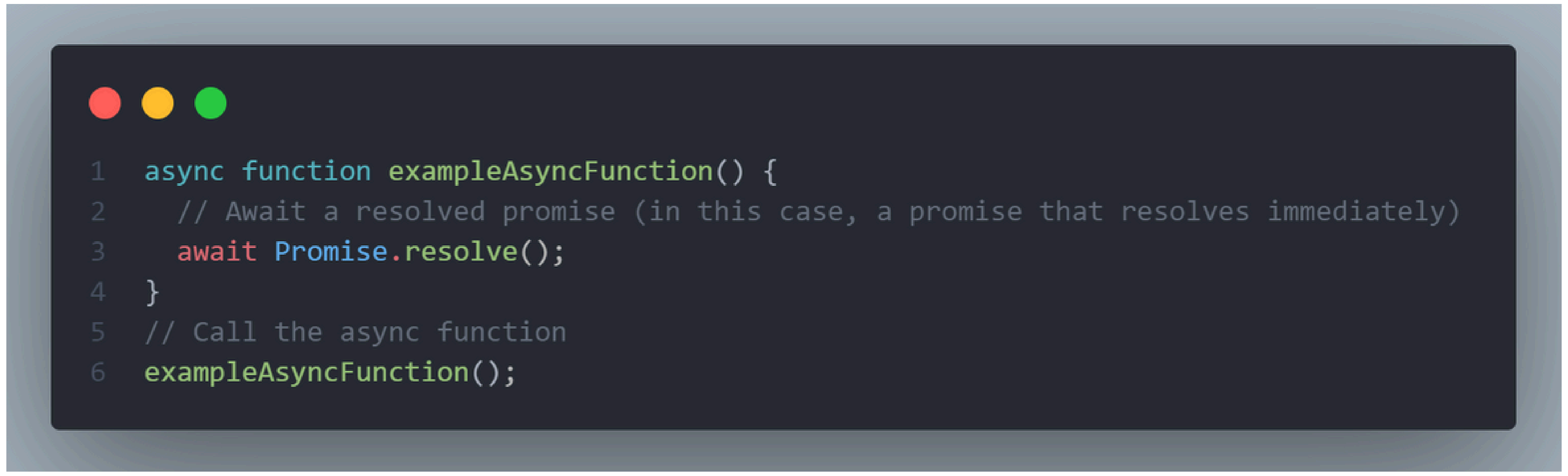
ASYNCHRONOUS JAVASCRIPT: ASYNC/AWAIT

ASYNC / AWAIT

👉 **Async functions**, declared with the **async** keyword, allow the use of **await** within their bodies. The await keyword **suspends execution until promises resolve**

👉 It can be in one of the three states: pending, fulfilled, or rejected.

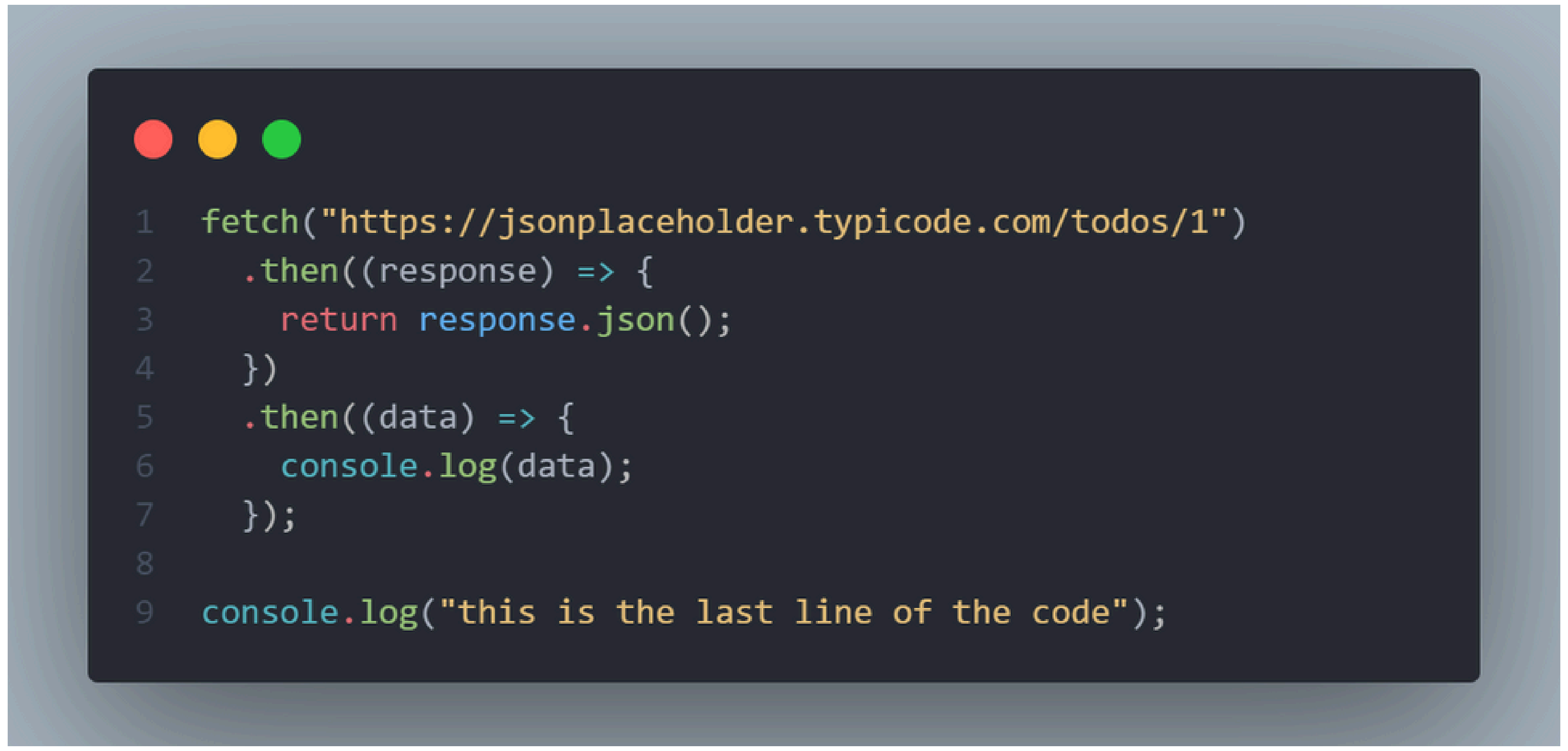
👉 The syntax is straightforward: async **declares** the function as **asynchronous**, while await is used **before an instruction** to pause execution until the promise resolves



```
async function exampleAsyncFunction() {
  // Await a resolved promise (in this case, a promise that resolves immediately)
  await Promise.resolve();
}
// Call the async function
exampleAsyncFunction();
```

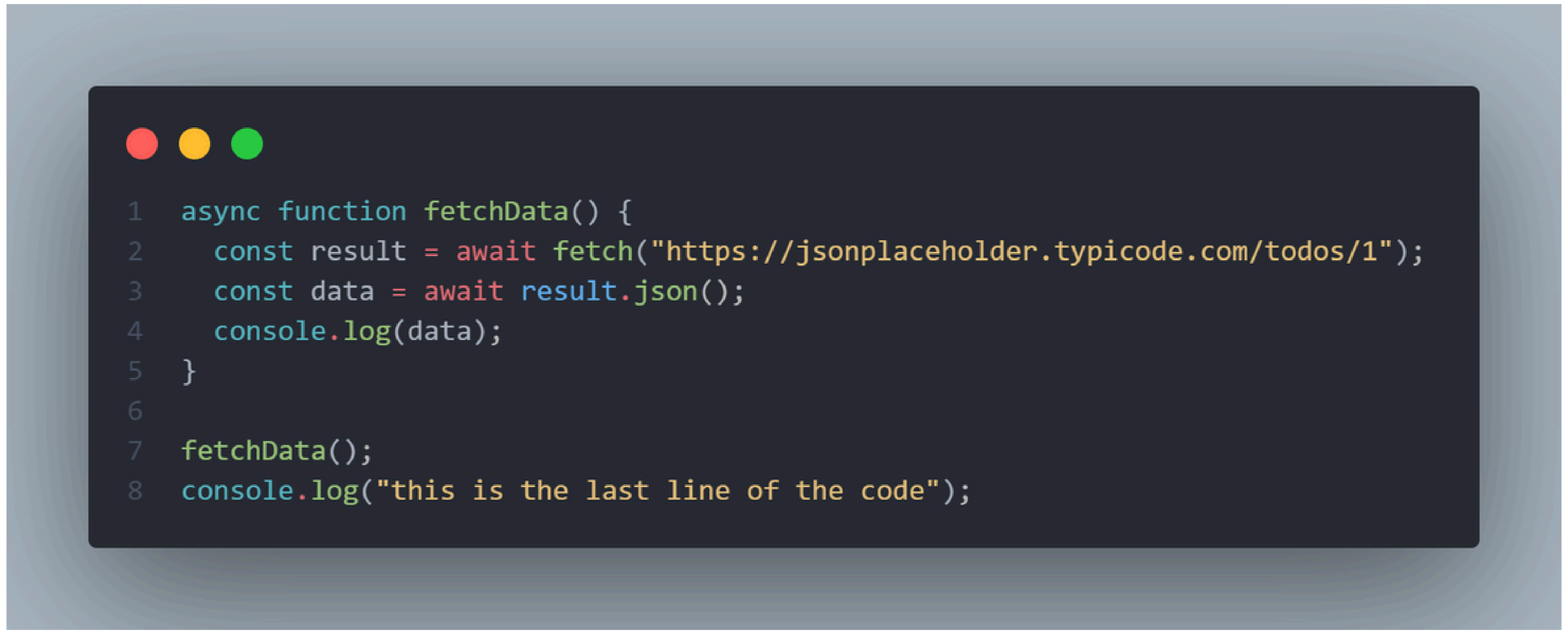
ASYNCHRONOUS JAVASCRIPT: ASYNC/AWAIT

👉 The following code snippets achieve the same task using different approaches: one utilizing promises, and the other employing async/await syntax :



```javascript
fetch("https://jsonplaceholder.typicode.com/todos/1")
  .then((response) => {
    return response.json();
  })
  .then((data) => {
    console.log(data);
  });

console.log("this is the last line of the code");
```



```javascript
async function fetchData() {
  const result = await fetch("https://jsonplaceholder.typicode.com/todos/1");
  const data = await result.json();
  console.log(data);
}

fetchData();
console.log("this is the last line of the code");
```

SEE YOU SOON...