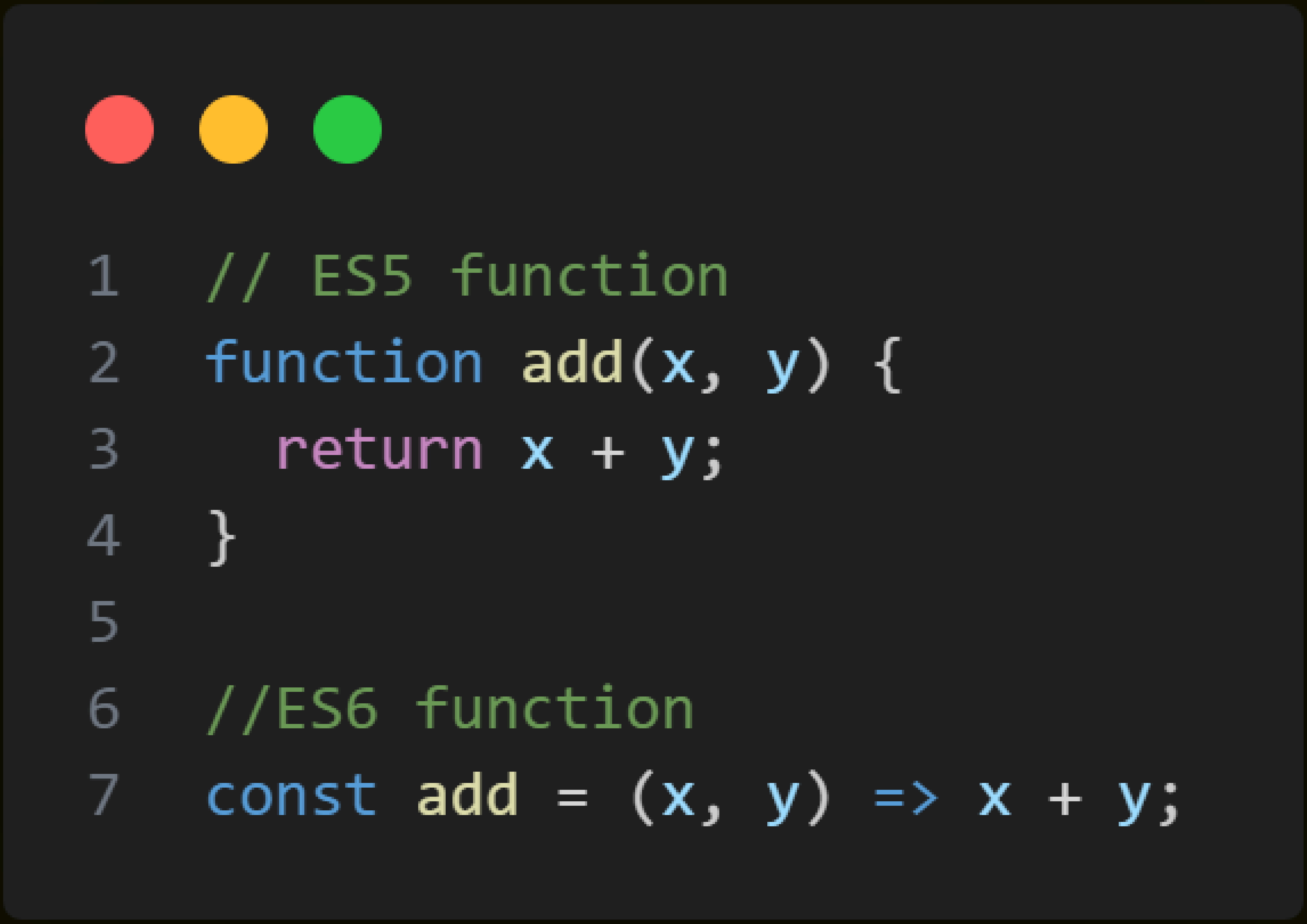
JavaScript {ES6} Features

save
post

swipe
to know

Arrow Functions



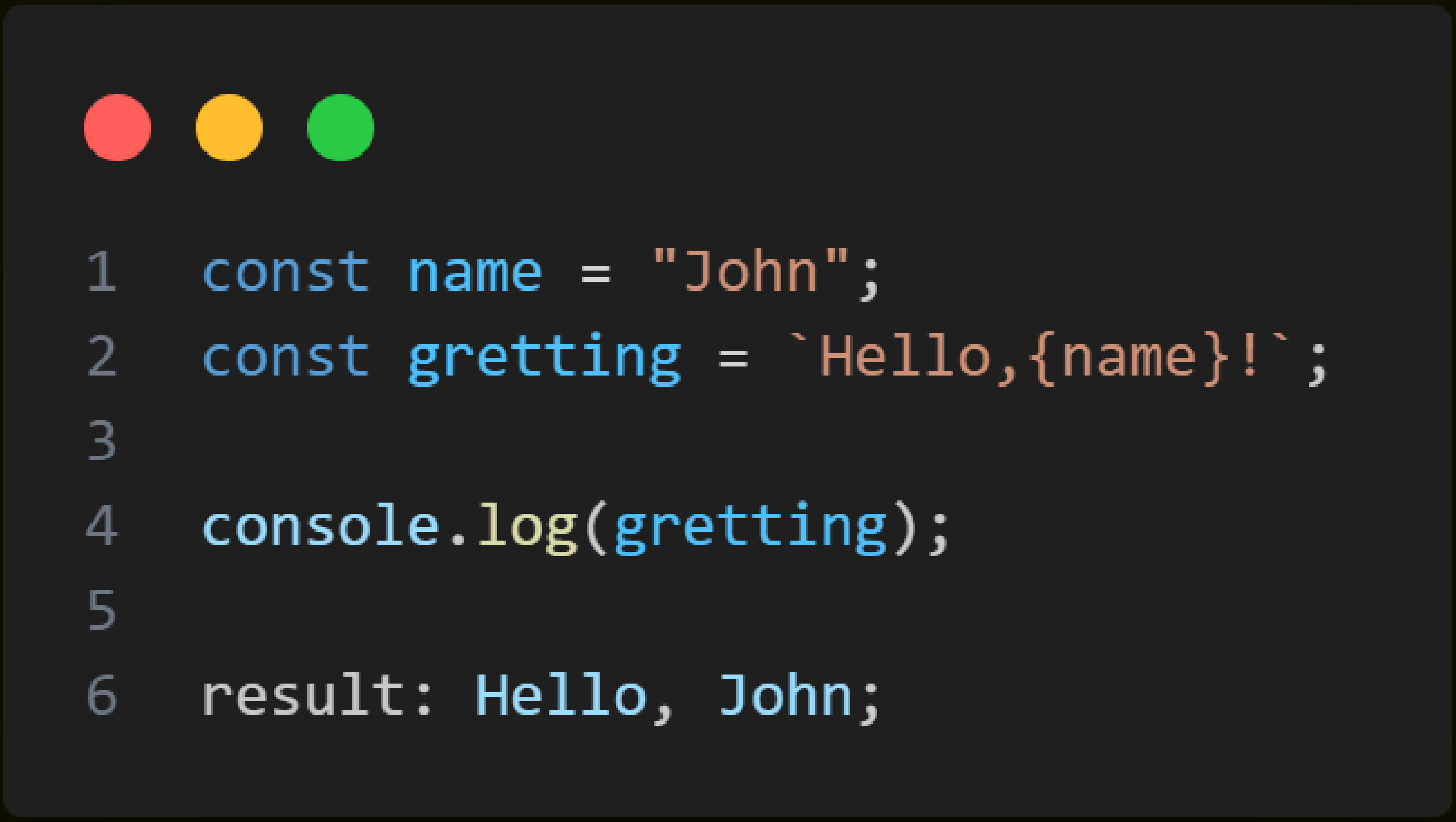
```javascript
// ES5 function
function add(x, y) {
  return x + y;
}

//ES6 function
const add = (x, y) => x + y;
```

Explanation: **Arrow Function** provides a concise syntax for writing functions, especially useful for **short, one-line** operations.

save post

swipe to know

Template Literals



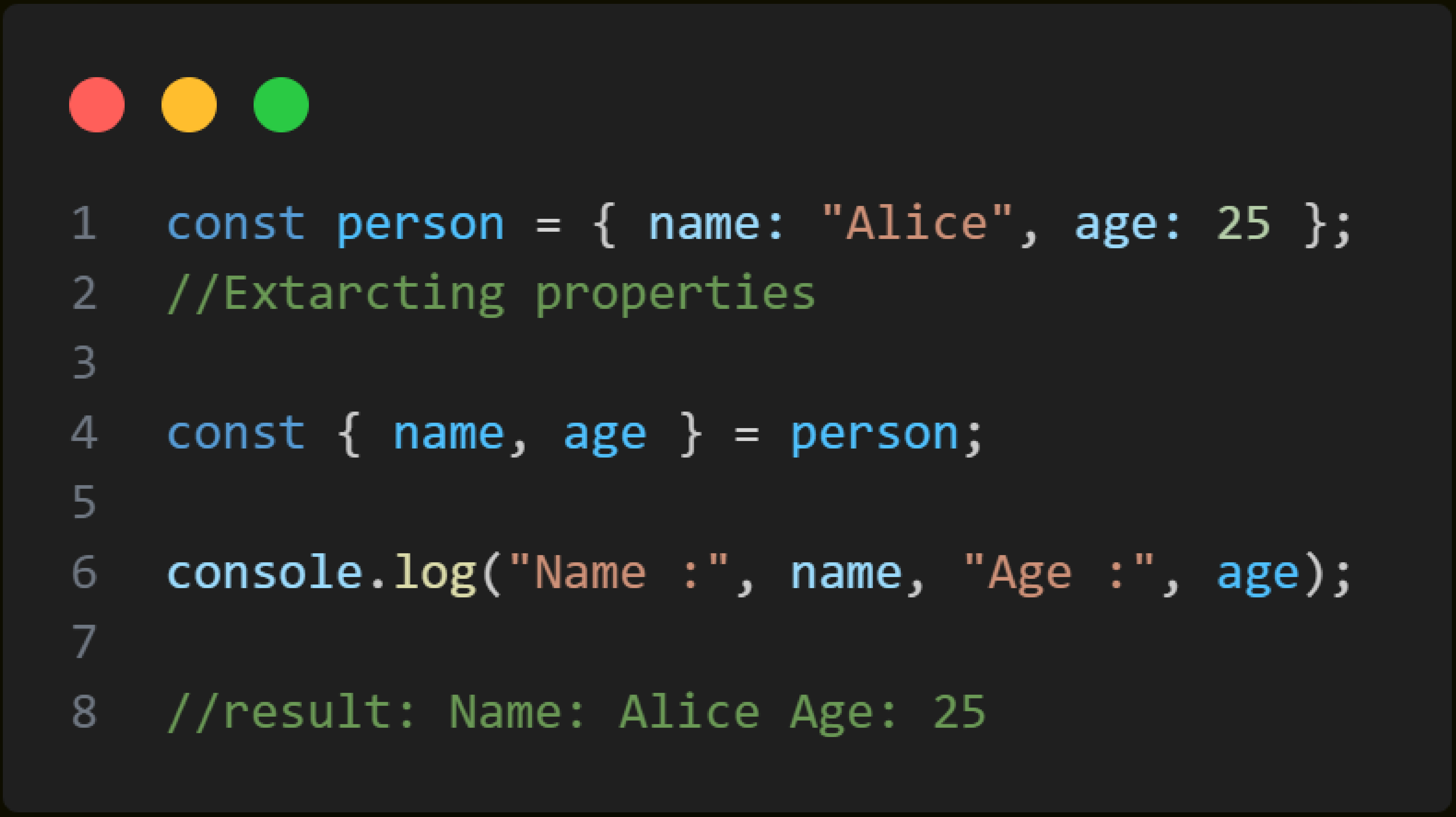
```
const name = "John";
const gretting = `Hello,{name}!`;

console.log(gretting);

result: Hello, John;
```

Explanation: Template literals allow embedding expressions inside strings, providing a **cleaner** and more **readable** way to concatenate strings.

save post

swipe to know

Destructuring Assignment



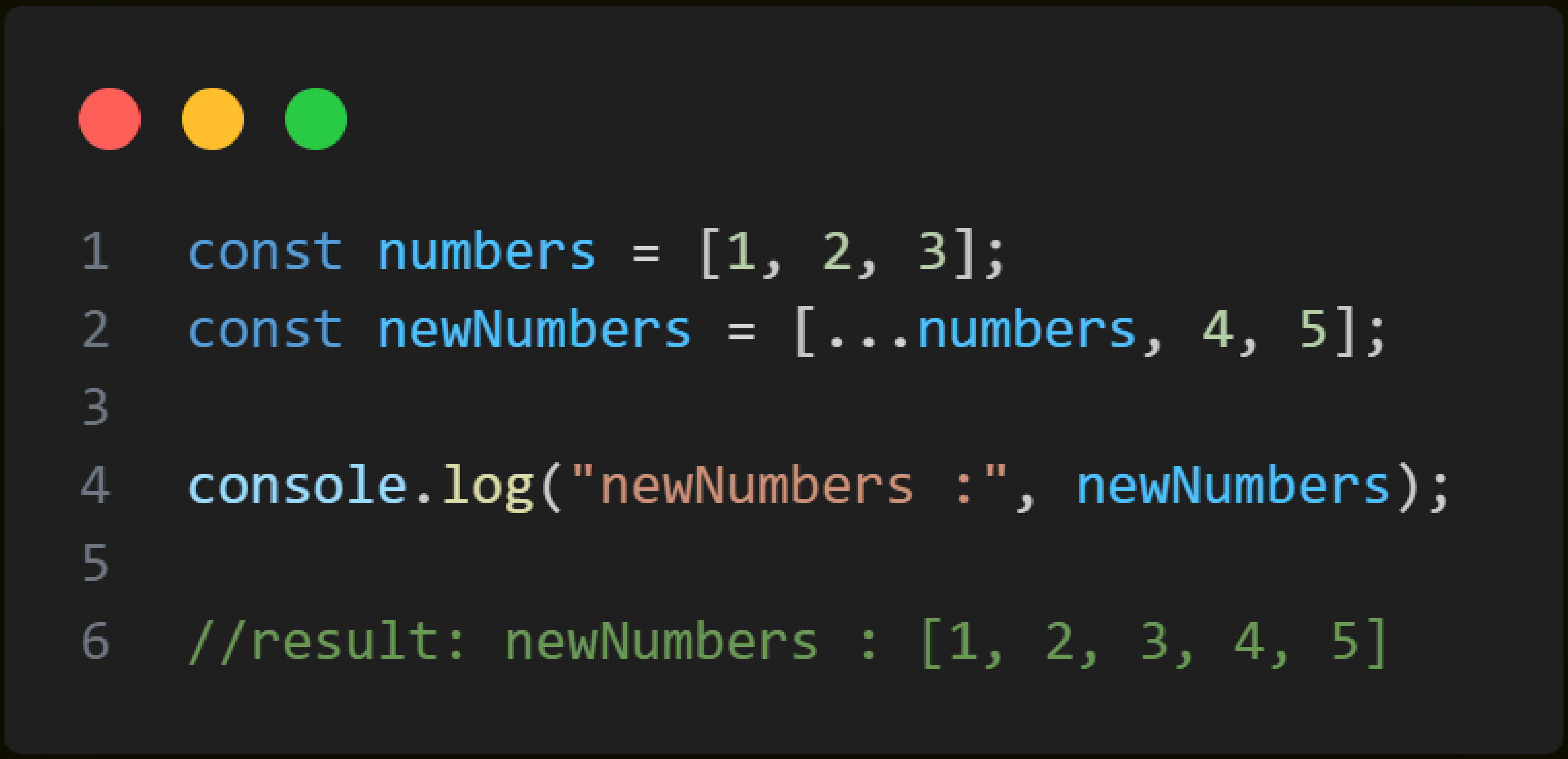
```
const person = { name: "Alice", age: 25 };
//Extarcting properties

const { name, age } = person;

console.log("Name :", name, "Age :", age);

//result: Name: Alice Age: 25
```

Explanation: Destructuring assignment simplifies the **extraction of values** from **objects** or **arrays** into **individual** variables.

save post

swipe to know

Spread Operator



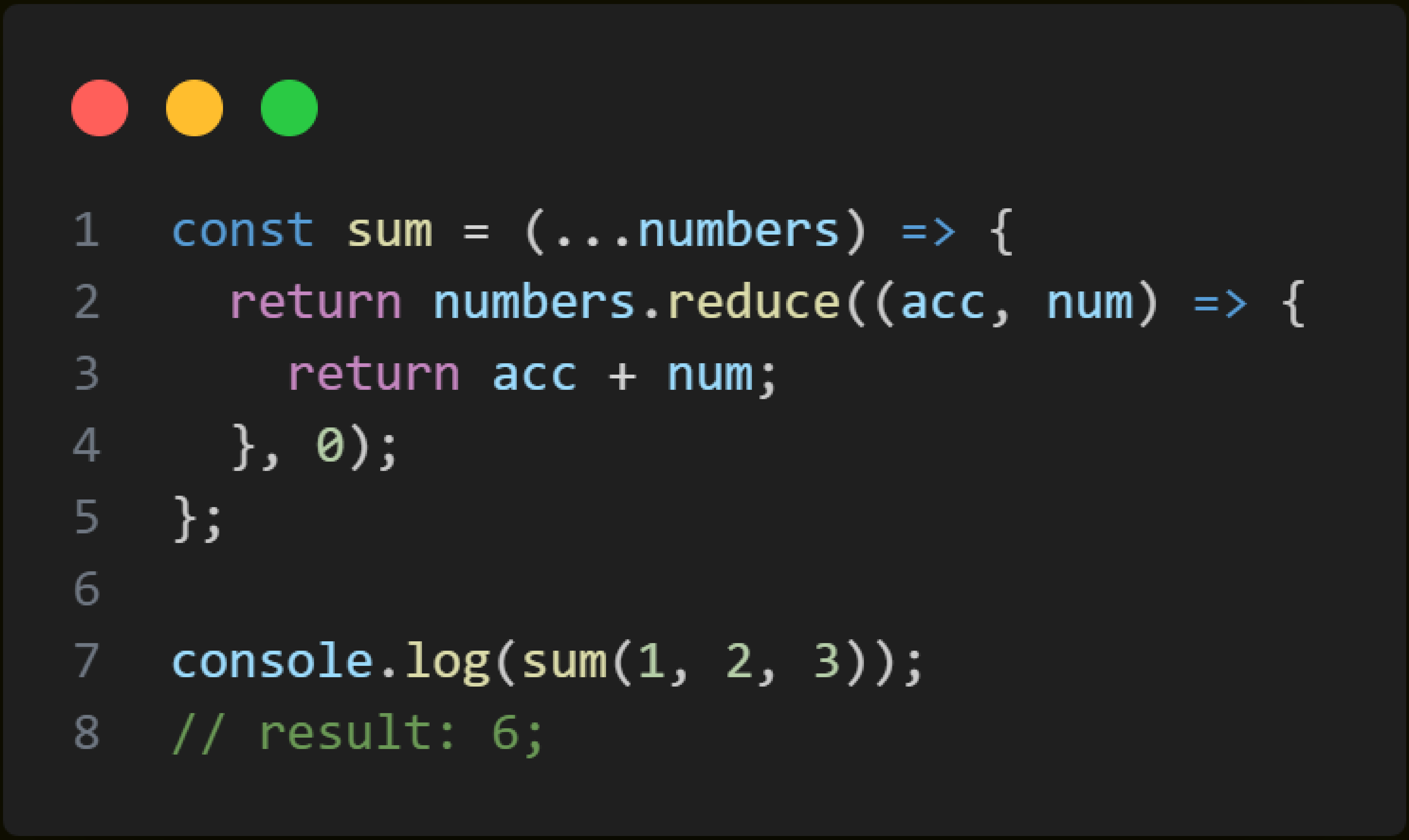
```
const numbers = [1, 2, 3];
const newNumbers = [...numbers, 4, 5];

console.log("newNumbers :", newNumbers);

//result: newNumbers : [1, 2, 3, 4, 5]
```

Explanation: **The spread operator** allows for the expansion of elements making it handy for creating **new arrays** or **objects** based on existing ones.

save post

swipe to know

Rest Parameter



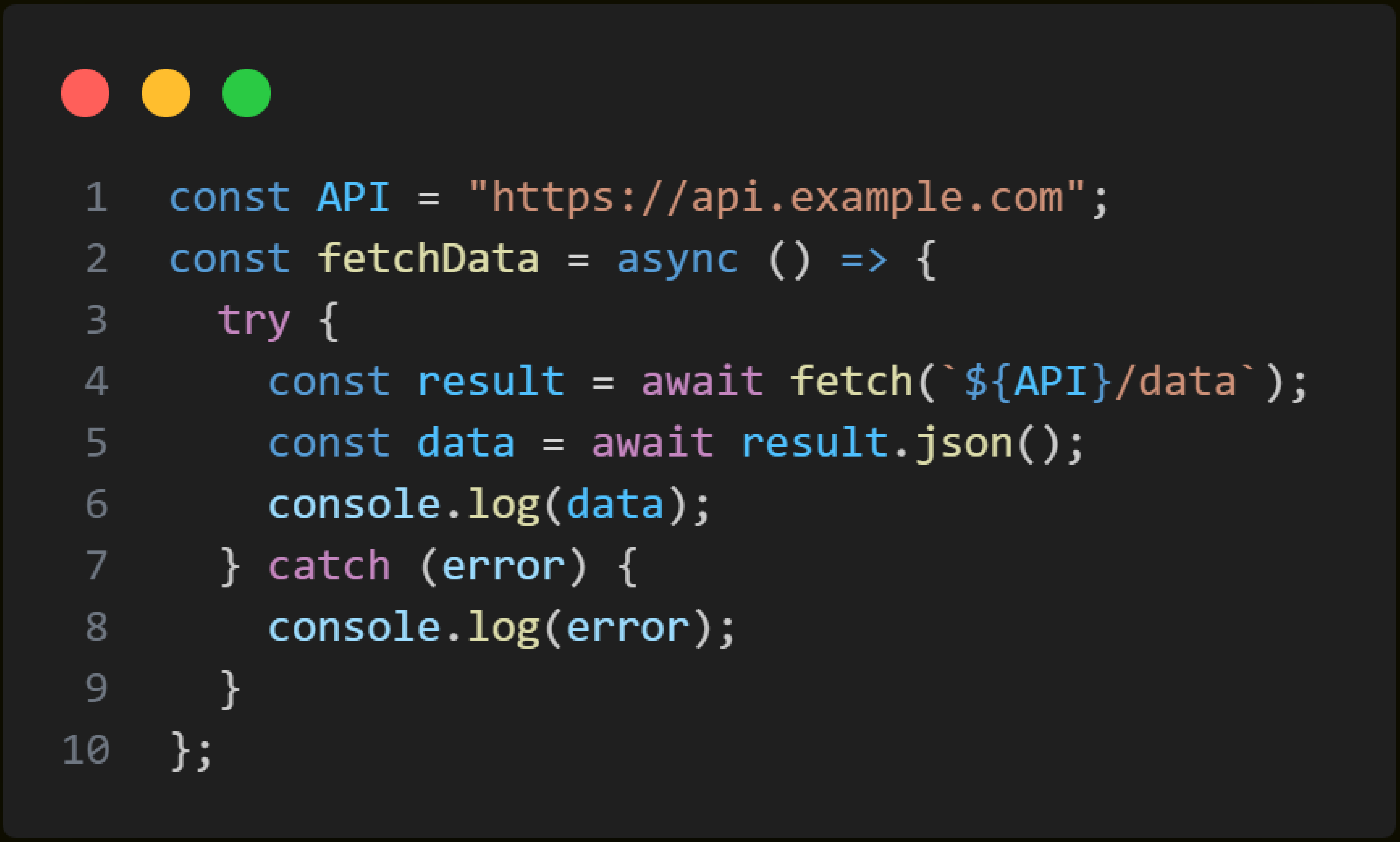
```javascript
const sum = (...numbers) => {
  return numbers.reduce((acc, num) => {
    return acc + num;
  }, 0);
};

console.log(sum(1, 2, 3));
// result: 6;
```

Explanation: The **rest parameter** allows functions to accept an **indefinite** number of arguments as an array, simplifying parameter handling.

save post

swipe to know

Async / Await



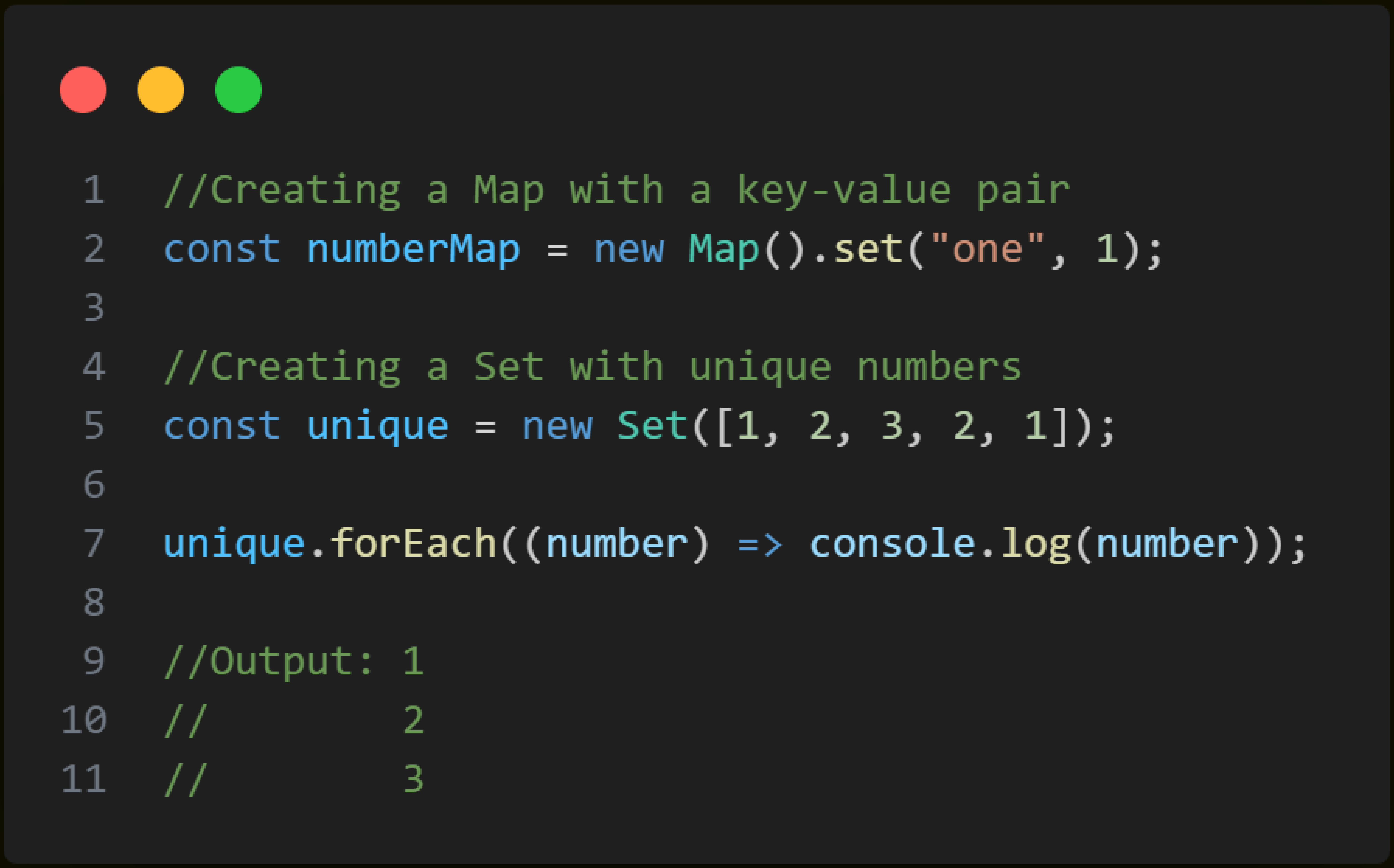
```javascript
const API = "https://api.example.com";
const fetchData = async () => {
  try {
    const result = await fetch(`${API}/data`);
    const data = await result.json();
    console.log(data);
  } catch (error) {
    console.log(error);
  }
};
```

Explanation: **Async/await** is a syntax for handling **asynchronous** code more concisely, providing a cleaner alternative to working with Promise.

save post

swipe to know

Map & Set



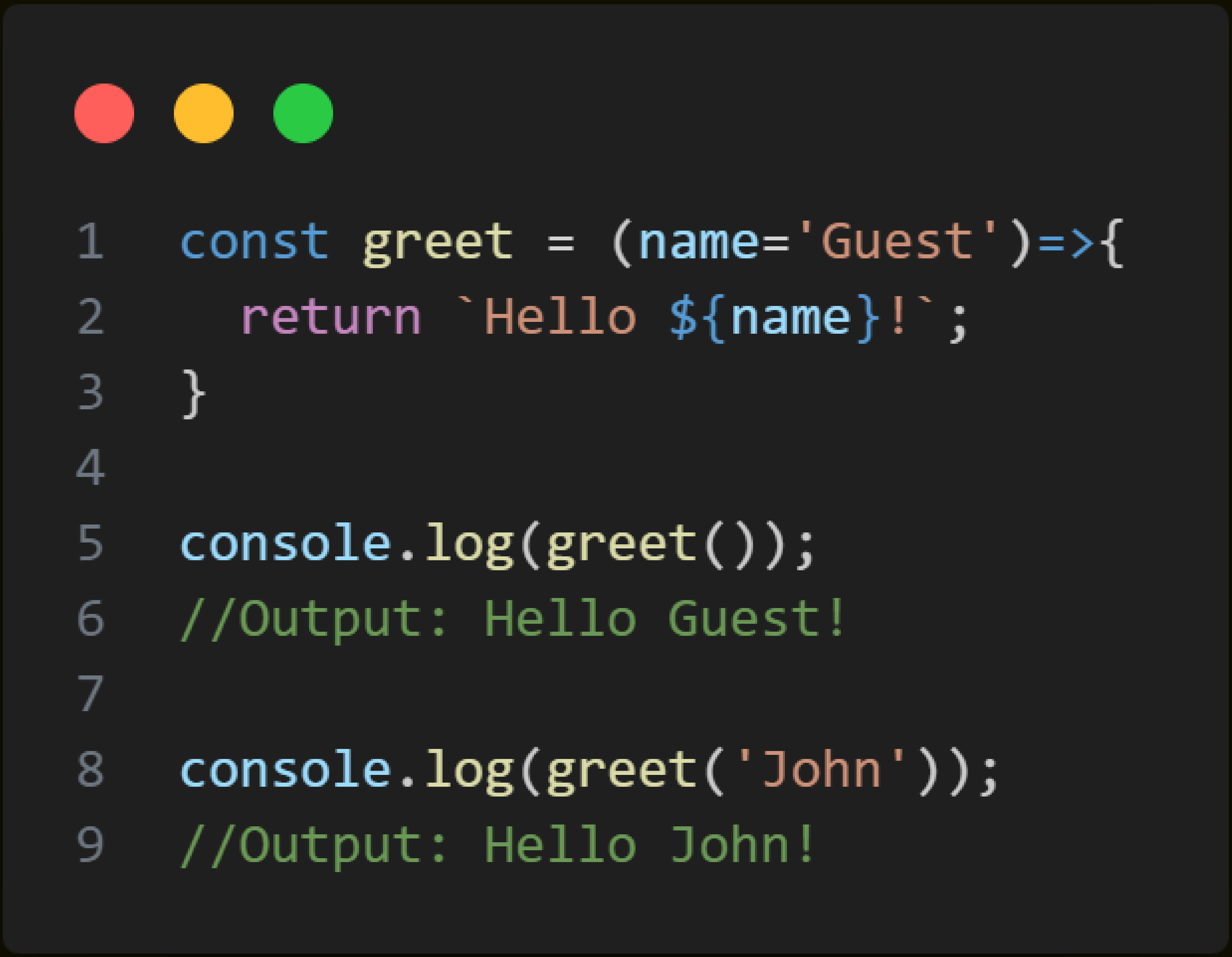
```
//Creating a Map with a key-value pair
const numberMap = new Map().set("one", 1);

//Creating a Set with unique numbers
const unique = new Set([1, 2, 3, 2, 1]);

unique.forEach((number) => console.log(number));

//Output: 1
//        2
//        3
```

Explanation: **Map** and **Set** are new data structures introduced in **ES6**
Map is an **ordered** collection of **key-value** pairs,
and **Set** is a collection of **unique** values.

save
post

swipe
to know

Default Parameters



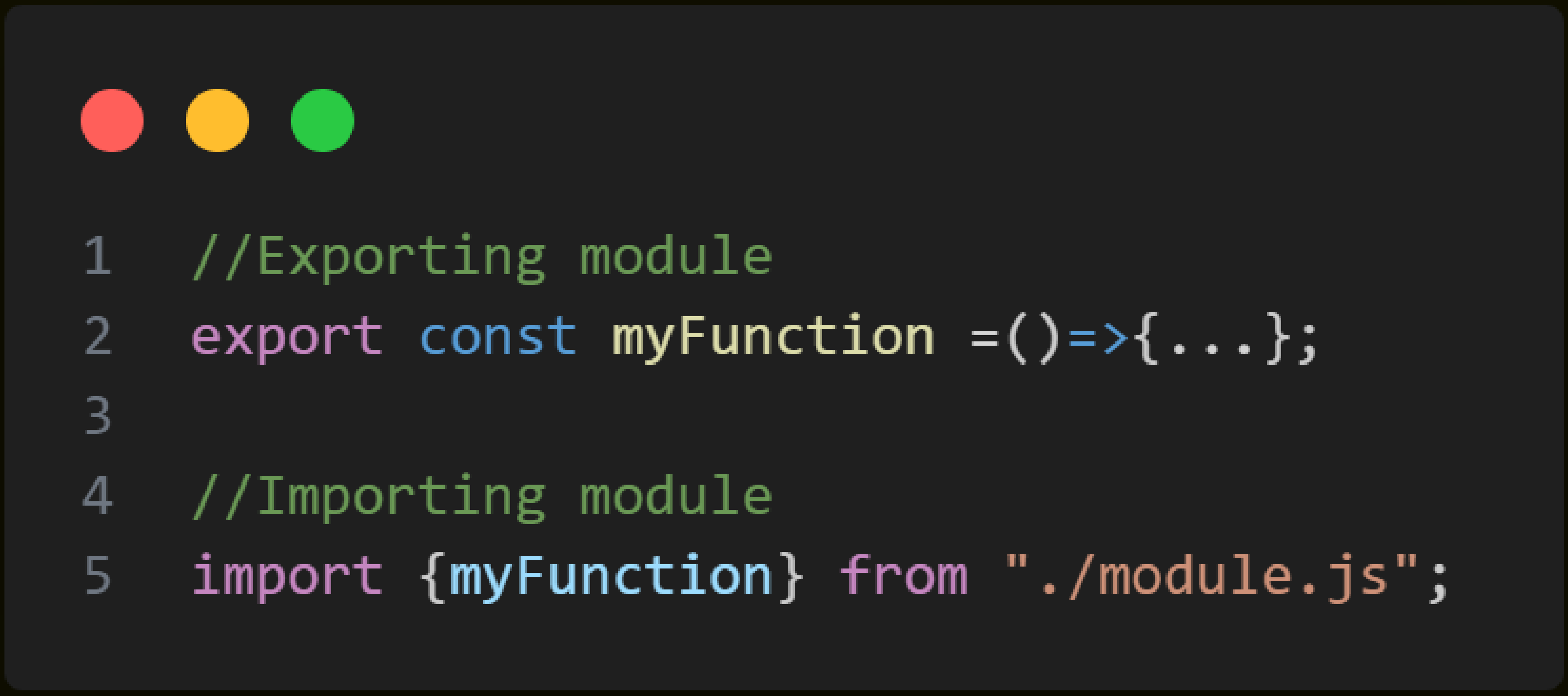
```javascript
const greet = (name='Guest')=>{
  return `Hello ${name}!`;
}

console.log(greet());
//Output: Hello Guest!

console.log(greet('John'));
//Output: Hello John!
```

Explanation: **Default parameters** provide values for function parameters if **none** are **provided**, improving **flexibility** and **reducing** the need for explicit checks.

save
post

swipe
to know

Modules



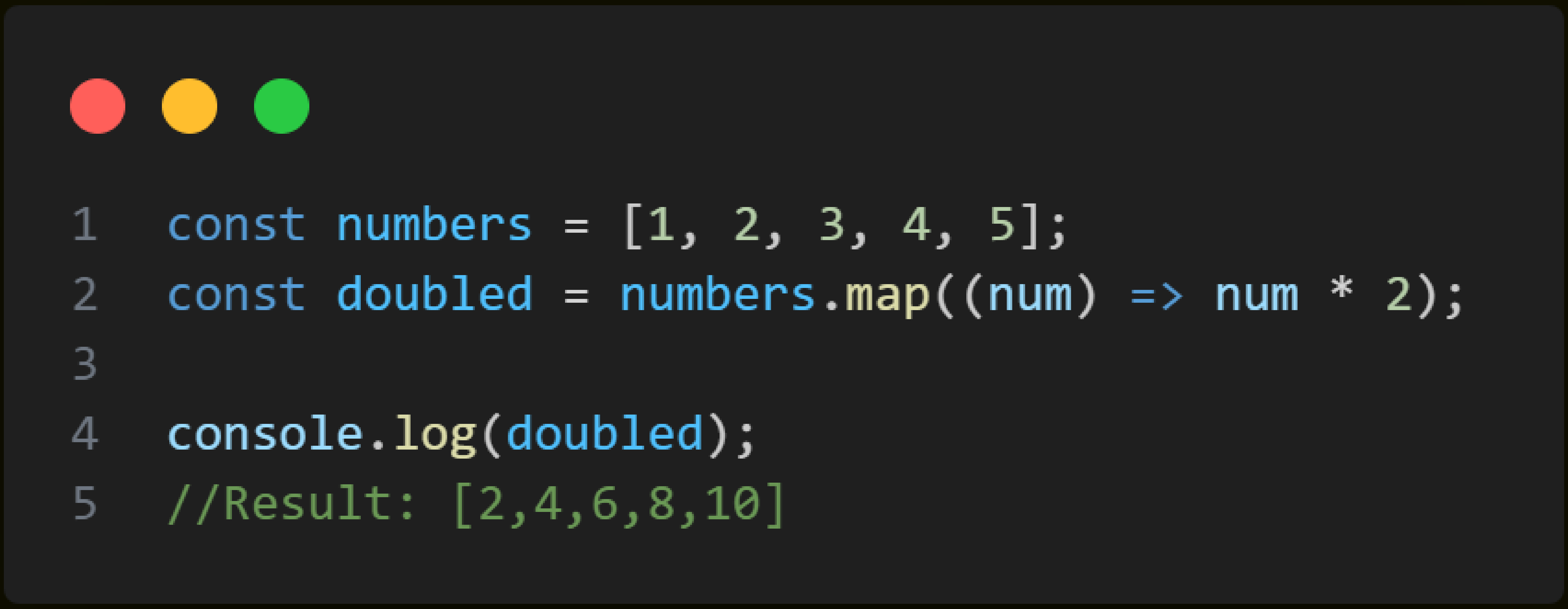
```
//Exporting module
export const myFunction =()=>{...};

//Importing module
import {myFunction} from "./module.js";
```

Explanation: ES6 modules provide a **clean** and **organized** way to structure and **import/export** code, improving **maintainability** and **reusablility**

save
post

swipe
to know

Map Method



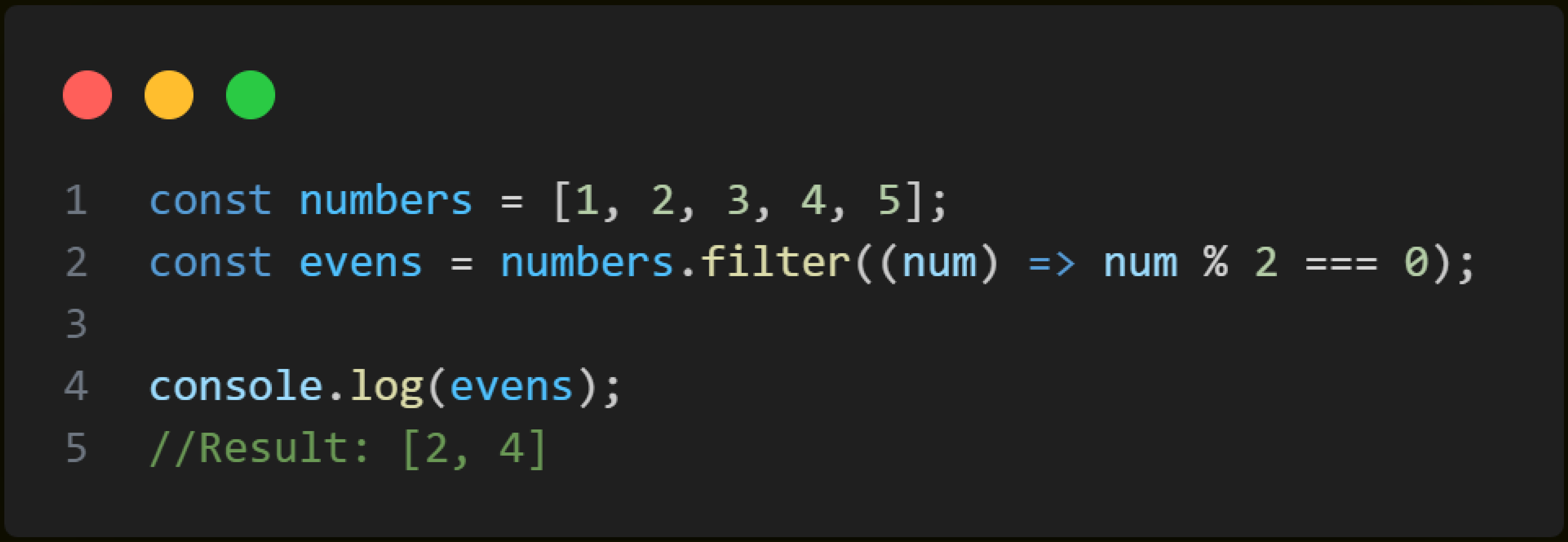
```javascript
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map((num) => num * 2);

console.log(doubled);
//Result: [2,4,6,8,10]
```

Explanation: The **map** method in JavaScript is used to create a **new array** by applying a provided function to **each** element of an **existing** array

save post

swipe to know

Filter Method



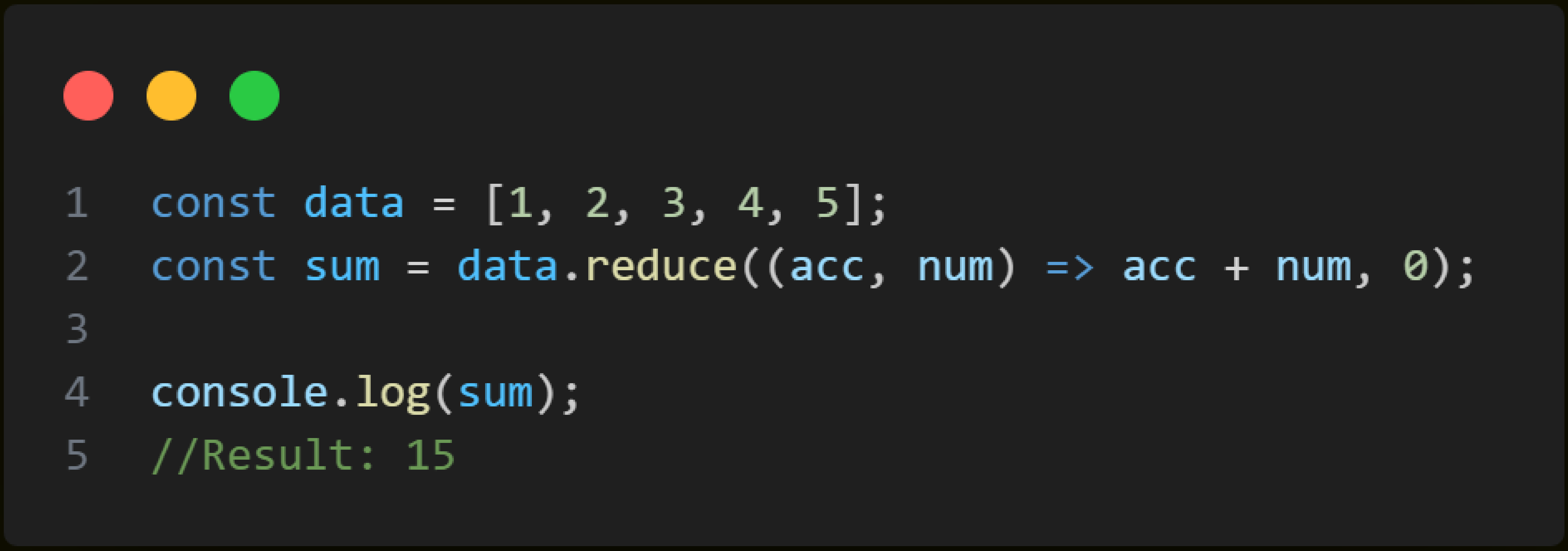
```javascript
const numbers = [1, 2, 3, 4, 5];
const evens = numbers.filter((num) => num % 2 === 0);

console.log(evens);
//Result: [2, 4]
```

Explanation: the **filter** method is used to create a **new array** containing only the elements that satisfy a **specified** condition.

save post

swipe to know

Reduce Method



```javascript
const data = [1, 2, 3, 4, 5];
const sum = data.reduce((acc, num) => acc + num, 0);

console.log(sum);
//Result: 15
```

Explanation: The **Reduce** method is used to **accumulate** the elements of an array into **single** value

save post

swipe to know