JavaScript Tips & Tricks

Closures

Master the art of closures to writemore elegant, scalable, and maintainable JavaScript code.

01

What are closures?

A closure is a function that has access to the variables in the scope in which it was created, even after that scope has closed. This means that a closure can access variables that are no longer available to other functions

02

Benefits of using closures

- Closures can help you to create private variables and functions.
- Closures can help you to create functions that are more reusable and flexible.
- Closures can help you to write more efficient and performant code.

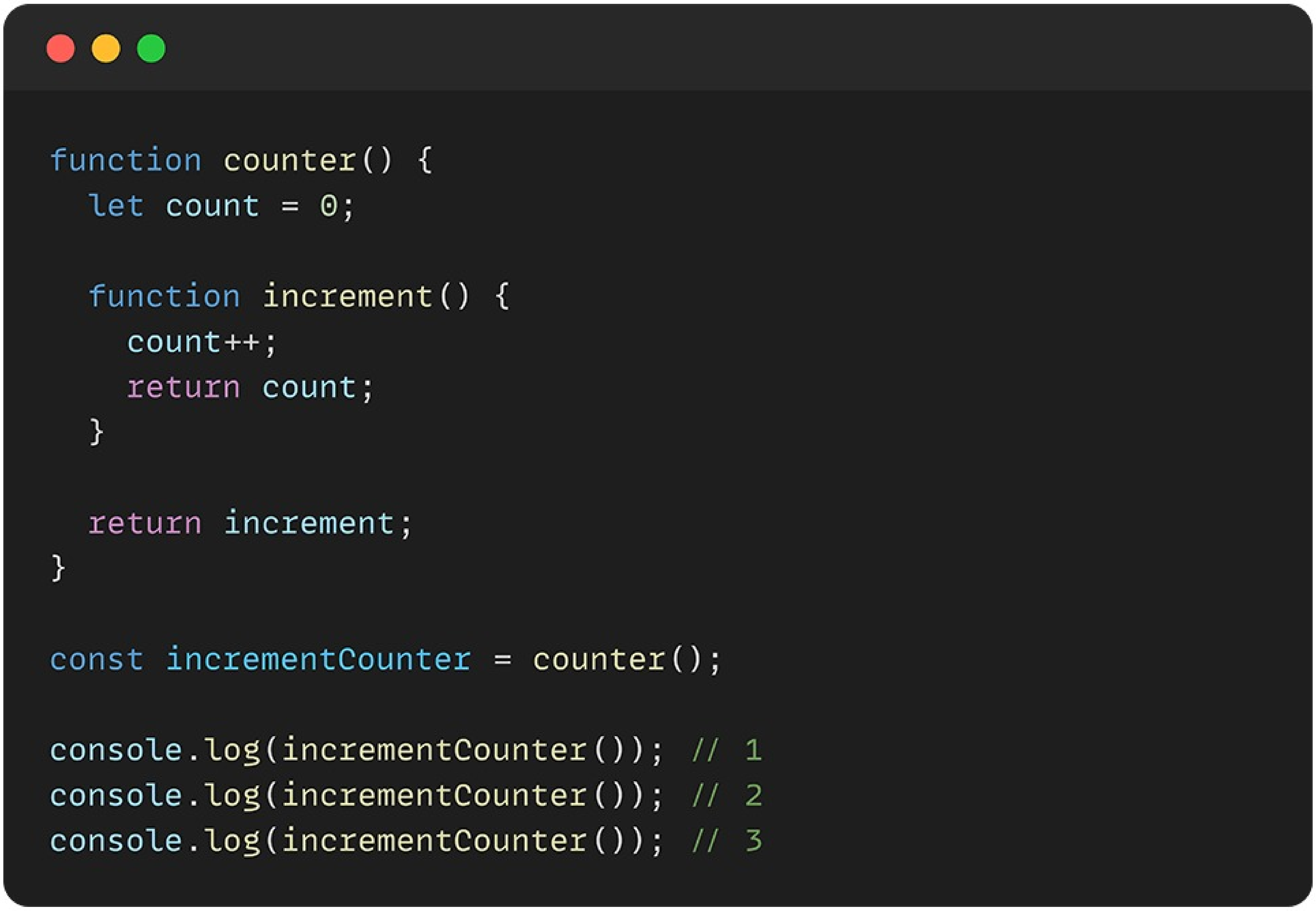
03

How to use closures

To use a closure, simply create a function inside of another function. The inner function will then have access to the variables in the outer function's scope, even after the outer function has returned.

04

Demo code



```
function counter() {
  let count = 0;

  function increment() {
    count++;
    return count;
  }

  return increment;
}

const incrementCounter = counter();

console.log(incrementCounter()); // 1
console.log(incrementCounter()); // 2
console.log(incrementCounter()); // 3
```

JS

05

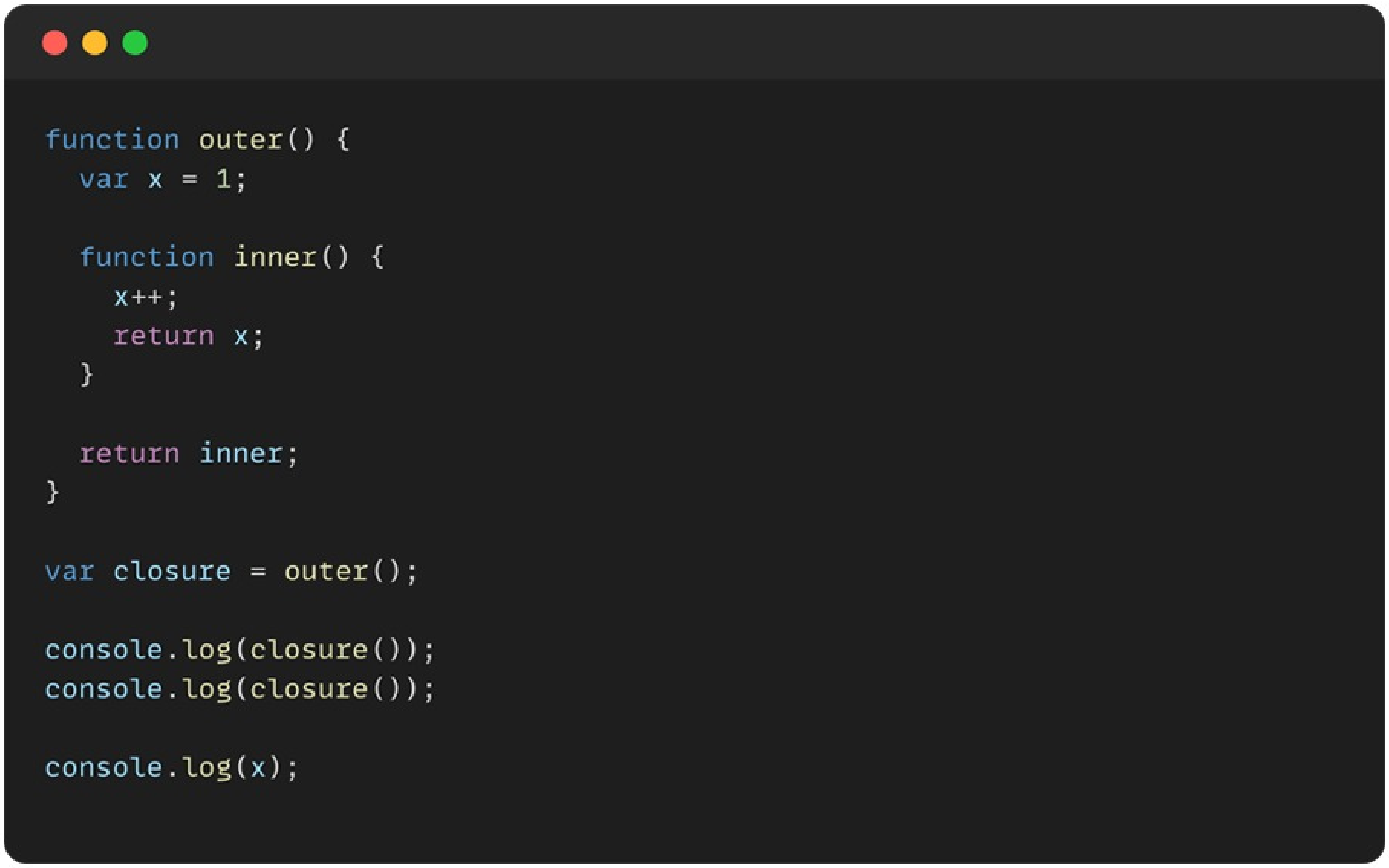
Conclusion

Closures are the secret sauce to writing elegant, flexible, and performant JavaScript code. They allow you to create private variables and functions, reusable functions, and efficient code.

06

Quiz Time!

What is the output of the last line of code? Write your answer in the comments



```
function outer() {
  var x = 1;

  function inner() {
    x++;
    return x;
  }

  return inner;
}

var closure = outer();

console.log(closure());
console.log(closure());

console.log(x);
```