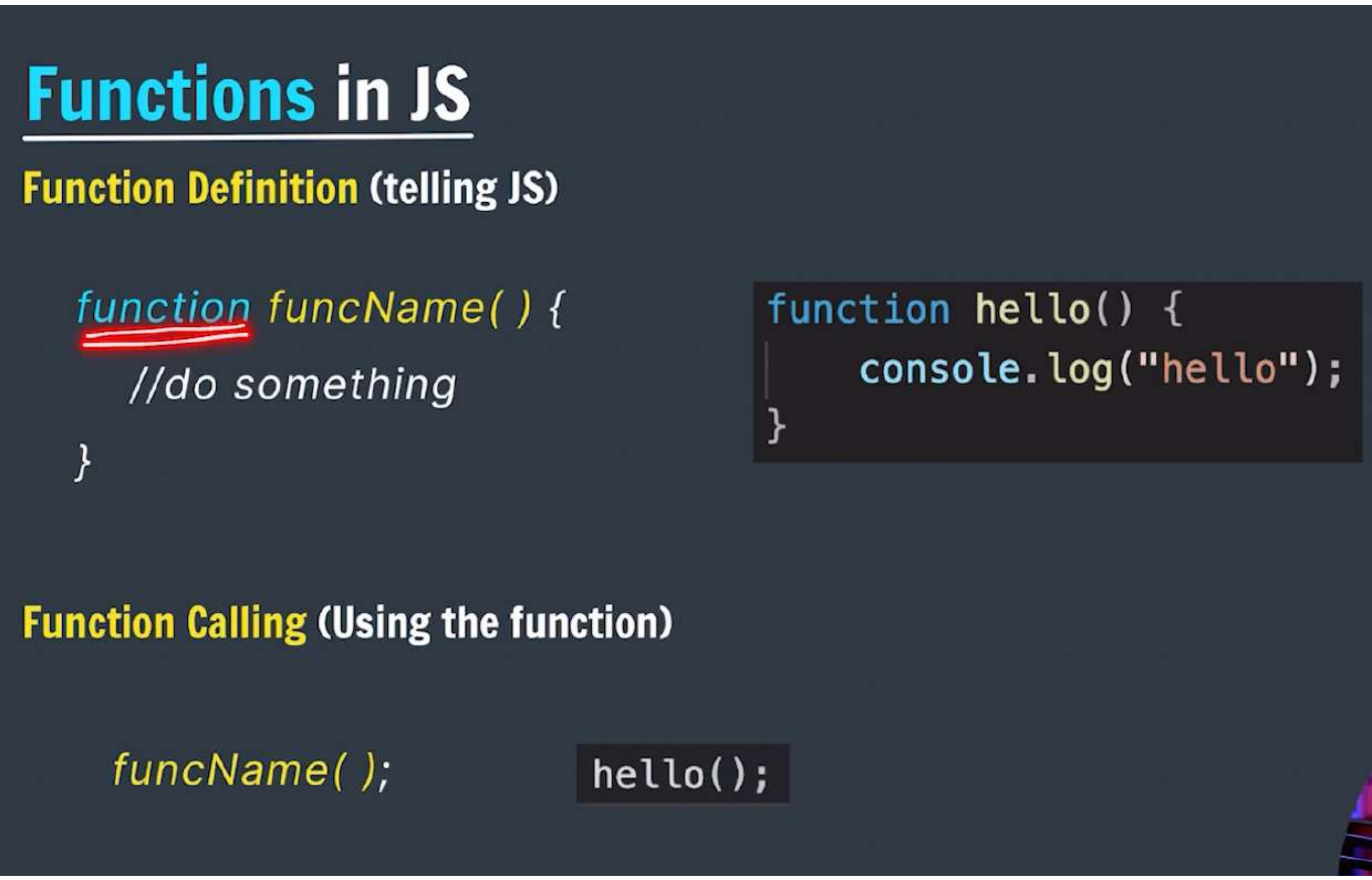


Functions in JS

Function Definition (telling JS)

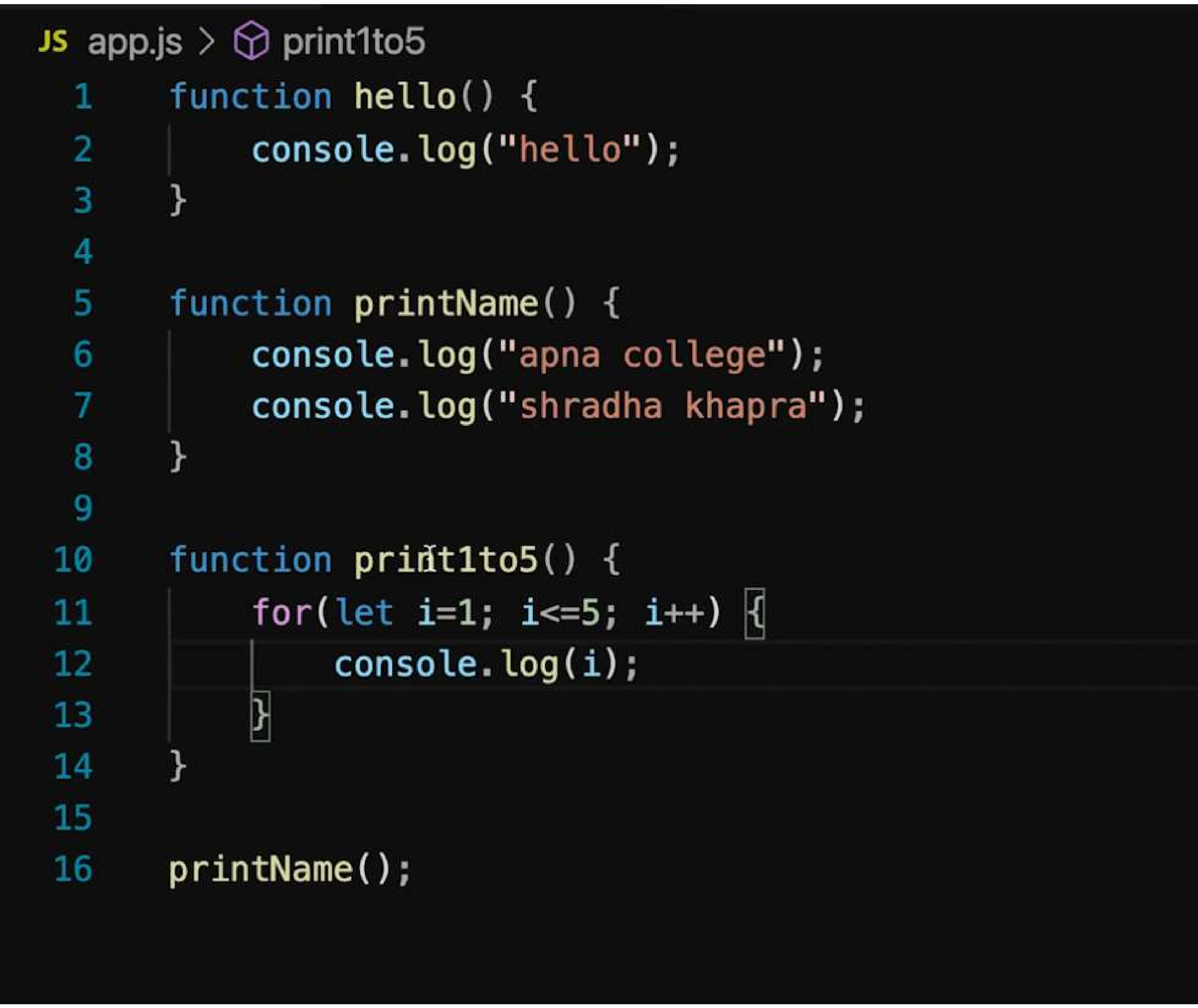
```
function funcName( ) {
    //do something
}
```

```
function hello() {
    console.log("hello");
}
```

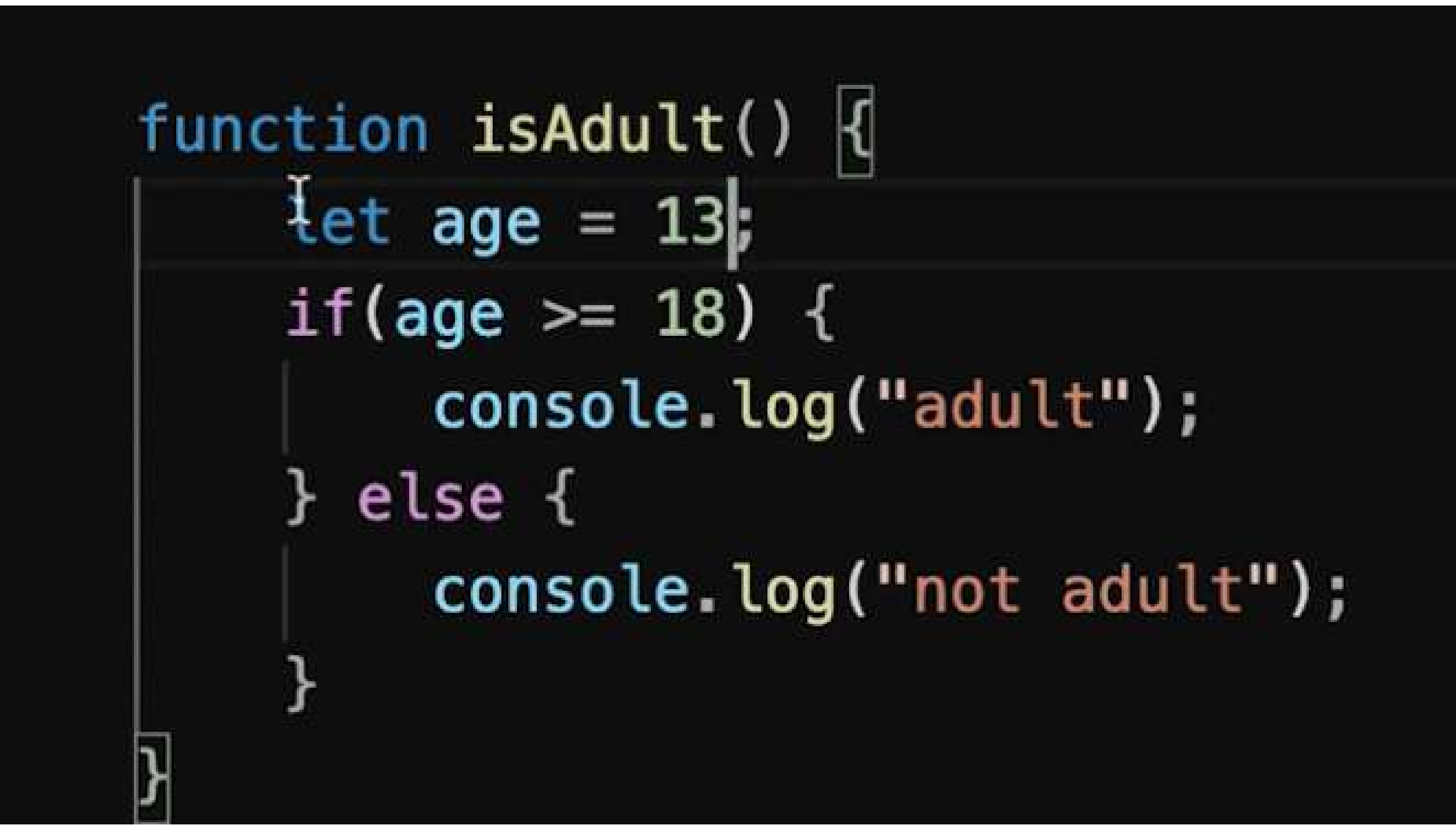
Function Calling (Using the function)

```
funcName( );
```

```
hello();
```



```
function hello() {
    console.log("hello");
}

function printName() {
    console.log("apna college");
    console.log("shradha khapra");
}

function print1to5() {
    for(let i=1; i<=5; i++) {
        console.log(i);
    }
}

printName();
```



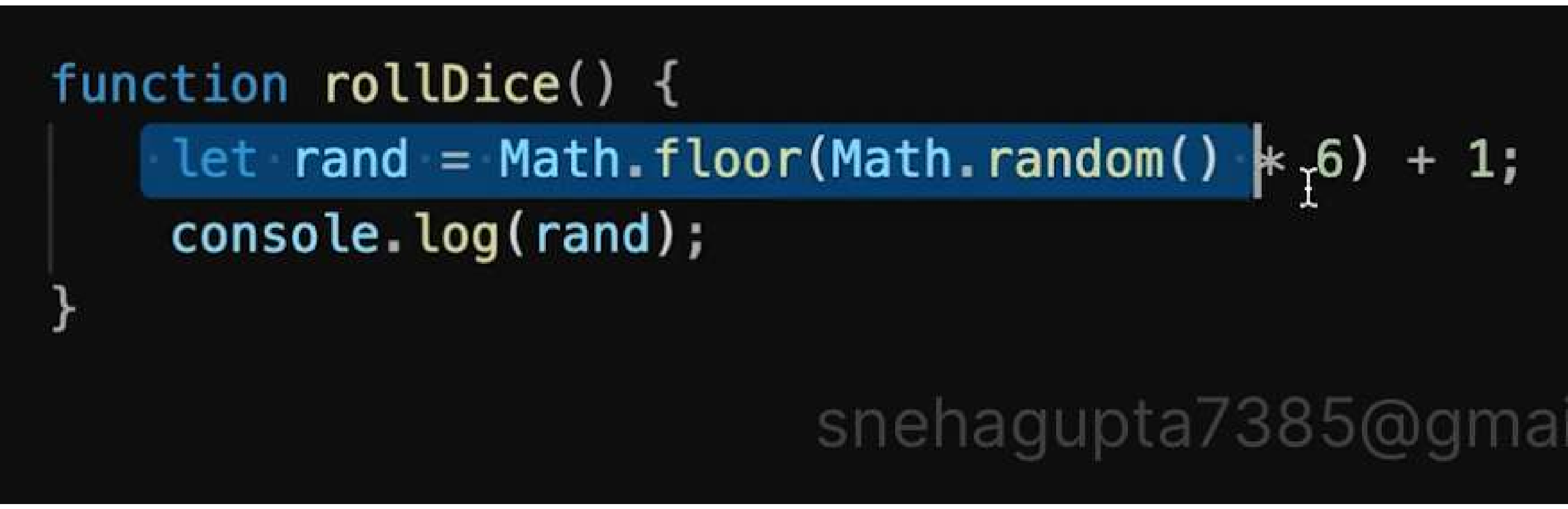
```
function isAdult() {
    let age = 13;
    if(age >= 18) {
        console.log("adult");
    } else {
        console.log("not adult");
    }
}
```

Practice Qs

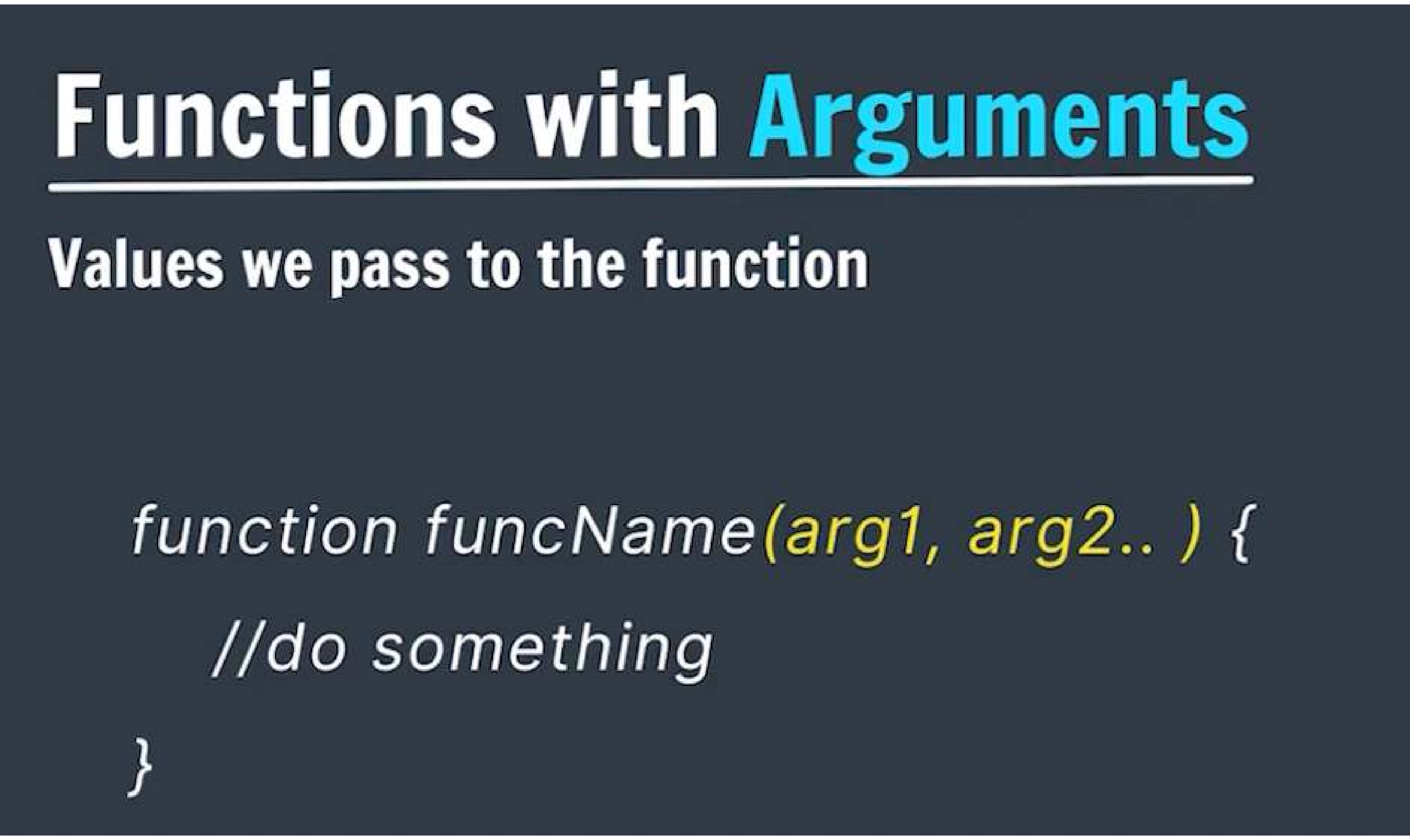
Create a function that prints a poem.

Practice Qs

Create a Function to roll a dice & always display the value of the dice (1 to 6).



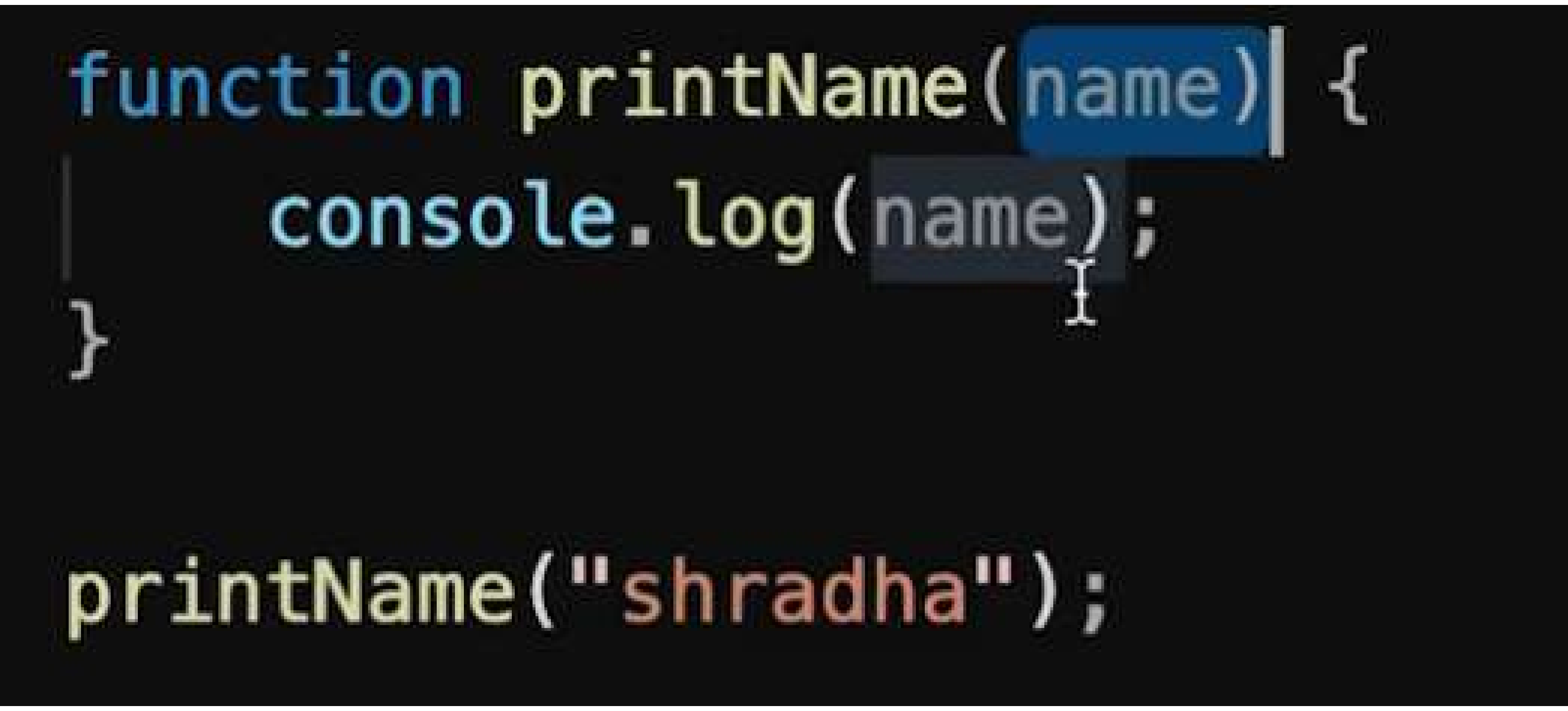
```
function rollDice() {
    let rand = Math.floor(Math.random() * 6) + 1;
    console.log(rand);
}
```



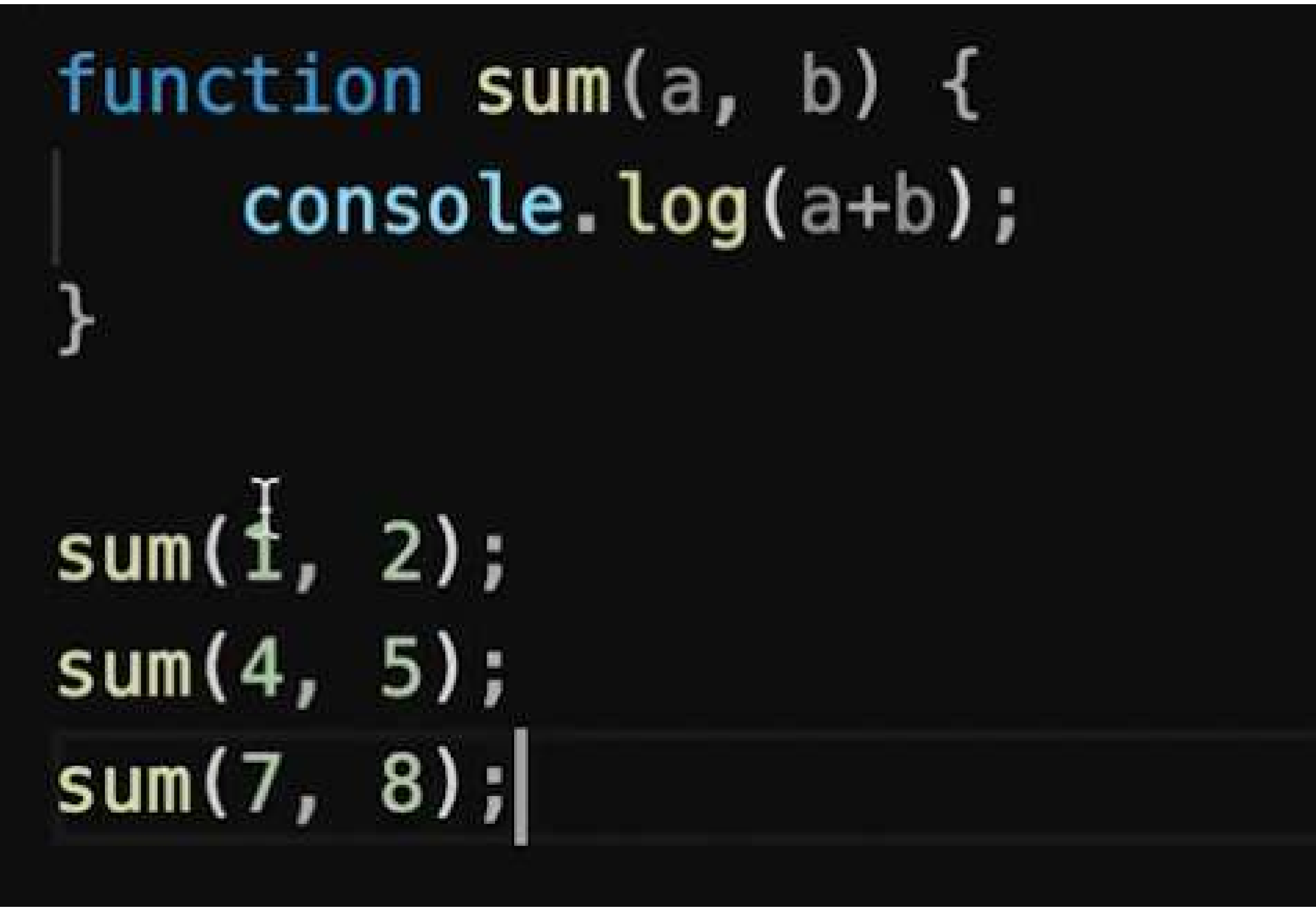
Functions with Arguments

Values we pass to the function

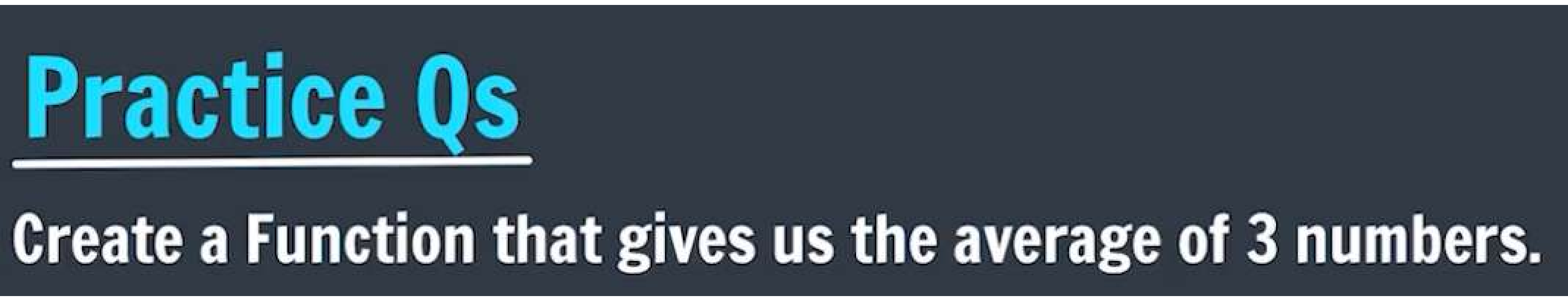
```
function funcName(arg1, arg2.. ) {
    //do something
}
```



```
function printName(name) {
    console.log(name);
}

printName("shradha");
```

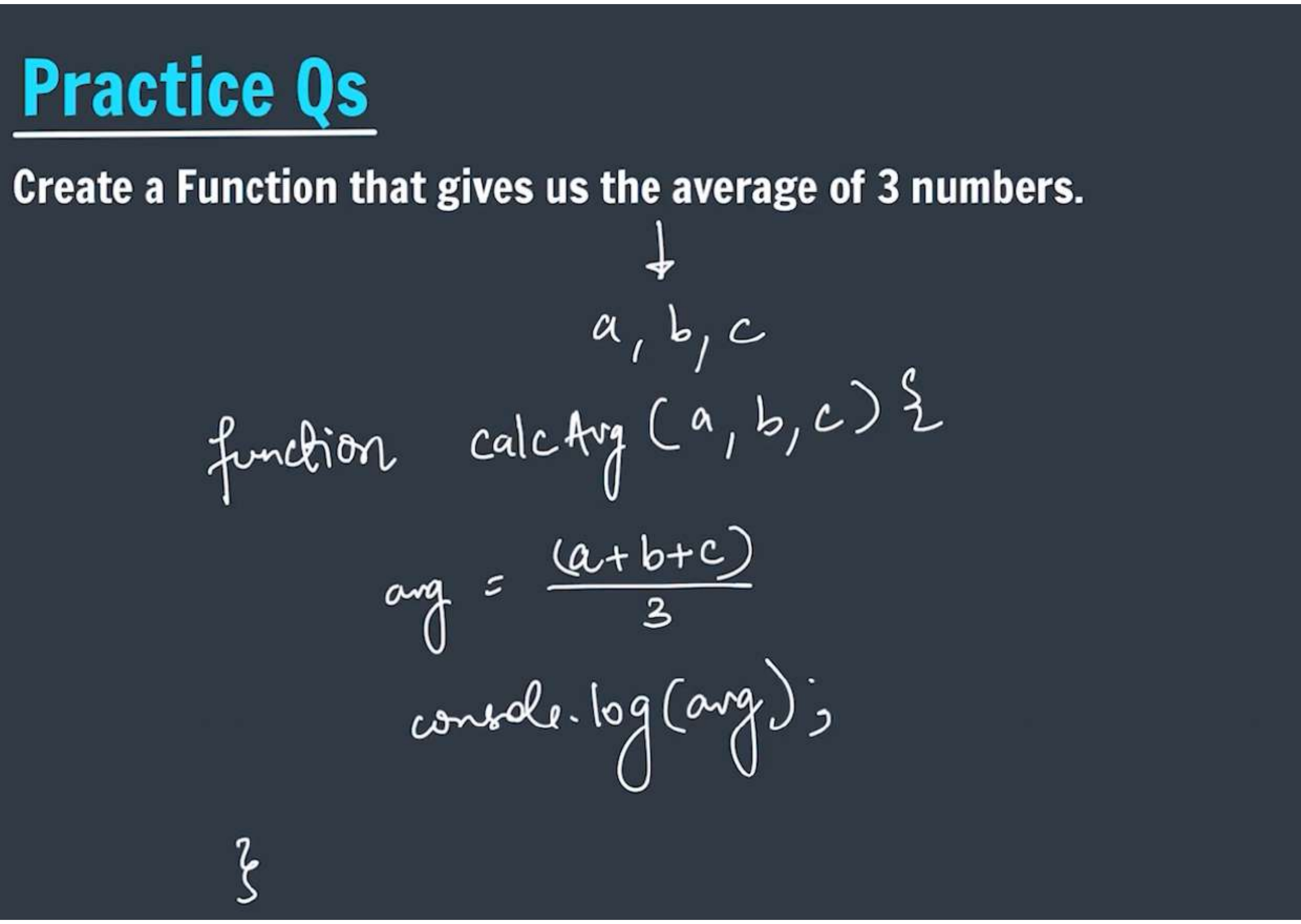


```
function sum(a, b) {
    console.log(a+b);
}


sum(1, 2);
sum(4, 5);
sum(7, 8);
```



Practice Qs

Create a Function that gives us the average of 3 numbers.



Practice Qs

Create a Function that gives us the average of 3 numbers.

$\downarrow$

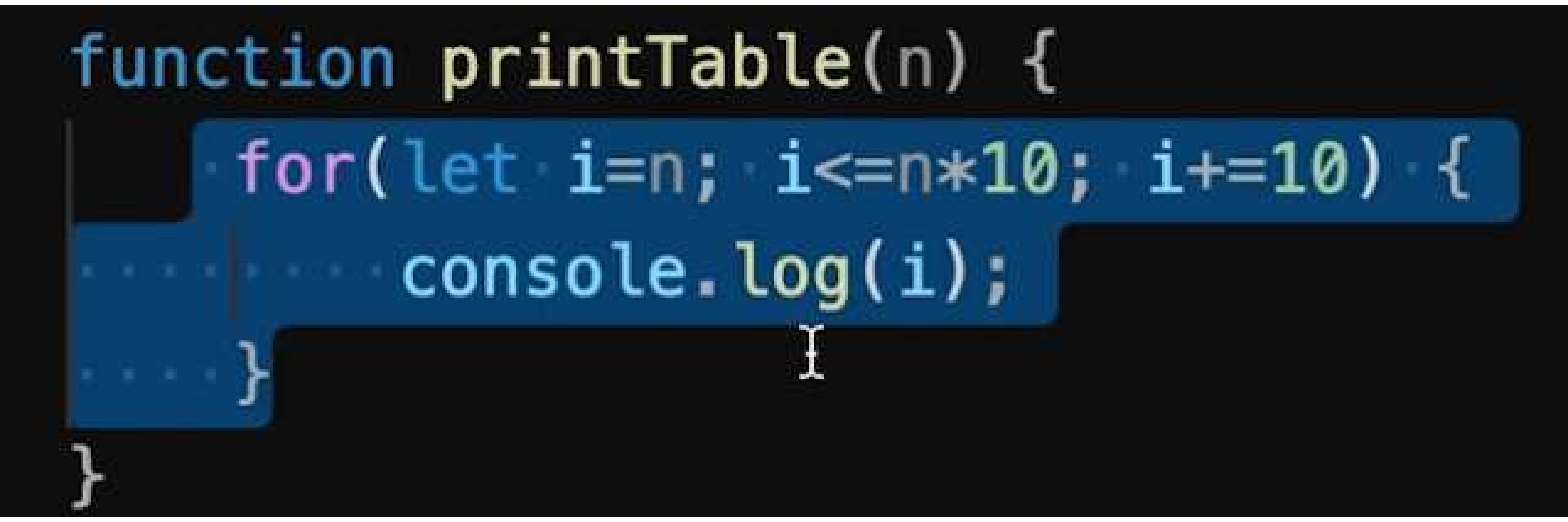
a, b, c

```
function calcAvg (a, b, c) {
        avg = (a+b+c) / 3
        console.log(avg);
}
```

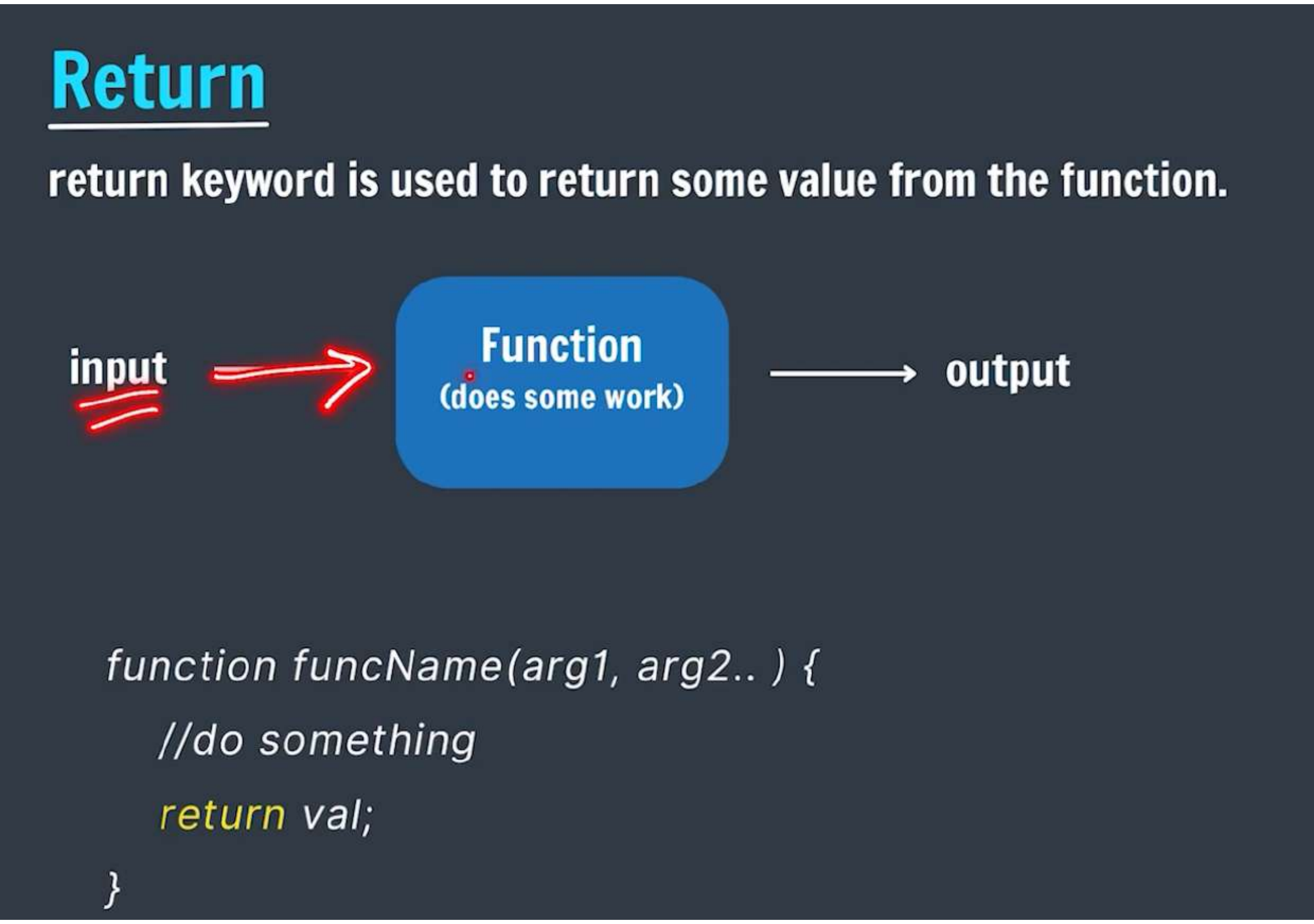
$$avg = \frac{(a+b+c)}{3}$$

Practice Qs

Create a Function that prints the multiplication table of a number.

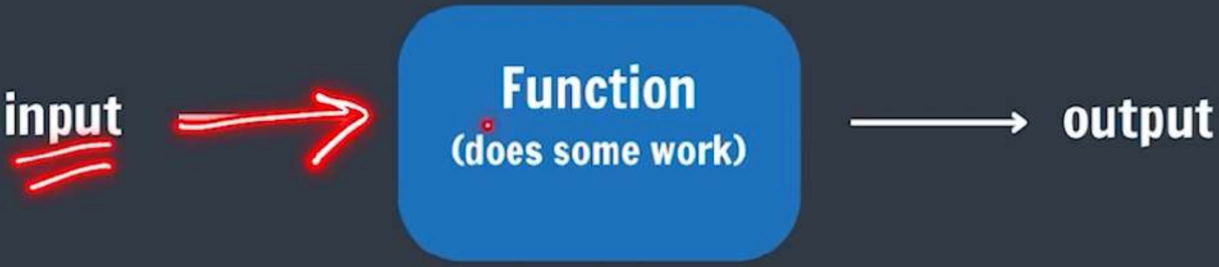


```
function printTable(n) {
    for(let i=n; i<=n*10; i+=10) {
        console.log(i);
    }
}
```

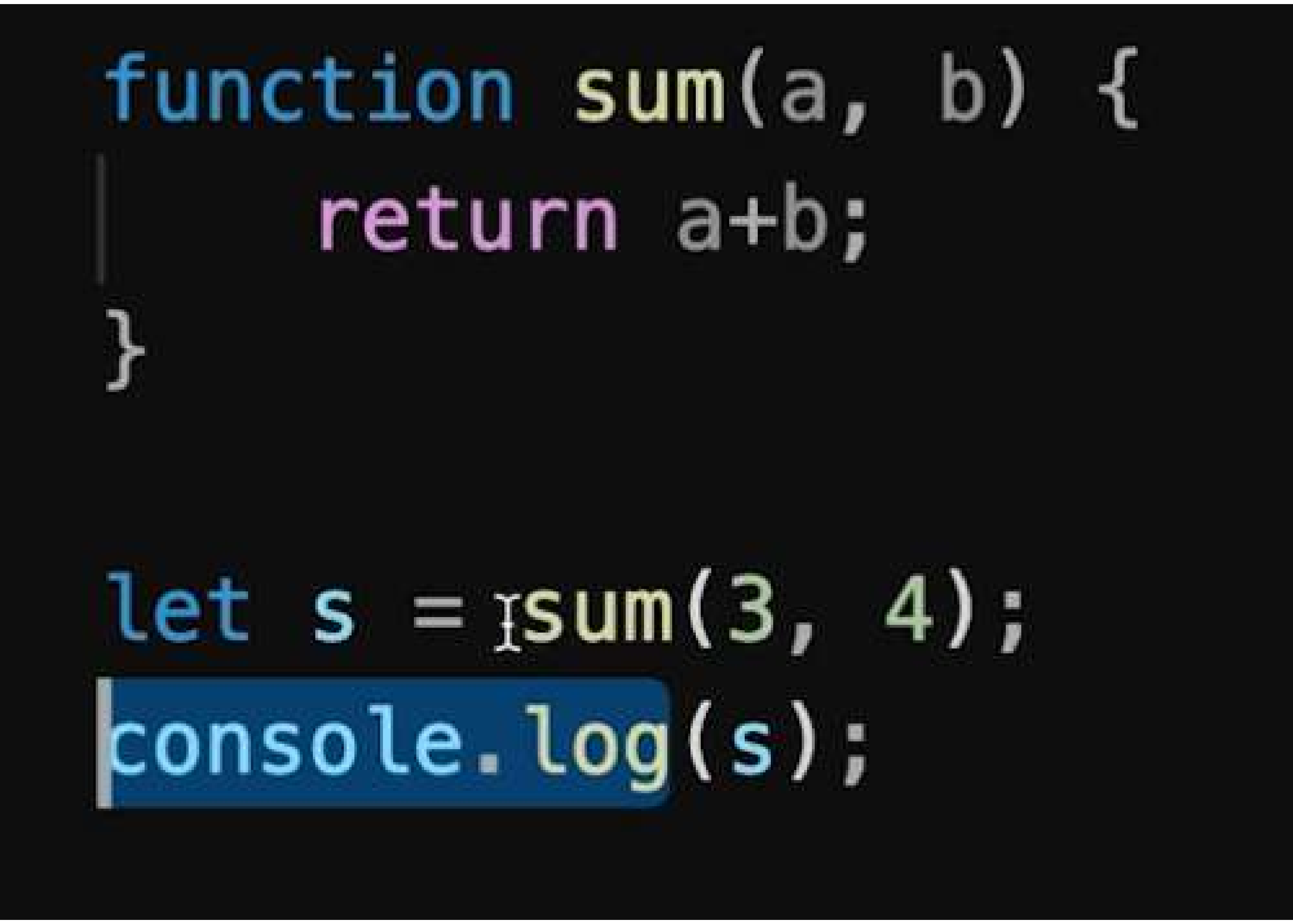


Return

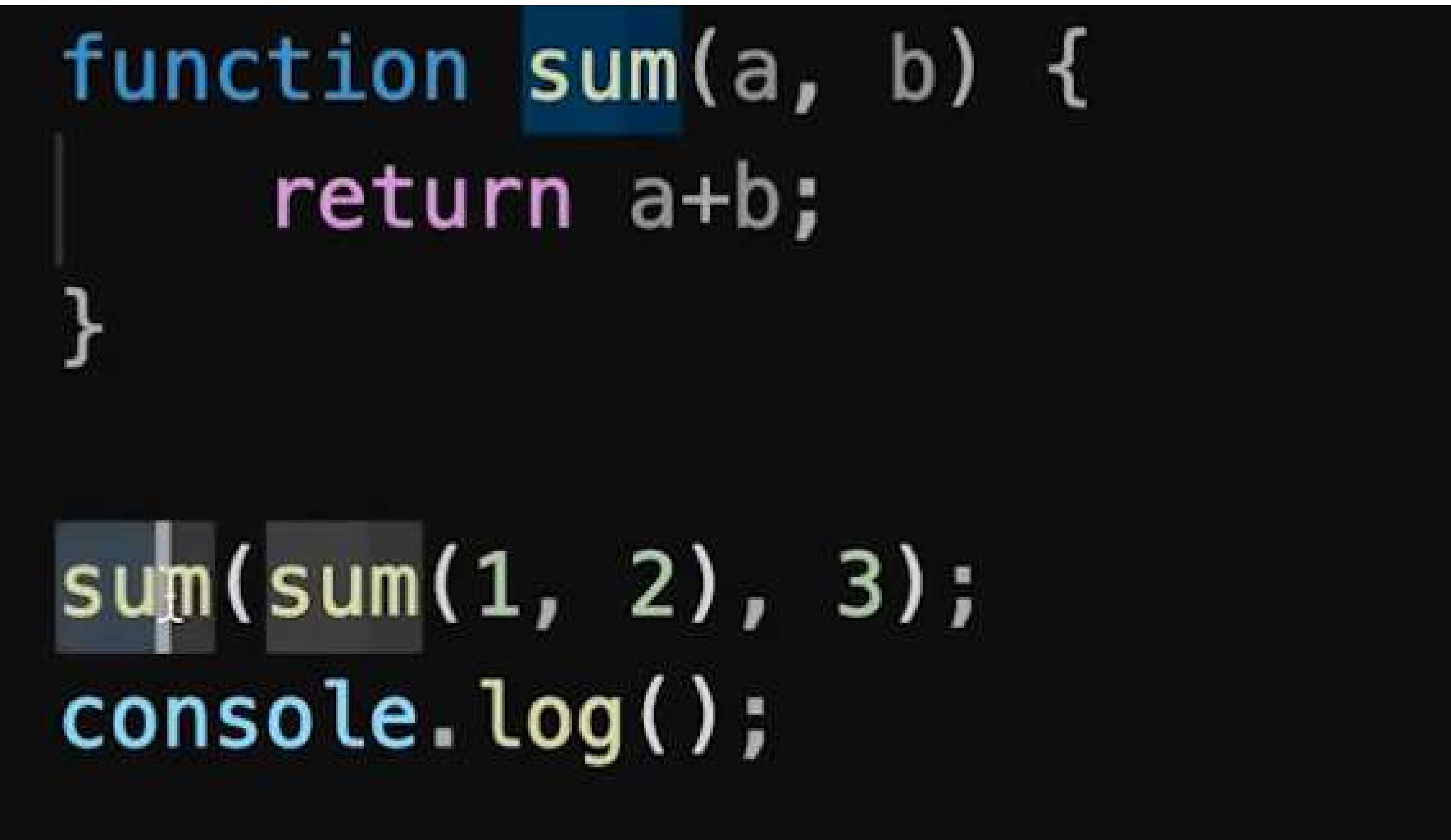
return keyword is used to return some value from the function.



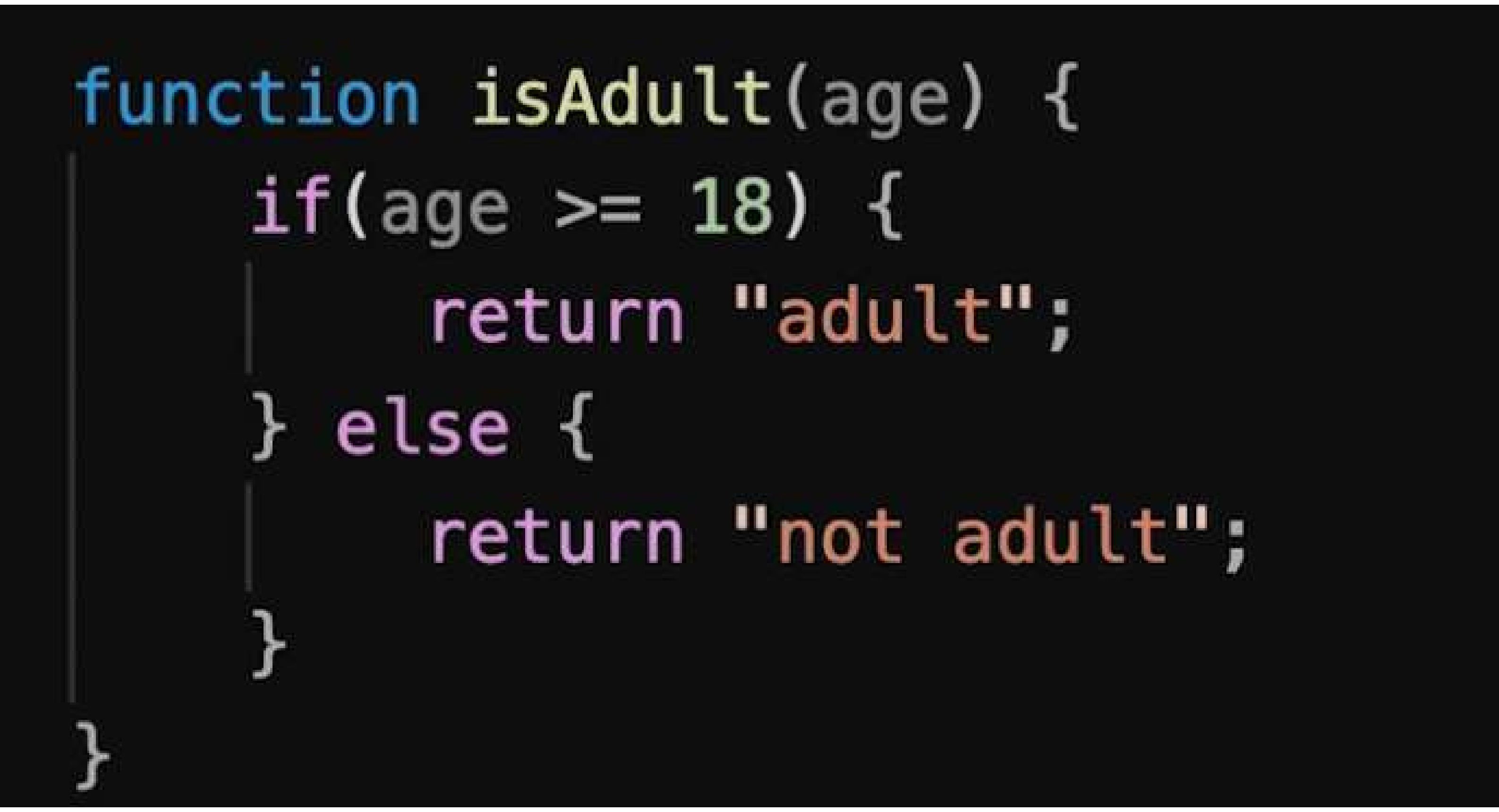
```
function funcName(arg1, arg2.. ) {
    //do something
    return val;
}
```



```
function sum(a, b) {
    return a+b;
}

let s = sum(3, 4);
console.log(s);
```



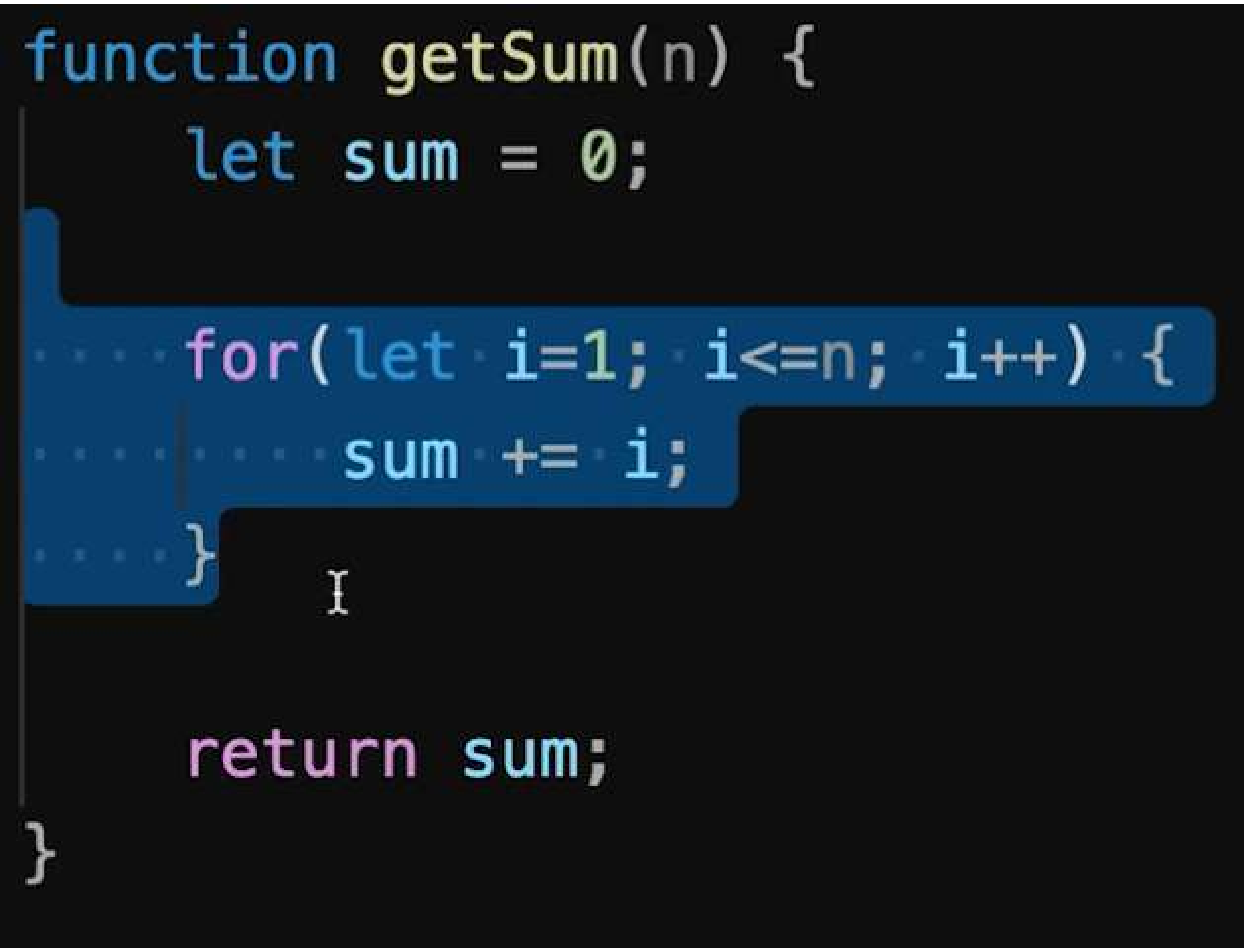
```
function sum(a, b) {
    return a+b;
}

sum(sum(1, 2), 3);
console.log();
```



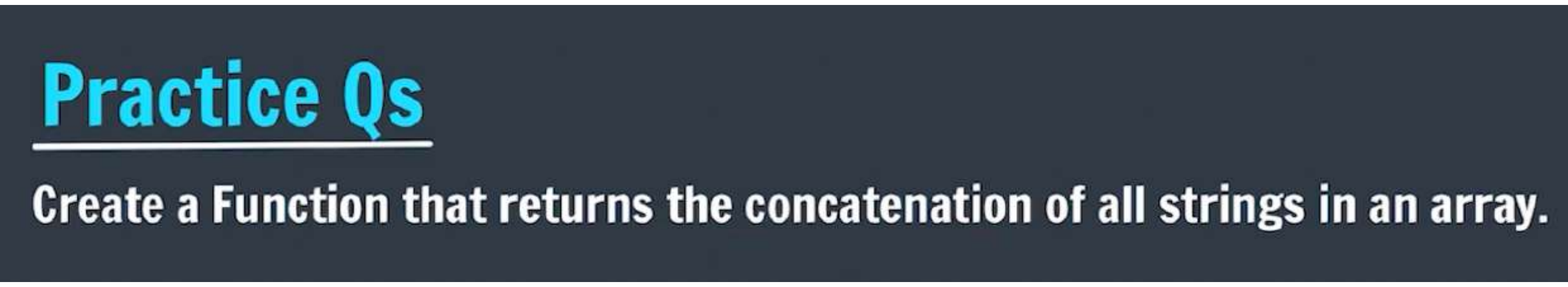
```
function isAdult(age) {
    if(age >= 18) {
        return "adult";
    } else {
        return "not adult";
    }
}
```

Practice Qs

Create a Function that returns the sum of numbers from 1 to n.

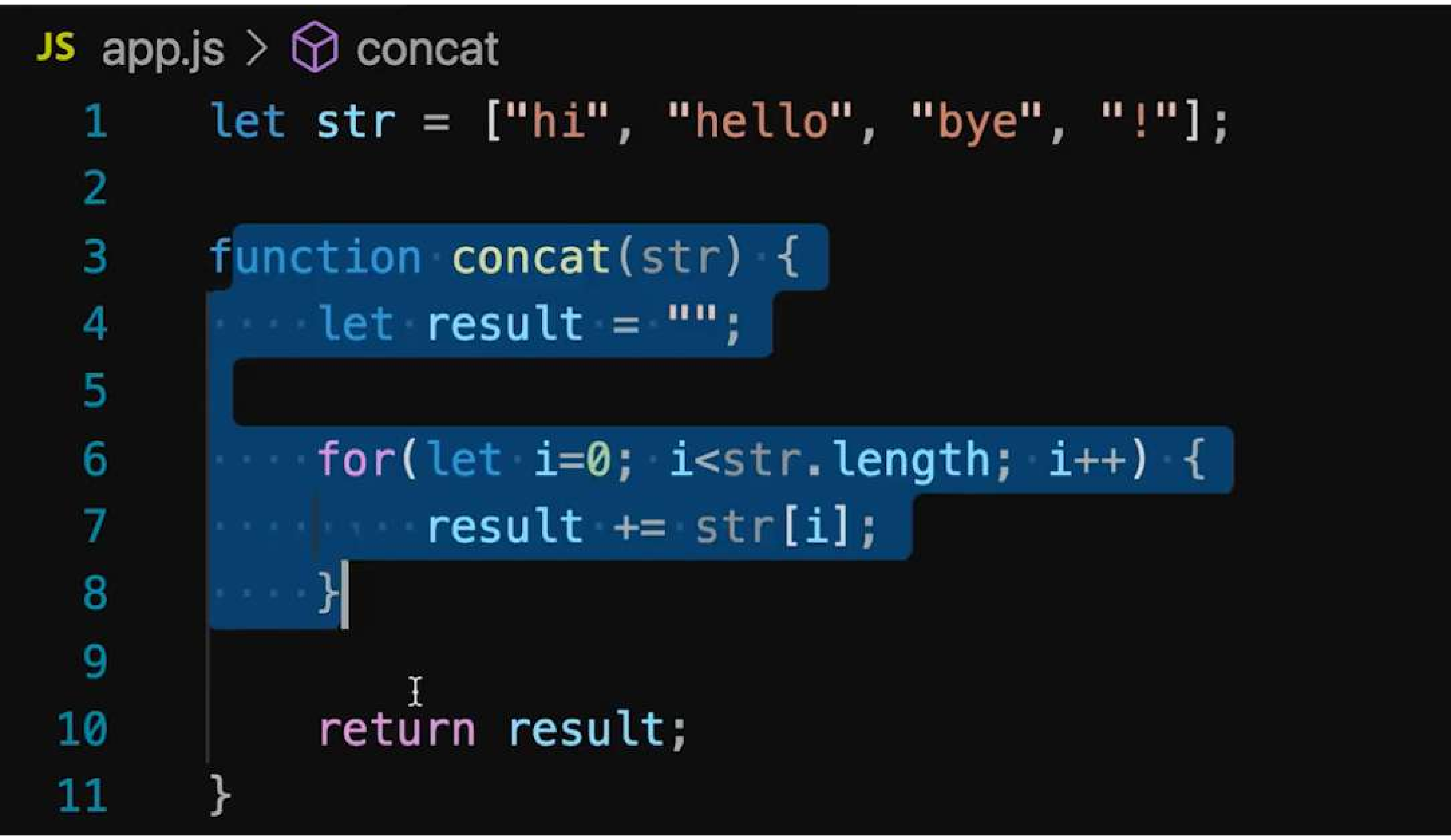


```
function getSum(n) {
    let sum = 0;

    for(let i=1; i<=n; i++) {
        sum += i;
    }

    return sum;
}
```

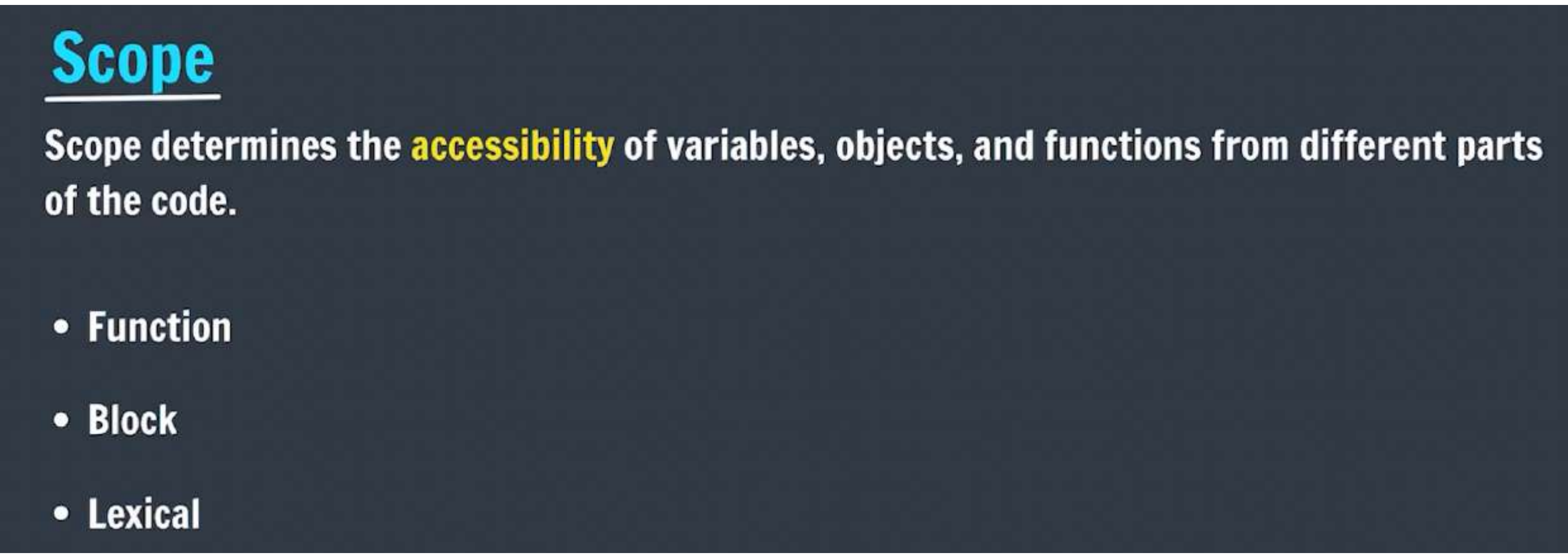


Practice Qs

Create a Function that returns the concatenation of all strings in an array.



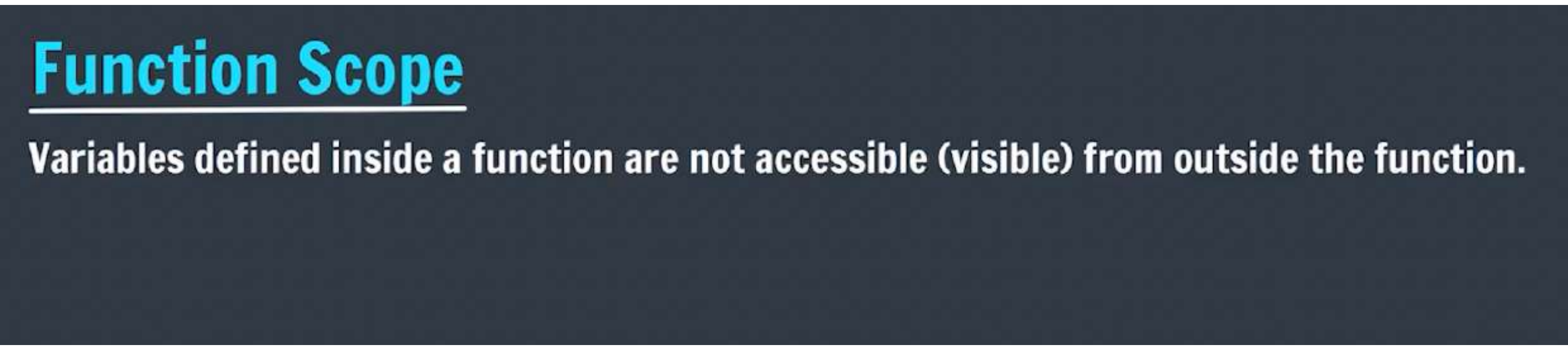
```javascript
let str = ["hi", "hello", "bye", "!"];

function concat(str) {
    let result = "";

    for(let i=0; i<str.length; i++) {
        result += str[i];
    }

    return result;
}
```



Scope

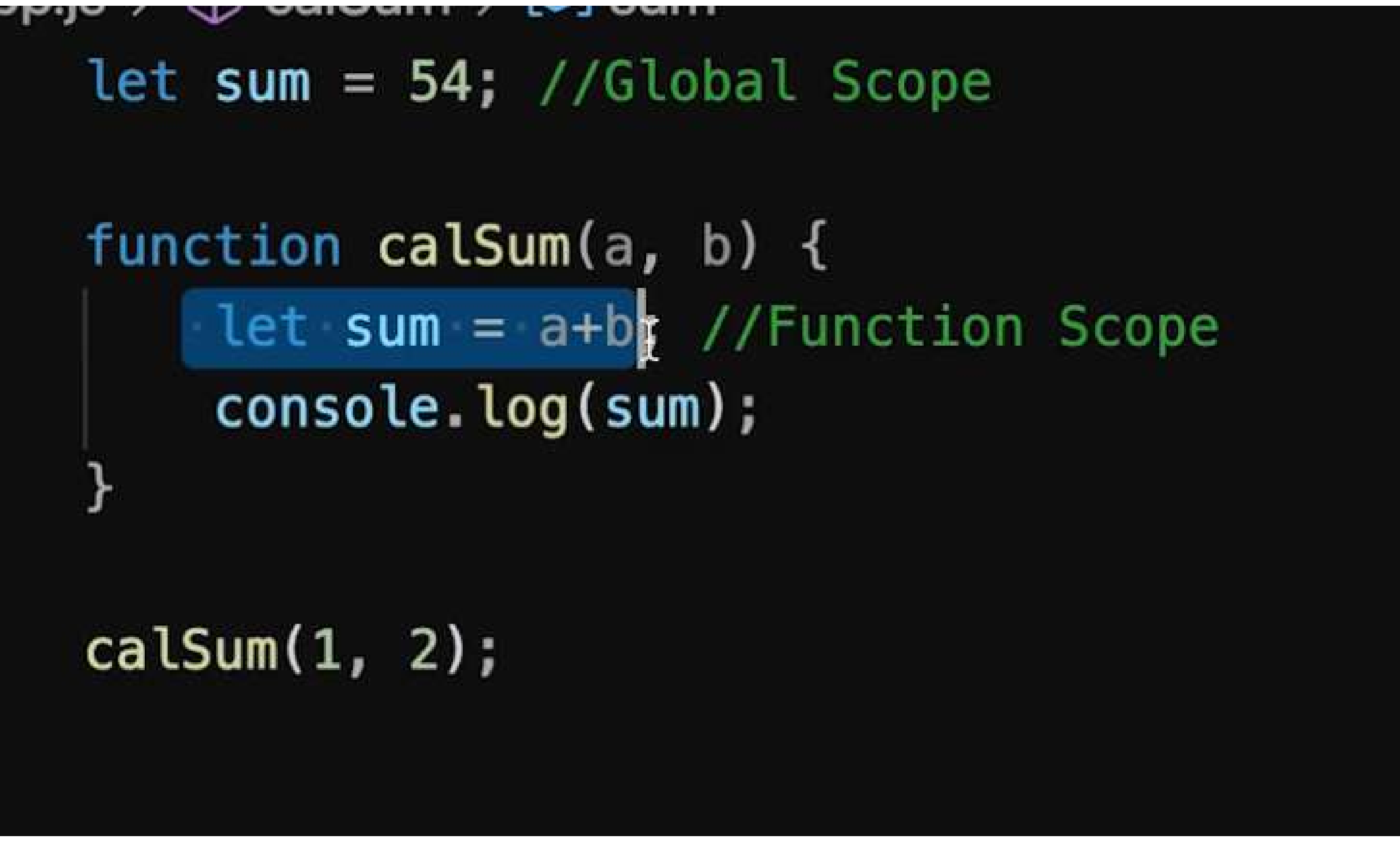
Scope determines the accessibility of variables, objects, and functions from different parts of the code.

- Function
- Block
- Lexical

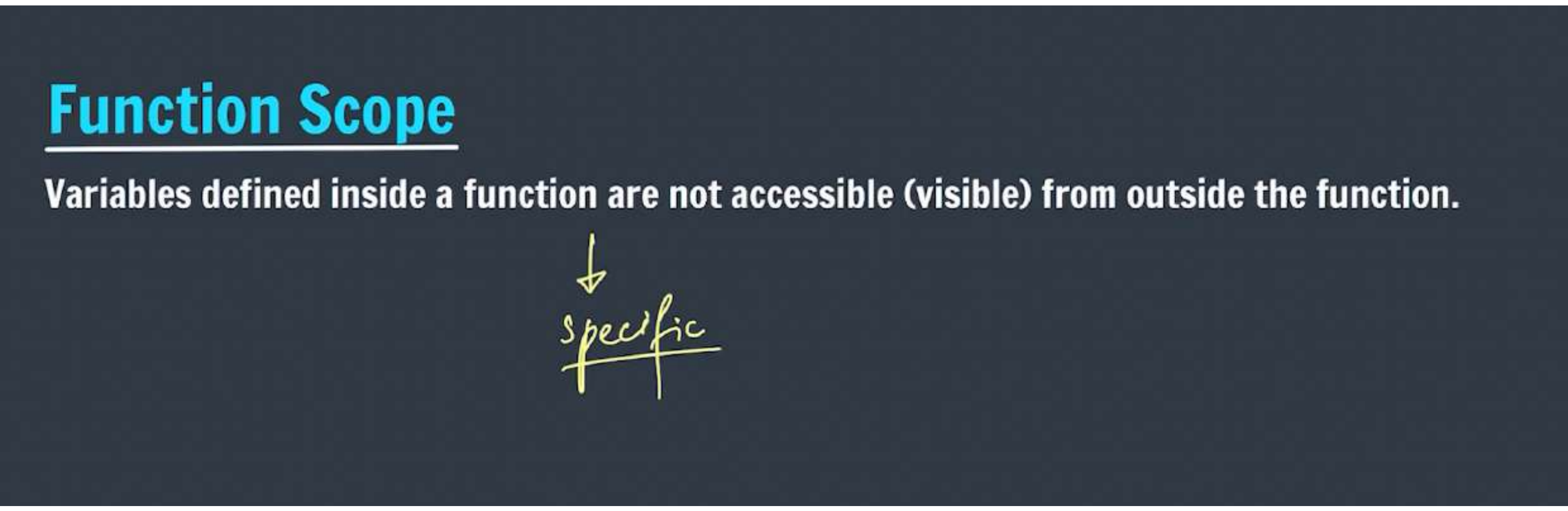


Function Scope

Variables defined inside a function are not accessible (visible) from outside the function.



```
let sum = 54; //Global Scope

function calSum(a, b) {
    let sum = a+b; //Function Scope
    console.log(sum);
}

calSum(1, 2);
```



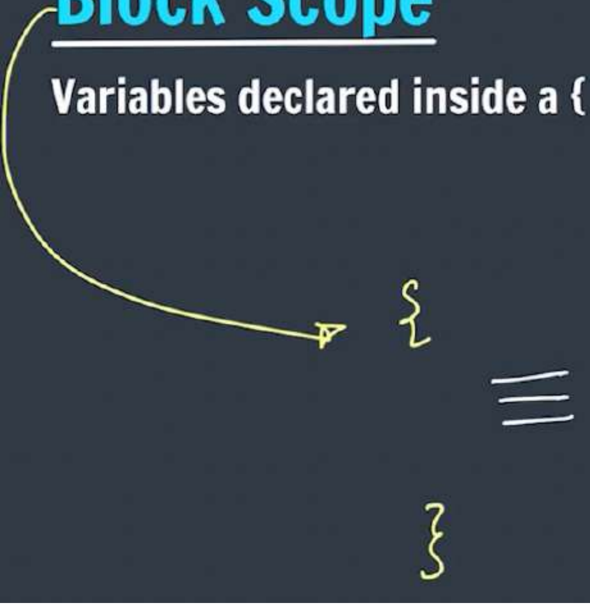
Function Scope

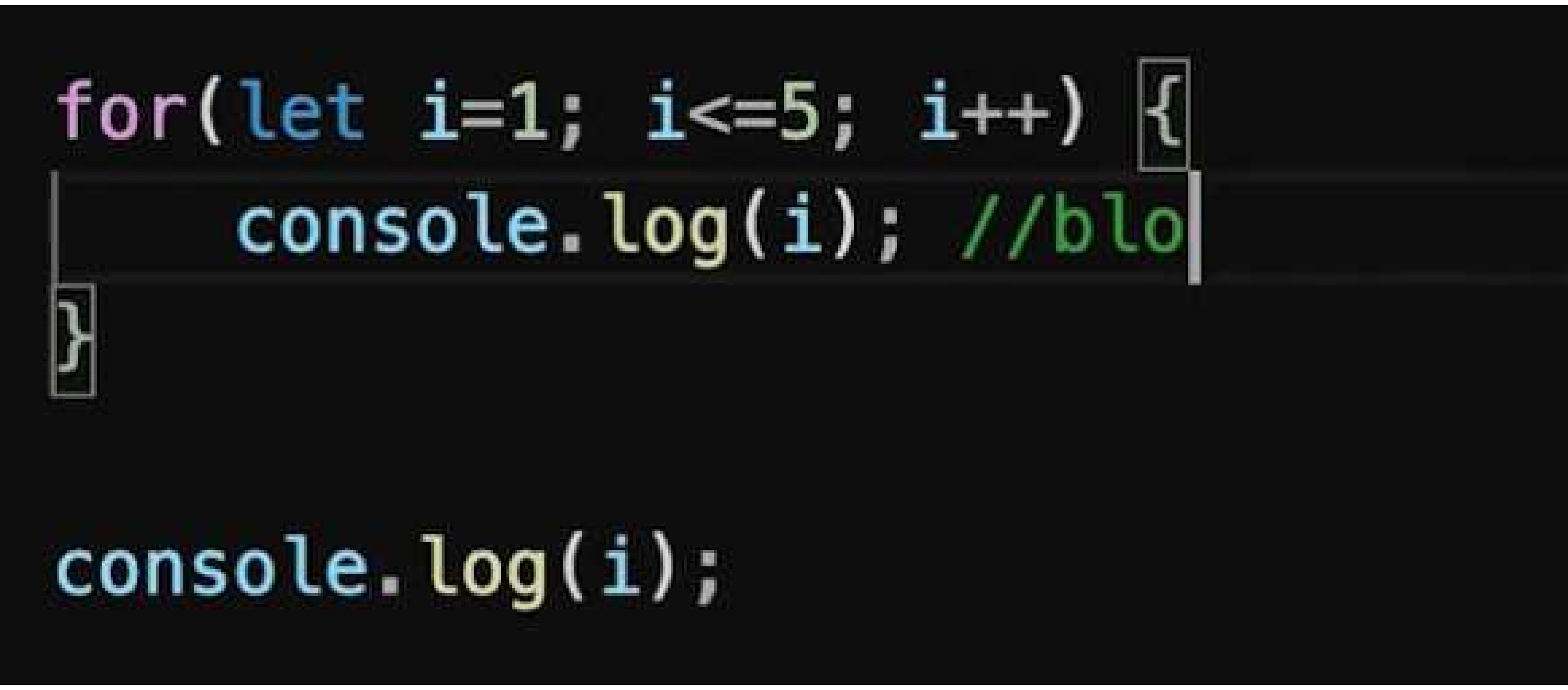
Variables defined inside a function are not accessible (visible) from outside the function.

specific

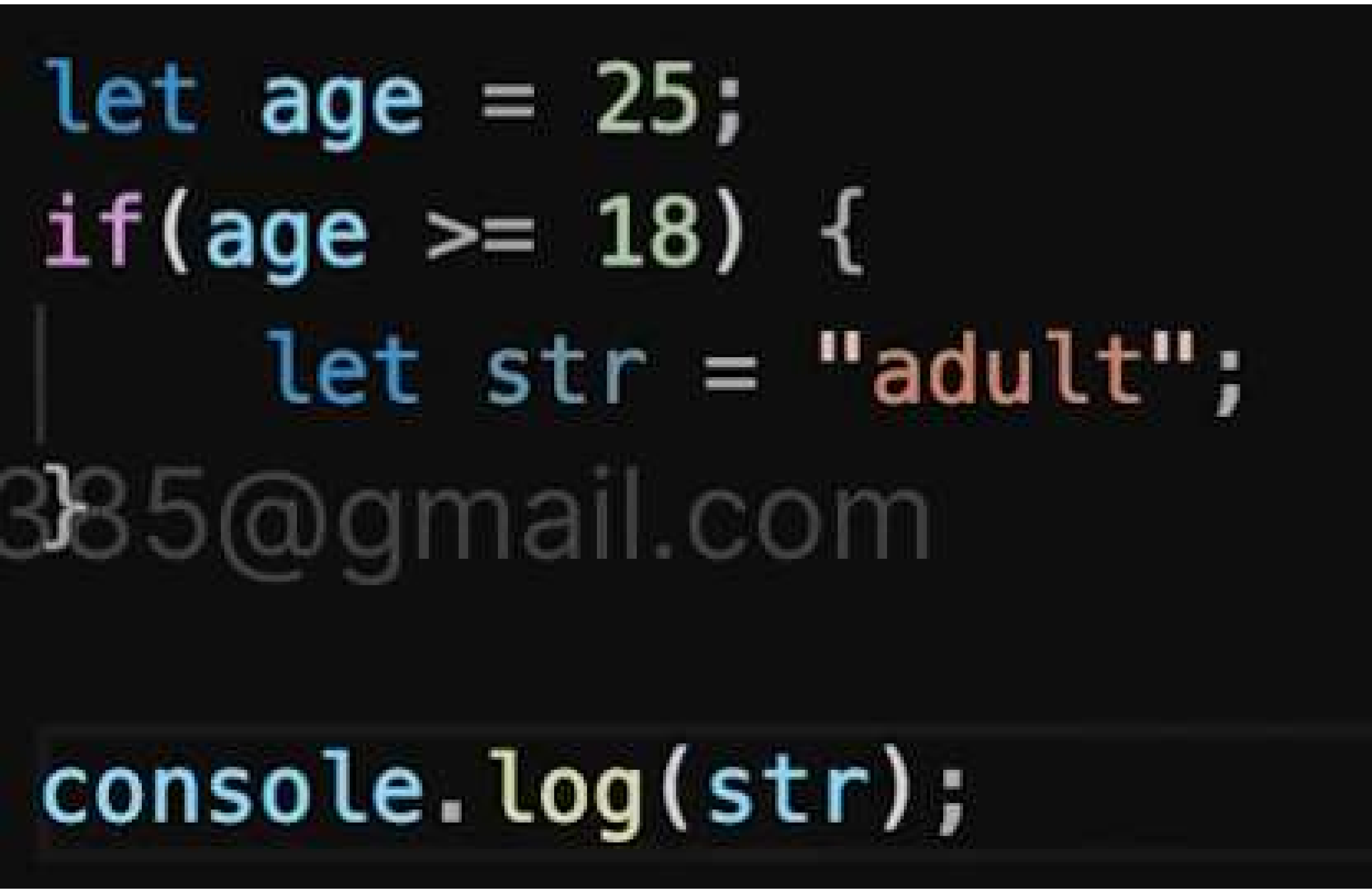
Block Scope

Variables declared inside a { } block cannot be accessed from outside the block.

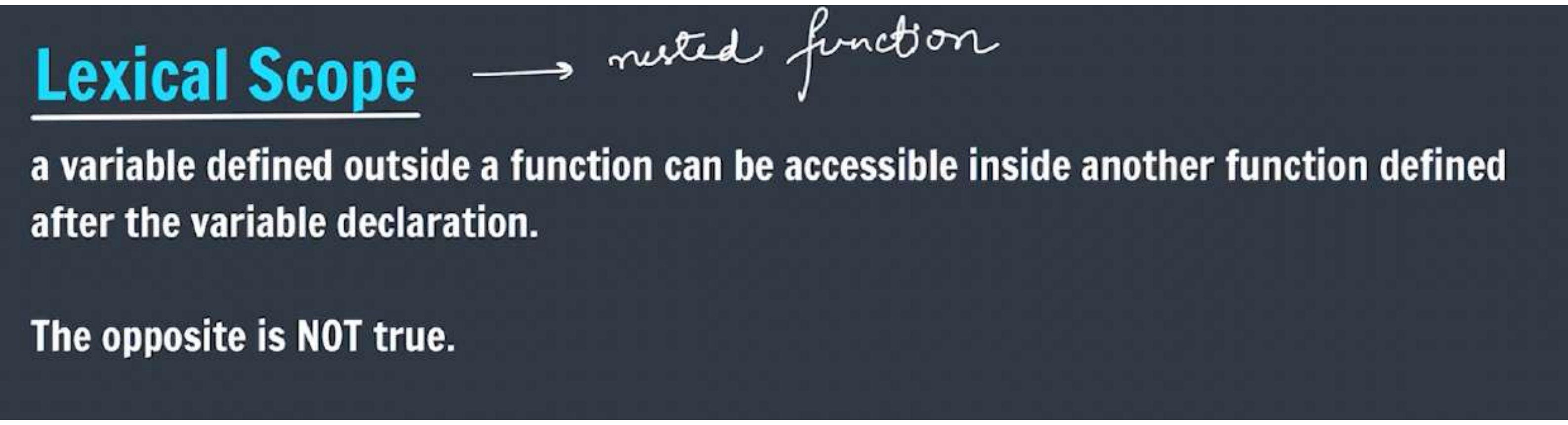




```
for(let i=1; i<=5; i++) {
    console.log(i); //blo
}

console.log(i);
```



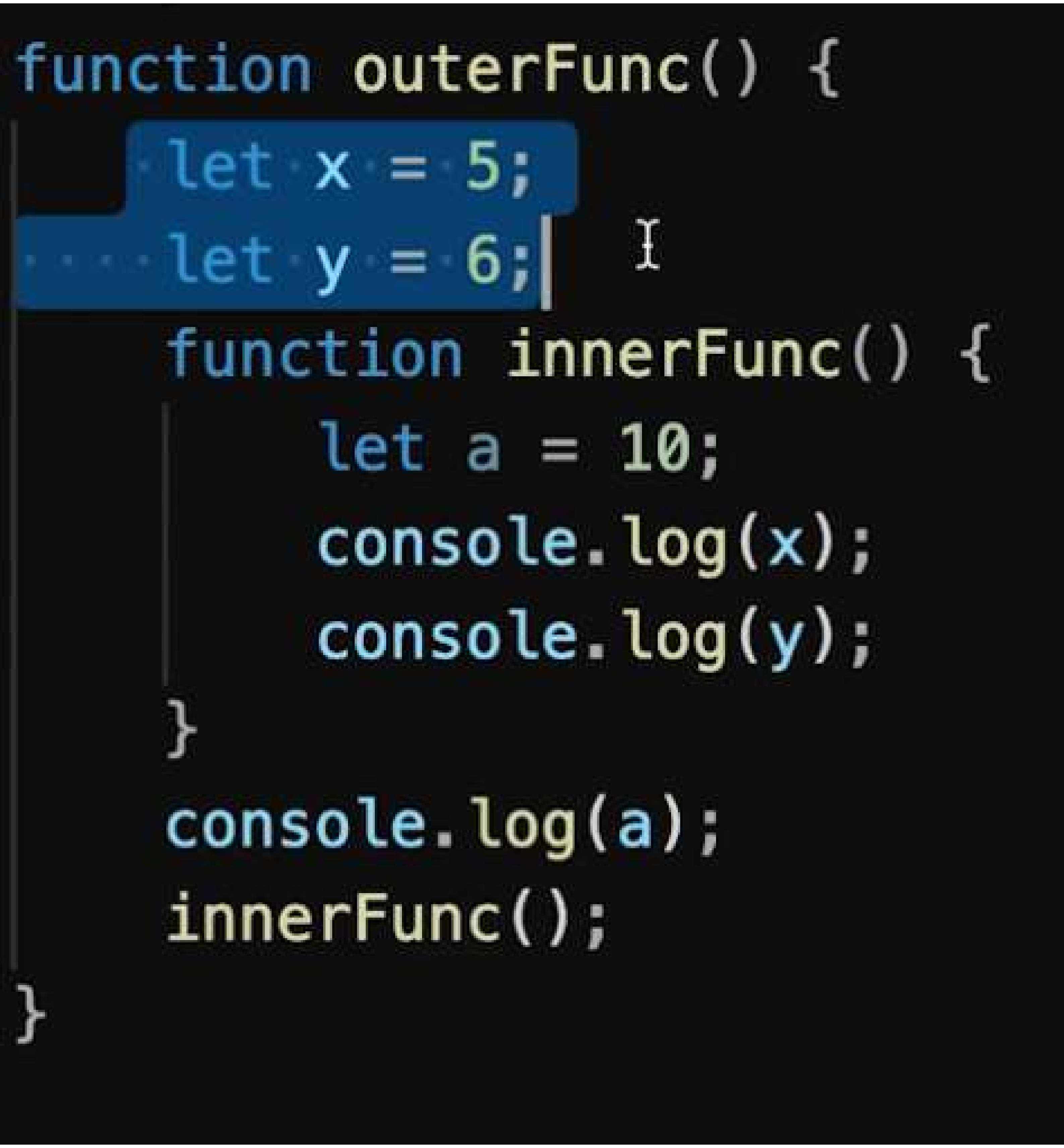
```
let age = 25;
if(age >= 18) {
    let str = "adult";
}

console.log(str);
```



Lexical Scope $\longrightarrow$ nested function

a variable defined outside a function can be accessible inside another function defined after the variable declaration.

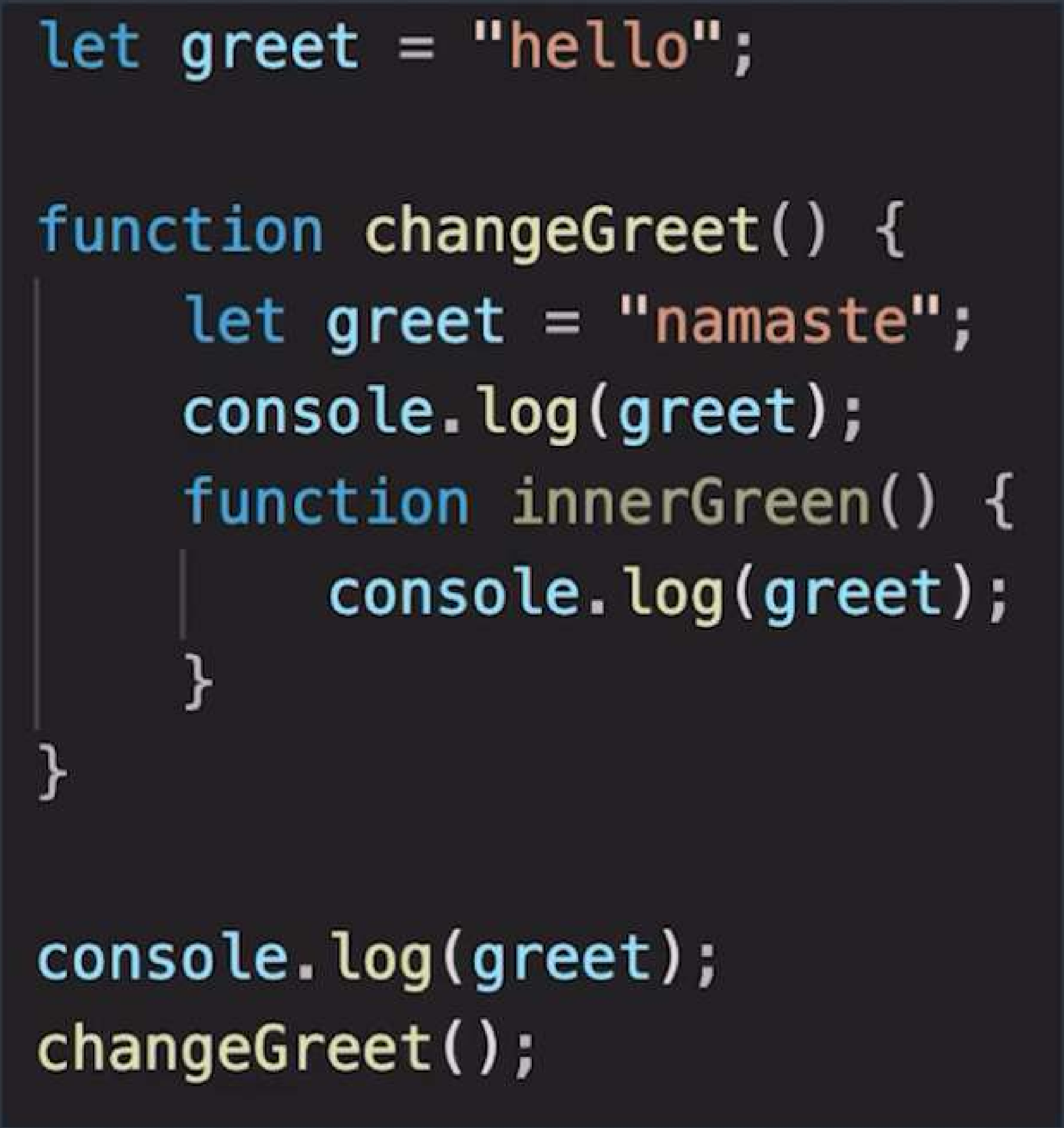
The opposite is NOT true.



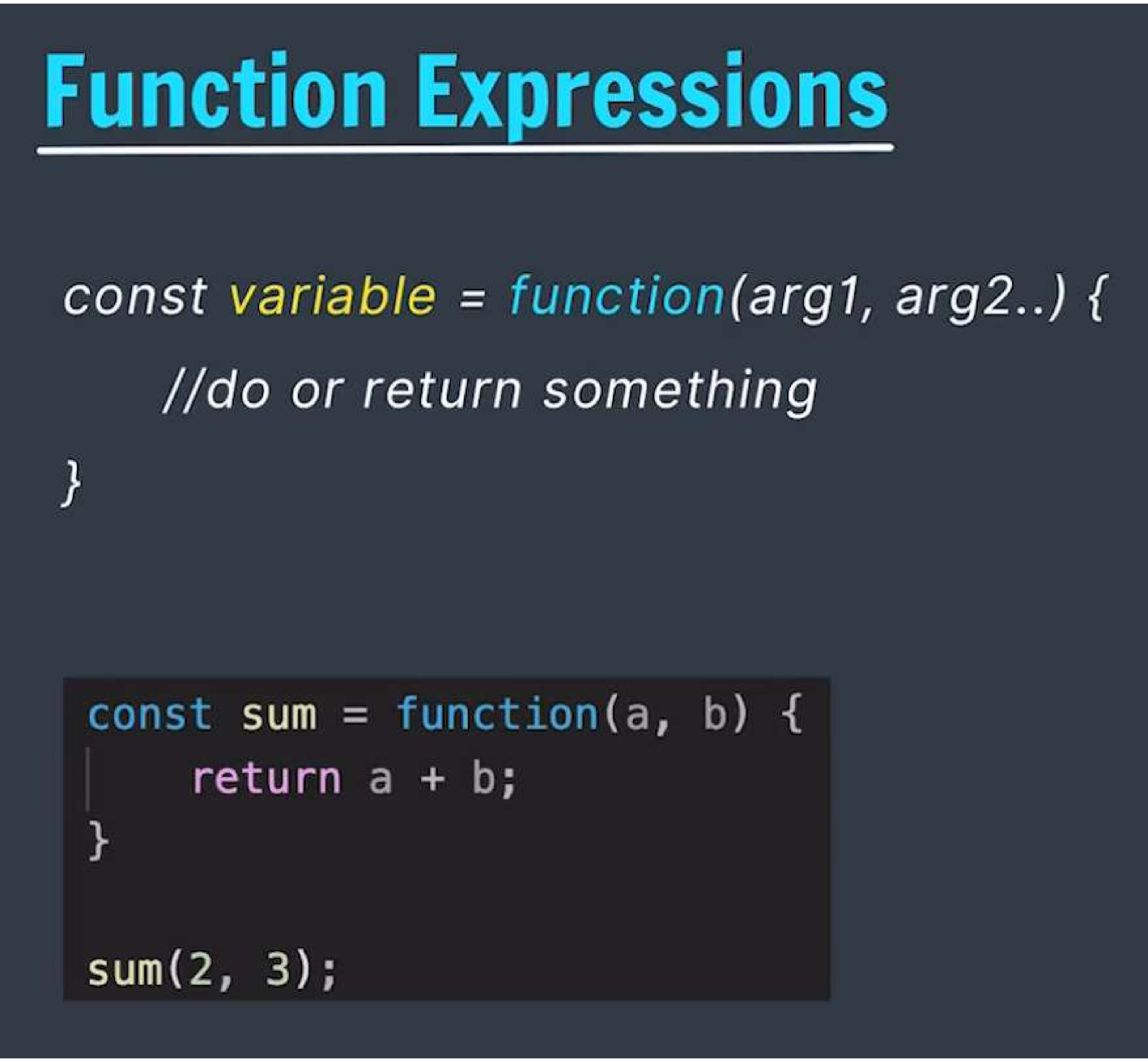
```
function outerFunc() {
    let x = 5;
    let y = 6;
    function innerFunc() {
        let a = 10;
        console.log(x);
        console.log(y);
    }
    console.log(a);
    innerFunc();
}
```

Practice Qs

What will be the output?



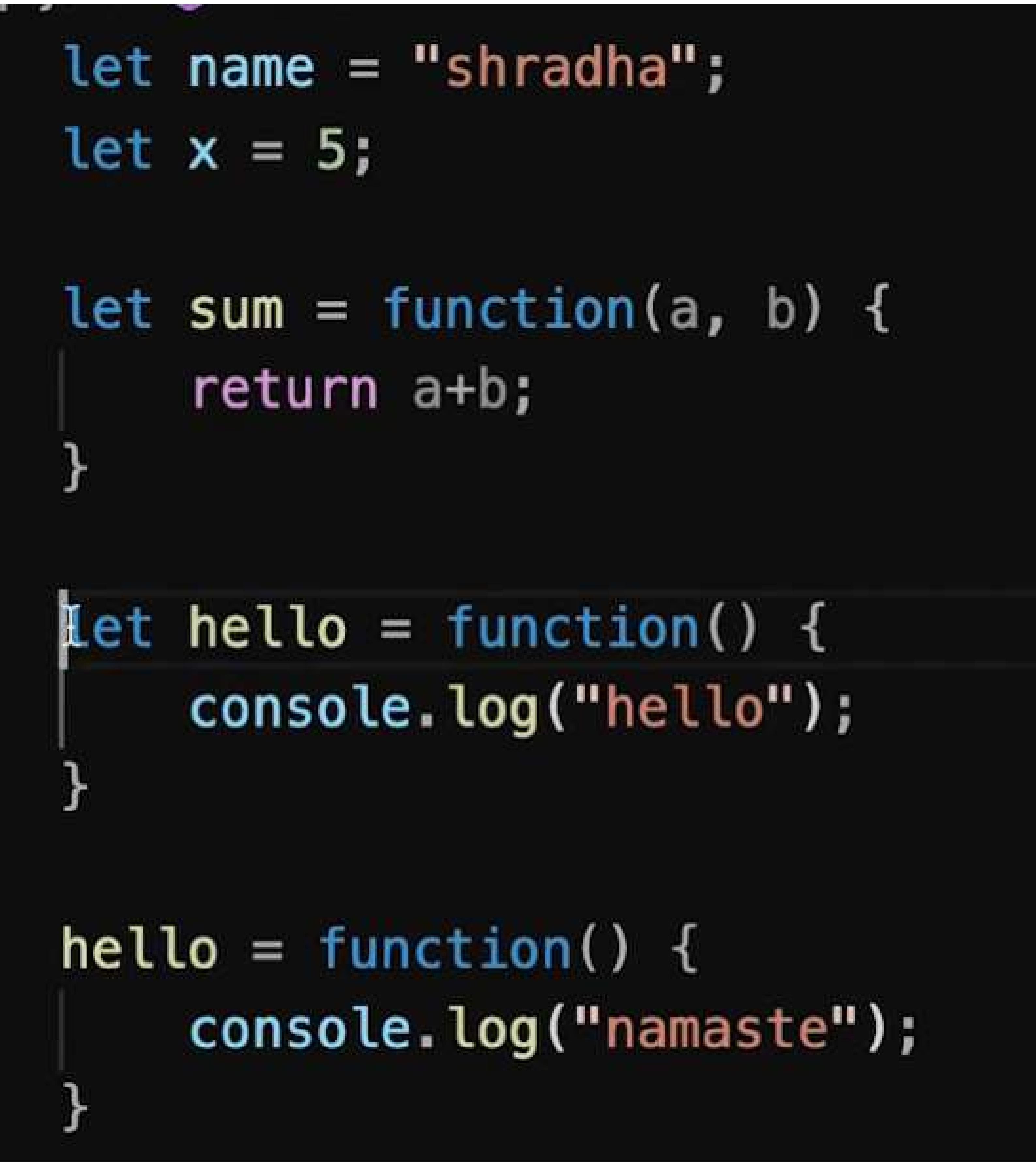
```
let greet = "hello";

function changeGreet() {
    let greet = "namaste";
    console.log(greet);
    function innerGreen() {
        console.log(greet);
    }
}

console.log(greet);
changeGreet();
```



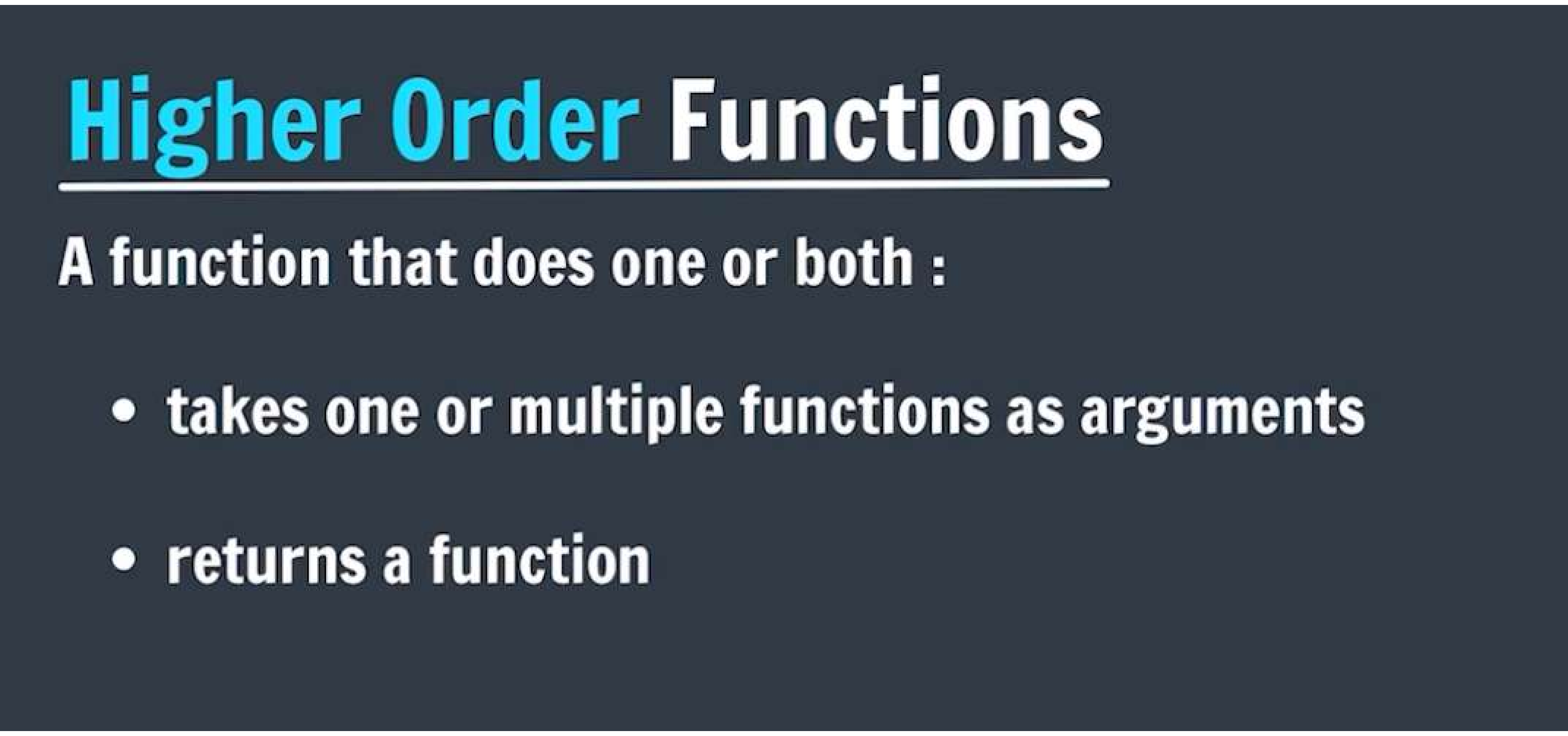
Function Expressions

```
const variable = function(arg1, arg2..) {
    //do or return something
}
```

```
const sum = function(a, b) {
    return a + b;
}

sum(2, 3);
```



```
let name = "shradha";
let x = 5;

let sum = function(a, b) {
    return a+b;
}

let hello = function() {
    console.log("hello");
}

hello = function() {
    console.log("namaste");
}
```



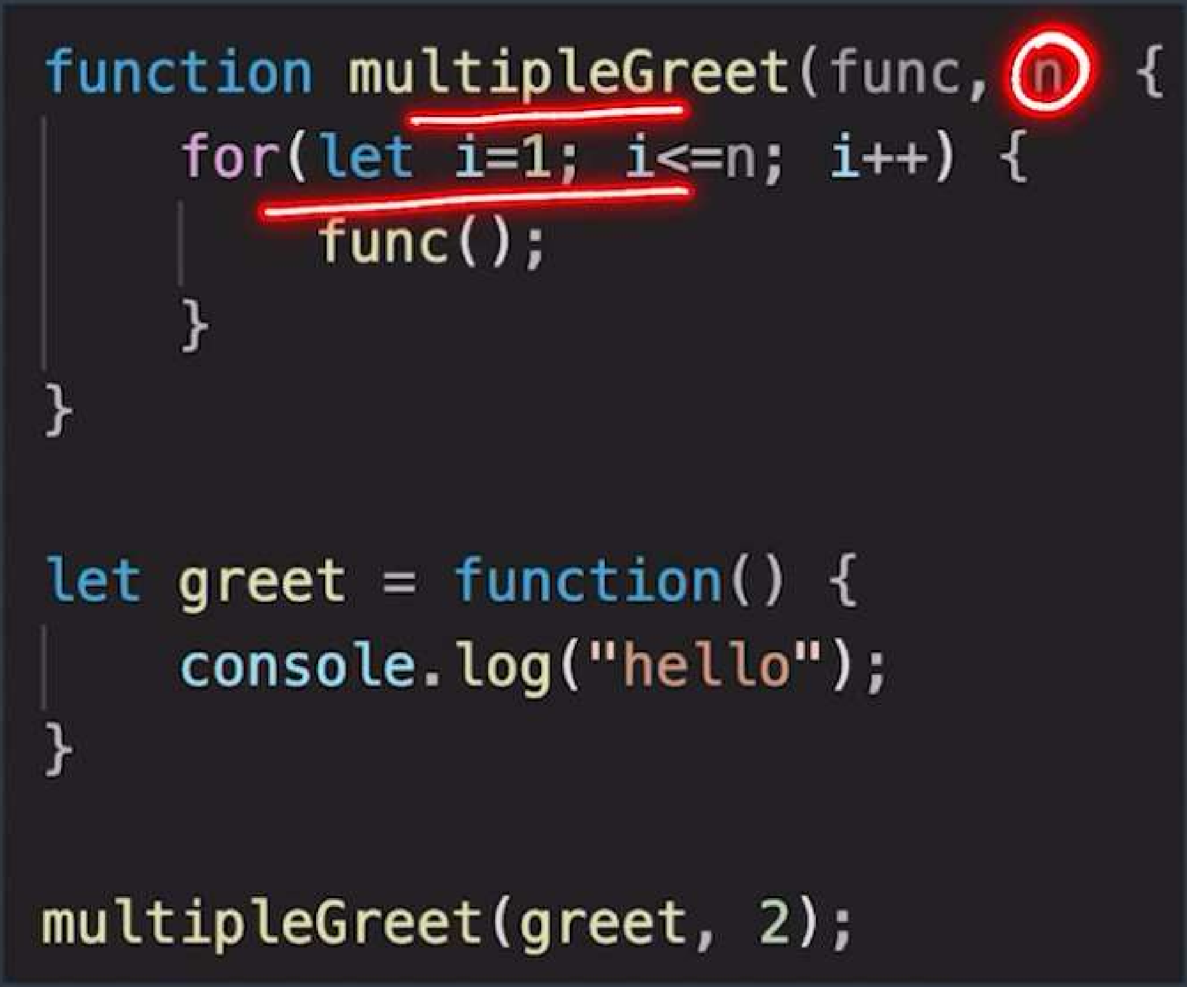
Higher Order Functions

A function that does one or both :

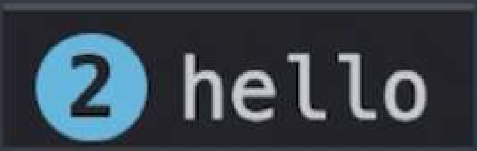
- takes one or multiple functions as arguments
- returns a function

Higher Order Functions

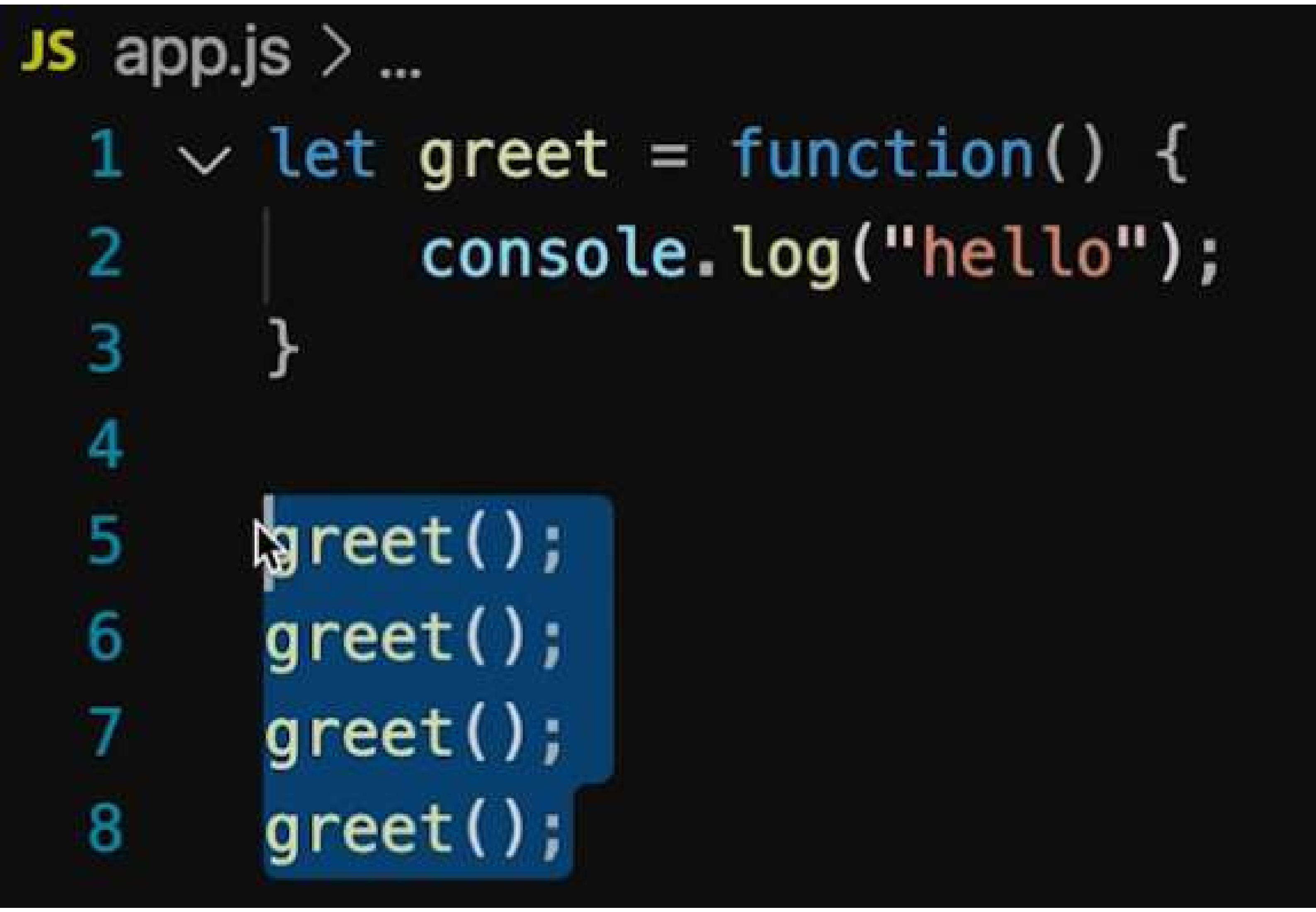
Takes one or multiple functions as arguments



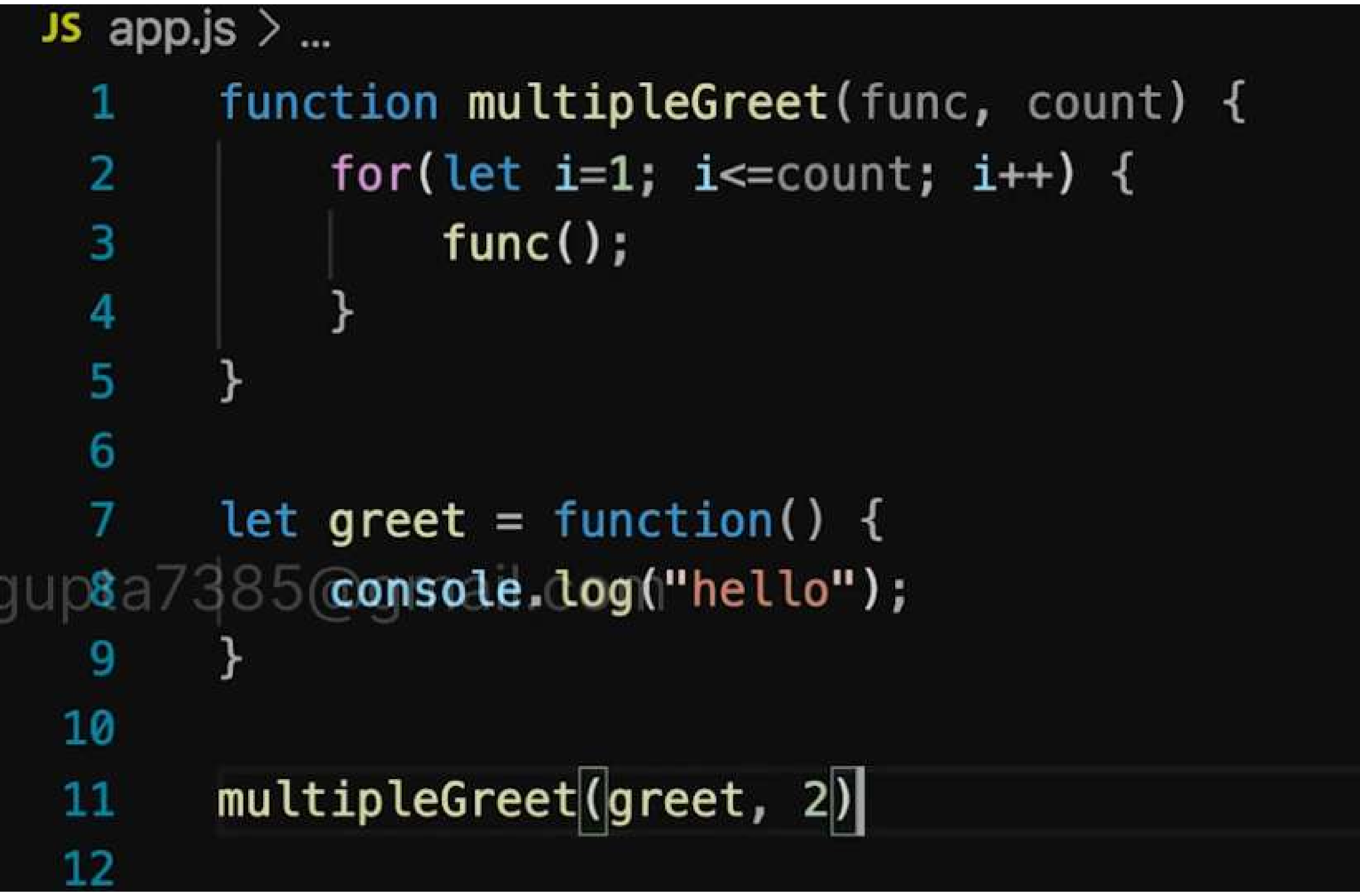
```
function multipleGreet(func, n) {
    for(let i=1; i<=n; i++) {
        func();
    }
}

let greet = function() {
    console.log("hello");
}

multipleGreet(greet, 2);
```



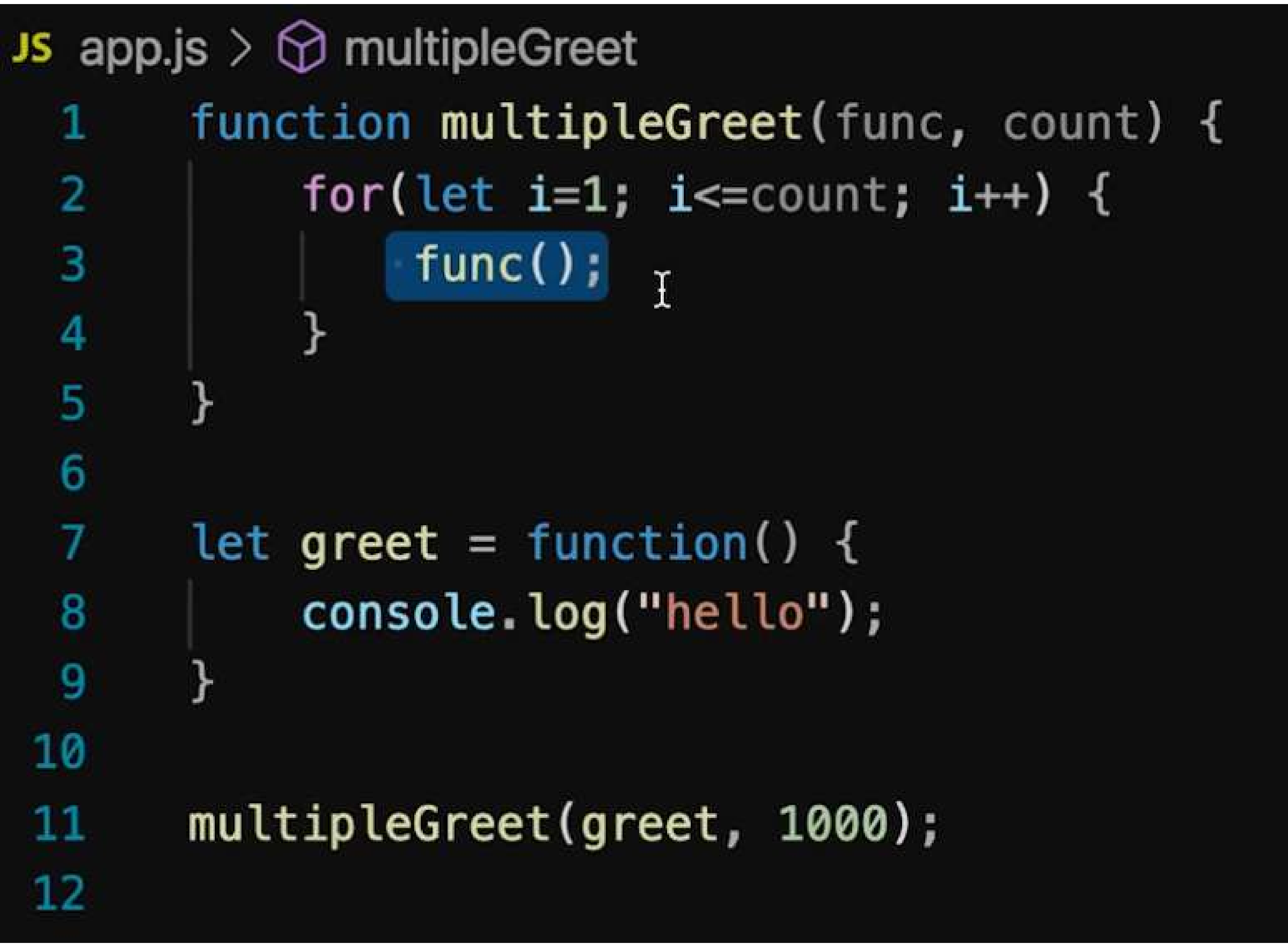
```
2 hello
```



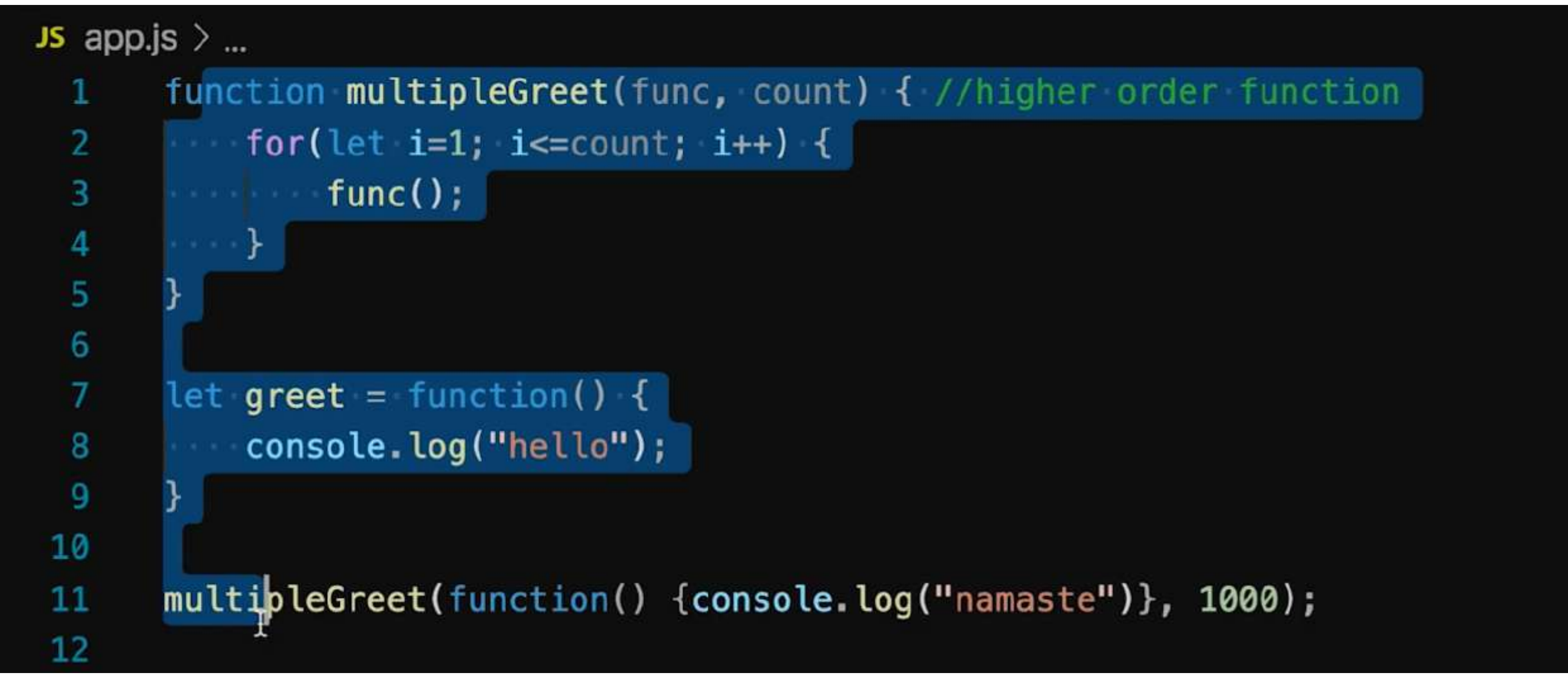
```javascript
let greet = function() {
    console.log("hello");
}

greet();
greet();
greet();
greet();
```



```javascript
function multipleGreet(func, count) {
    for(let i=1; i<=count; i++) {
        func();
    }
}

let greet = function() {
    console.log("hello");
}

multipleGreet(greet, 2)
```

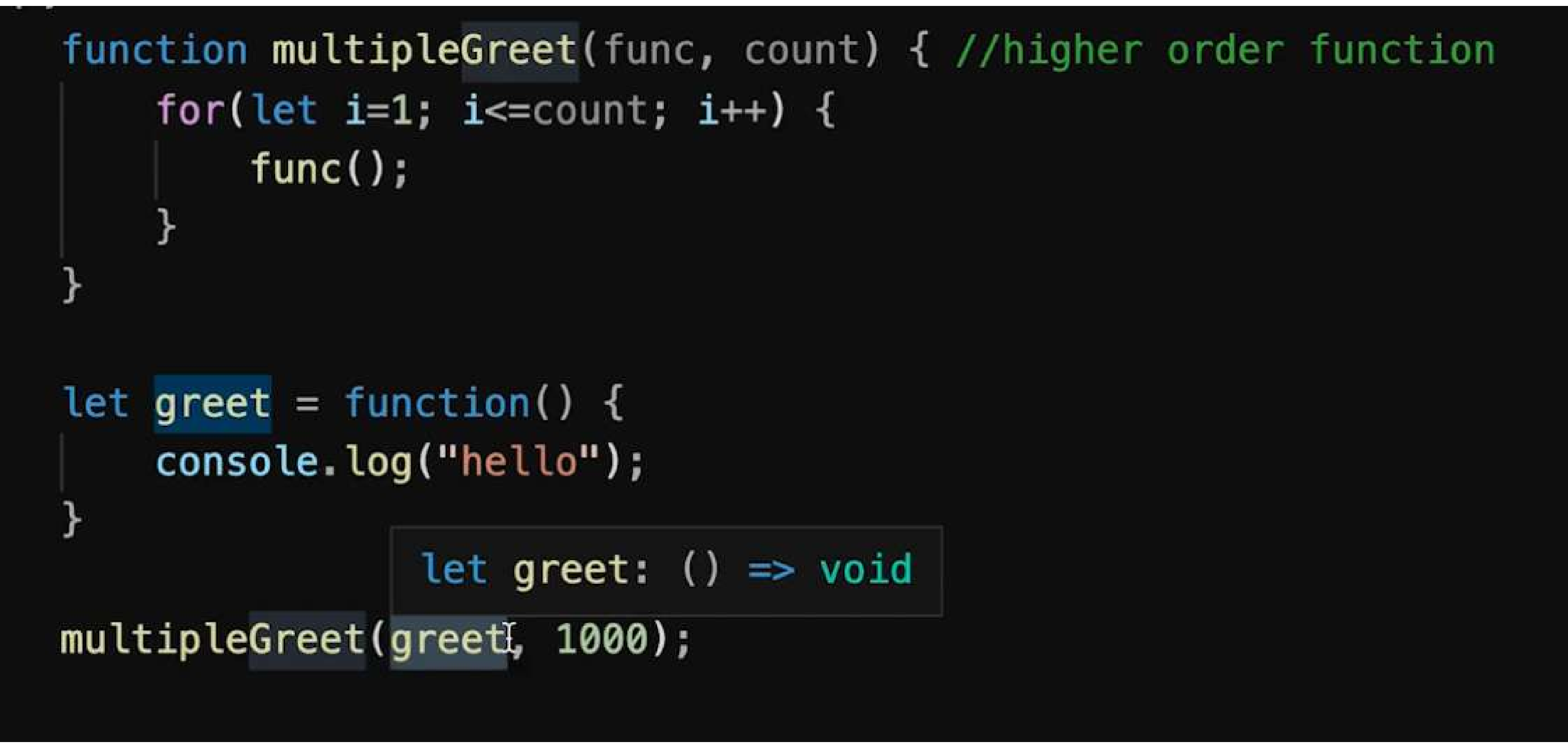


```javascript
function multipleGreet(func, count) {
    for(let i=1; i<=count; i++) {
        func();
    }
}

let greet = function() {
    console.log("hello");
}

multipleGreet(greet, 1000);
```

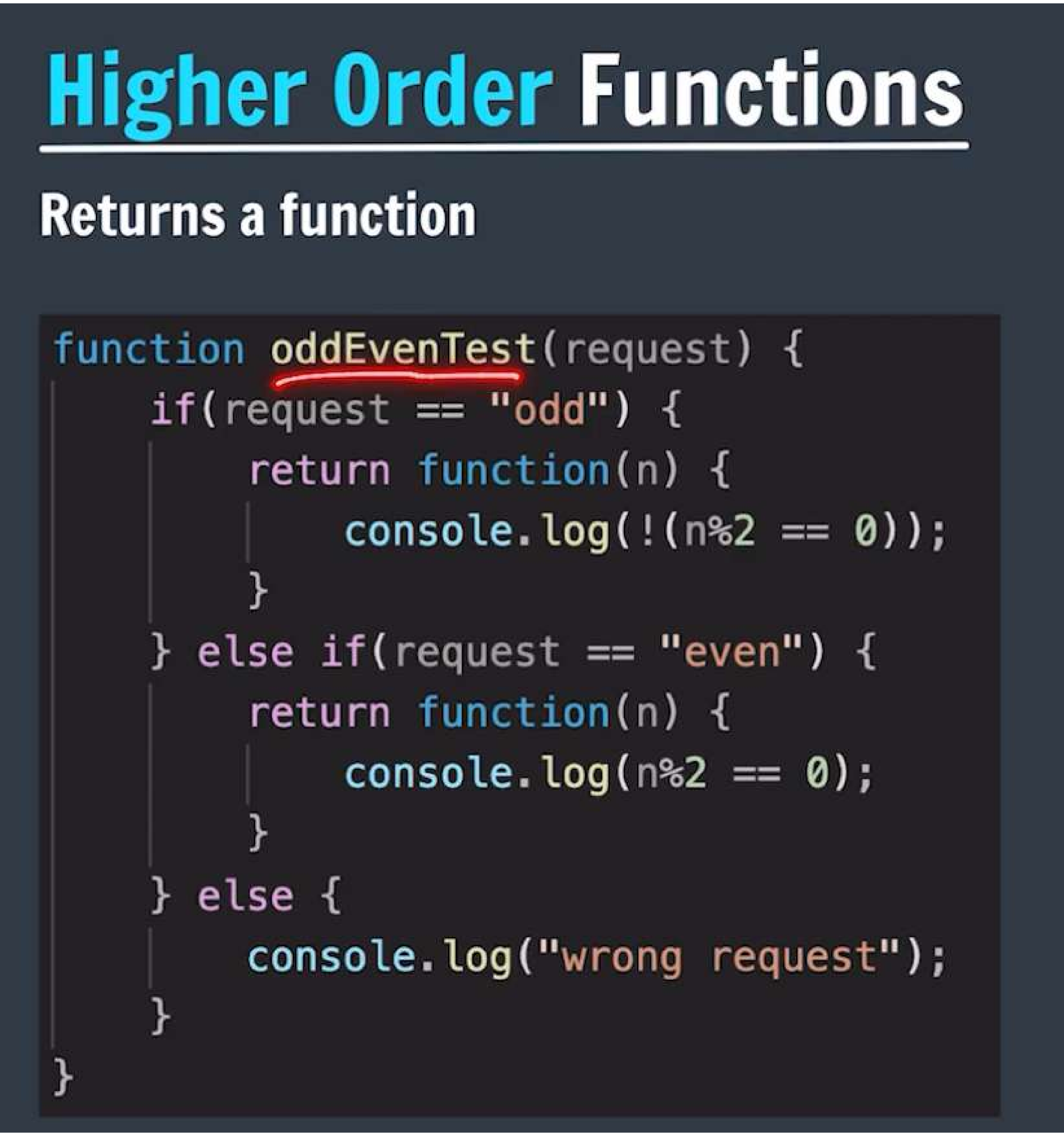


app.js > ...

```javascript
function multipleGreet(func, count) { //higher order function
    for(let i=1; i<=count; i++) {
        func();
    }
}

let greet = function() {
    console.log("hello");
}

multipleGreet(function() {console.log("namaste")}, 1000);
```



```
function multipleGreet(func, count) { //higher order function
    for(let i=1; i<=count; i++) {
        func();
    }
}

let greet = function() {
    console.log("hello");
}
                    let greet: () => void
multipleGreet(greet, 1000);
```



Higher Order Functions

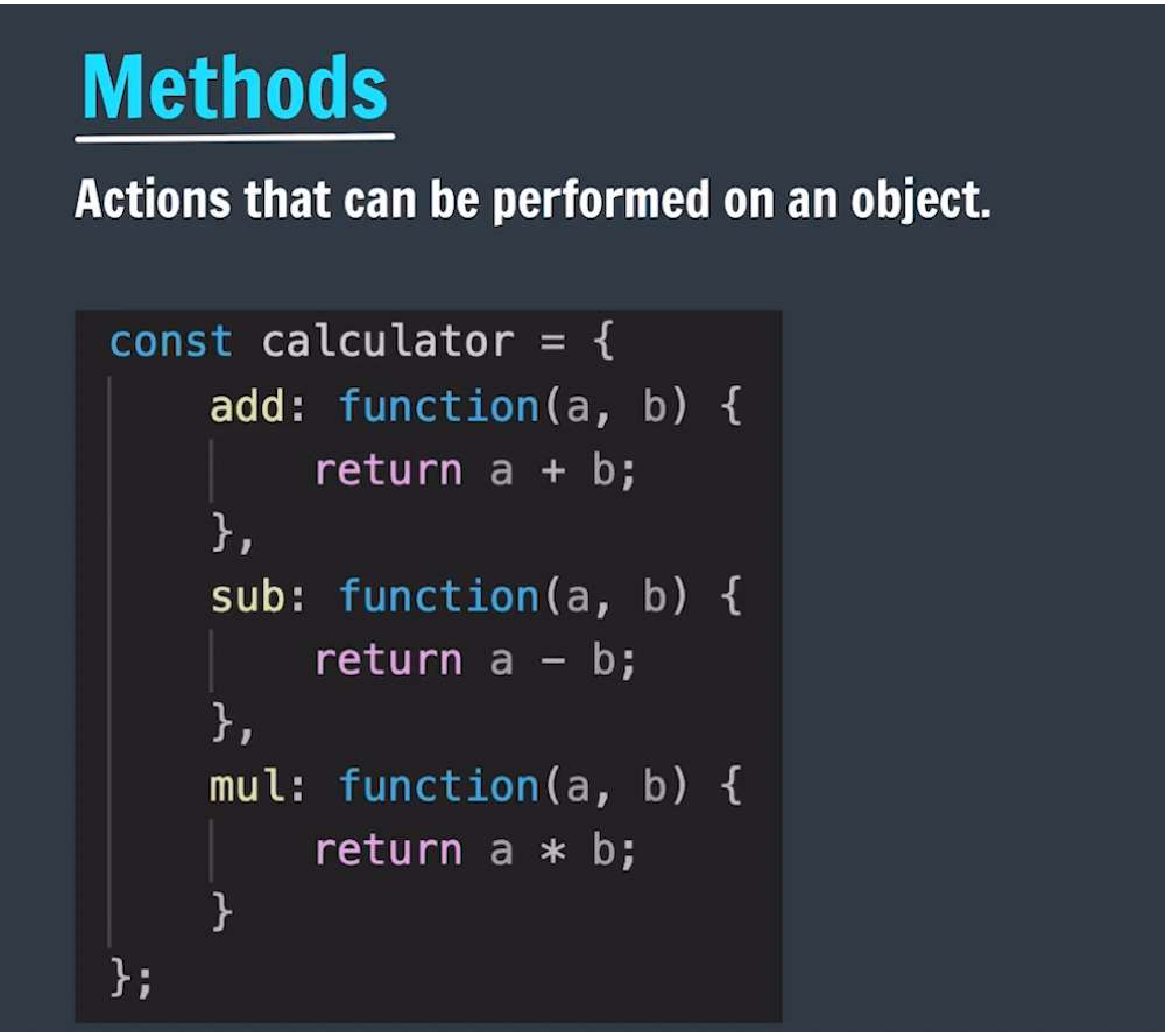
Returns a function

```javascript
function oddEvenTest(request) {
    if(request == "odd") {
        return function(n) {
            console.log(!(n%2 == 0));
        }
    } else if(request == "even") {
        return function(n) {
            console.log(n%2 == 0);
        }
    } else {
        console.log("wrong request");
    }
}
```

Methods

Actions that can be performed on an object.

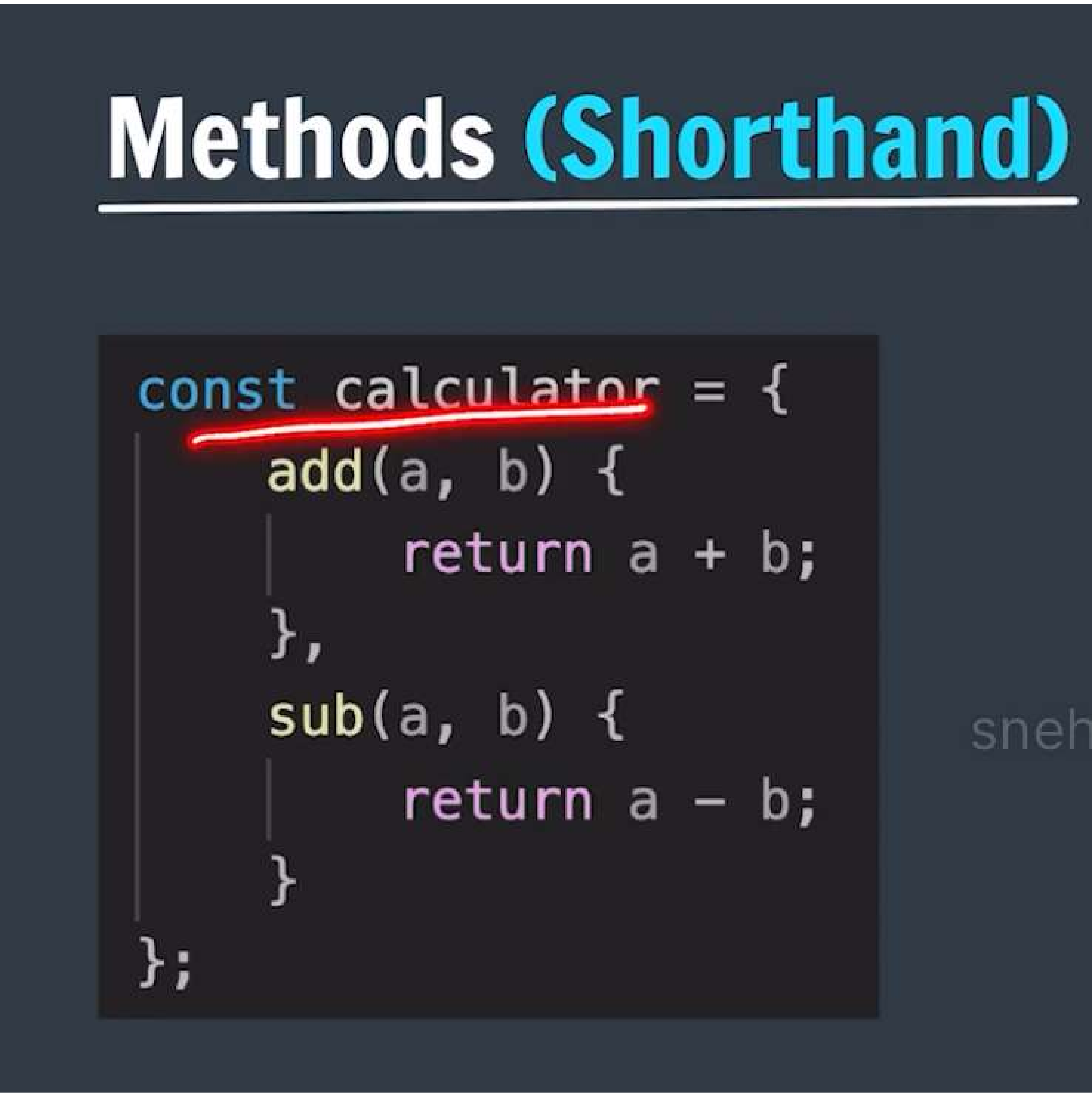
```
const calculator = {
    add: function(a, b) {
        return a + b;
    },
    sub: function(a, b) {
        return a - b;
    },
    mul: function(a, b) {
        return a * b;
    }
};
```



Methods

Actions that can be performed on an object.

```
const calculator = {
    add: function(a, b) {
        return a + b;
    },
    sub: function(a, b) {
        return a - b;
    },
    mul: function(a, b) {
        return a * b;
    }
};
```



Methods (Shorthand)

```
const calculator = {
    add(a, b) {
        return a + b;
    },
    sub(a, b) {
        return a - b;
    }
};
```