QUIZ!

What is the correct way to write a JavaScript array?

a. var colors = "red", "green", "blue“
b. var colors = (1:"red", 2:"green", 3:"blue")
c. var colors = 1 = ("red"), 2 = ("green"), 3 = ("blue")
d. var colors = ["red", "green", "blue"]

What is the correct way to write a JavaScript array?

a. var colors = "red", "green", "blue“
b. var colors = (1:"red", 2:"green", 3:"blue")
c. var colors = 1 = ("red"), 2 = ("green"), 3 = ("blue")
d. var colors = ["red", "green", "blue"]

- How do you call a function named "myFunction" in Javascript?

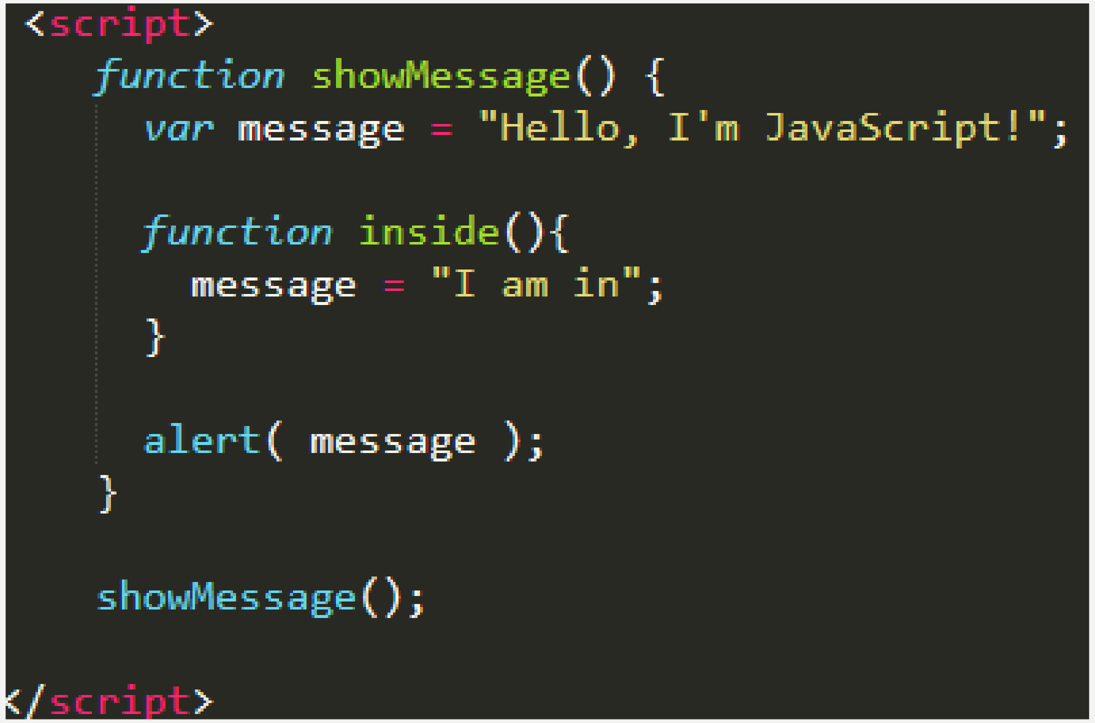
a. call myFunction()
b. function myFunction()
c. myFunction()

- How do you call a function named "myFunction" in Javascript?

a. call myFunction()
b. function myFunction()
c. myFunction()

What will be alerted when the code below is executed?

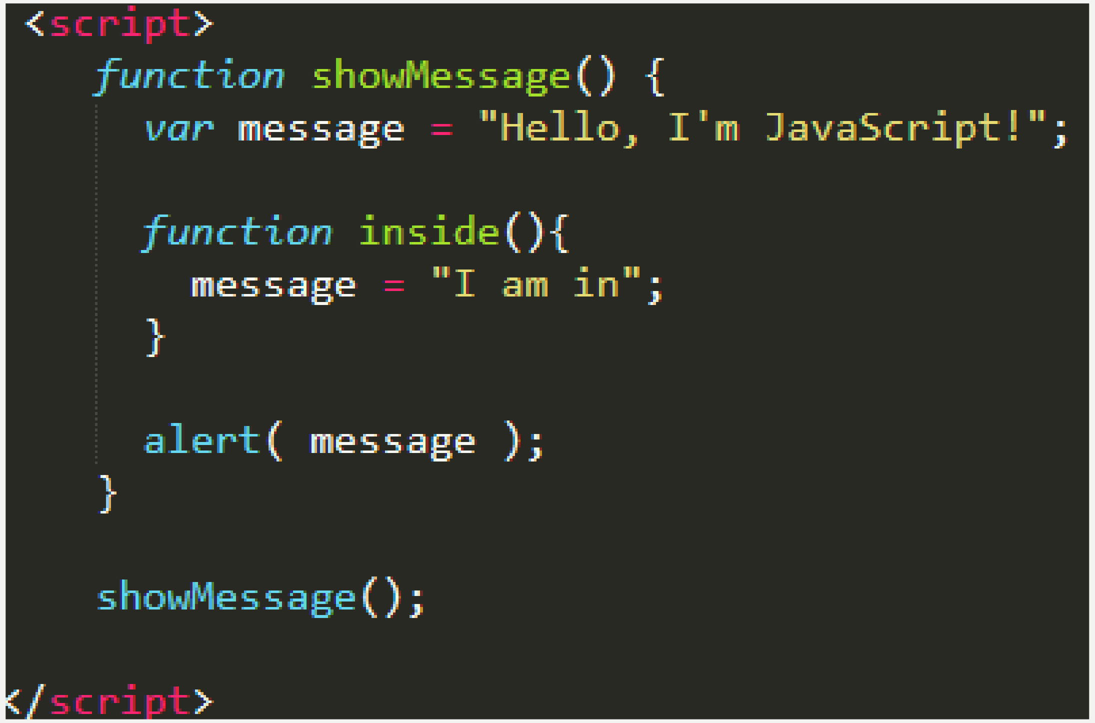
- "Hello, I'm JavaScript!"
- "I am in"
- An error will show



```
<script>
   function showMessage() {
     var message = "Hello, I'm JavaScript!";

     function inside(){
       message = "I am in";
     }

     alert( message );
   }

   showMessage();

</script>
```

What will be alerted when the code below is executed?

- "Hello, I'm JavaScript!"
- "I am in"
- An error will show



```
<script>
   function showMessage() {
     var message = "Hello, I'm JavaScript!";

     function inside(){
       message = "I am in";
     }

     alert( message );
   }

   showMessage();

</script>
```

Which of these is not a logical operator?

a. !
b. &
c. &&
d. ||

Which of these is not a logical operator?

a. !
b. &
c. &&
d. ||

What is the value of x ?

```
var a = false;
 var x = a ? "A" : "B";
```

a. Undefined
b. True
c. "A"
d. "B"

What is the value of x ?

```
var a = false;
 var x = a ? "A" : "B";
```

a. Undefined
b. True
c. "A"
d. "B"

This is translated: if a is truthy then A otherwise B

if (a) { x = A; } else { x = B; }

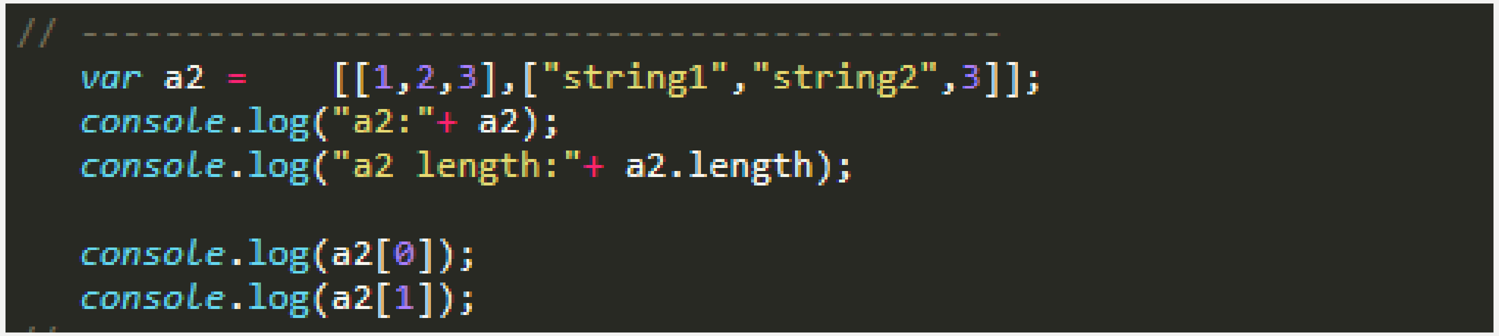
WHAT ARE B* BELOW? TRUE OR FALSE?

```
var    B1    =          2!="2";

var    B2     =         2=="2";

var    B3     =         2!=="2";

var    B4     =        2==="2";
```

QUESTION IS B4 BELOW?
TRUE OR FALSE?

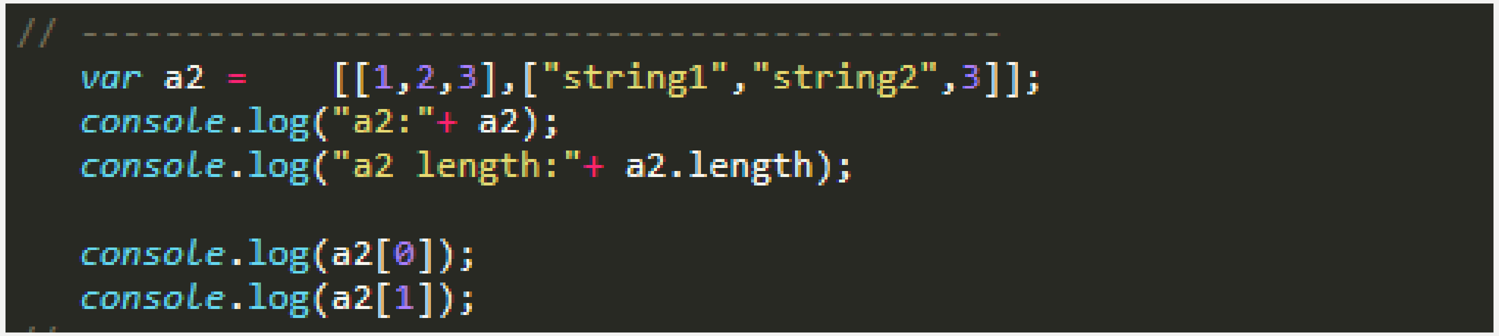
```
var   b1   =        2!="2";            >>  false

var   b2   =        2=="2";            >>  true

var   b3   =        2!=="2";           >>  true

var   b4   =       2==="2";            >>  false
```

WHAT DO I EXPECT TO SEE?

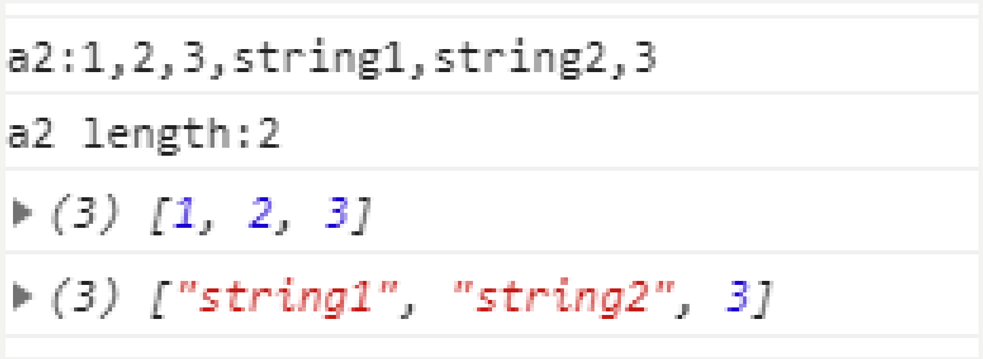


```
// ----------------------------------------------
   var a2 =    [[1,2,3],["string1","string2",3]];
   console.log("a2:"+ a2);
   console.log("a2 length:"+ a2.length);

   console.log(a2[0]);
   console.log(a2[1]);
```

WHAT DO I EXPECT TO SEE?

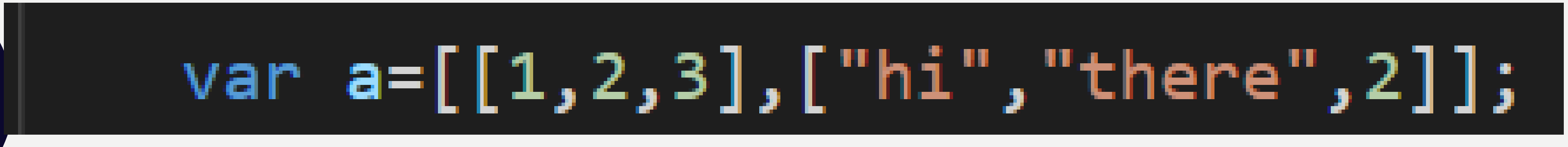


```
// -------------------------------------------
   var a2 =    [[1,2,3],["string1","string2",3]];
   console.log("a2:"+ a2);
   console.log("a2 length:"+ a2.length);

   console.log(a2[0]);
   console.log(a2[1]);
```



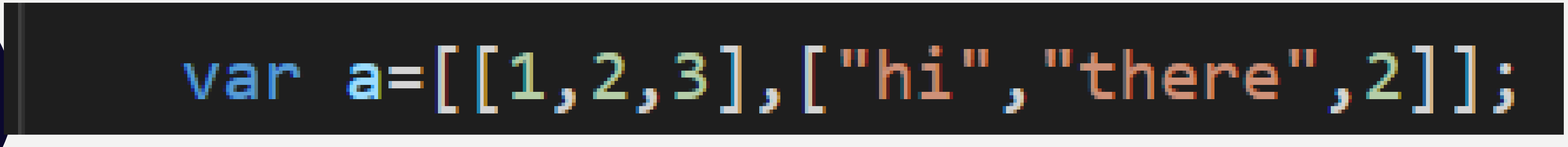
```
a2:1,2,3,string1,string2,3
a2 length:2
▶ (3) [1, 2, 3]
▶ (3) ["string1", "string2", 3]
```

- Can I access “hi”? If yes how?

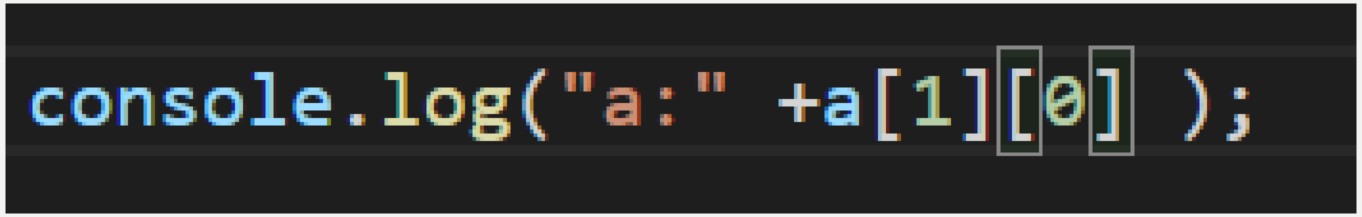


```
var a=[[1,2,3],["hi","there",2]];
```

- Can I access “hi”? If yes how?



```
var a=[[1,2,3],["hi","there",2]];
```



```
console.log("a:" +a[1][0] );
```

How can we append a value to an array in Javascript?

How can we append a value to an array in Javascript?

```
arr[arr.length] = value
arr.push(value) ;
```

What is the purpose of a "return" statement in a function ?

a. Returns the value and continues executing rest of the statements, if any

b. Returns the error if any

c. Stops executing the function and returns the value

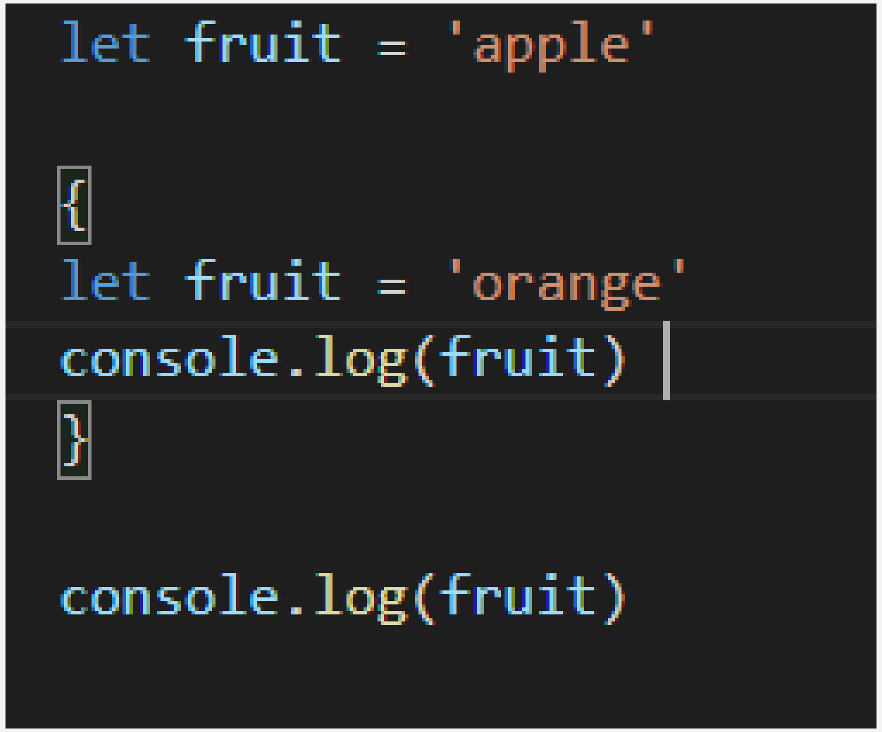
What is the purpose of a "return" statement in a function ?

a. Returns the value and continues executing rest of the statements, if any

b. Returns the error if any

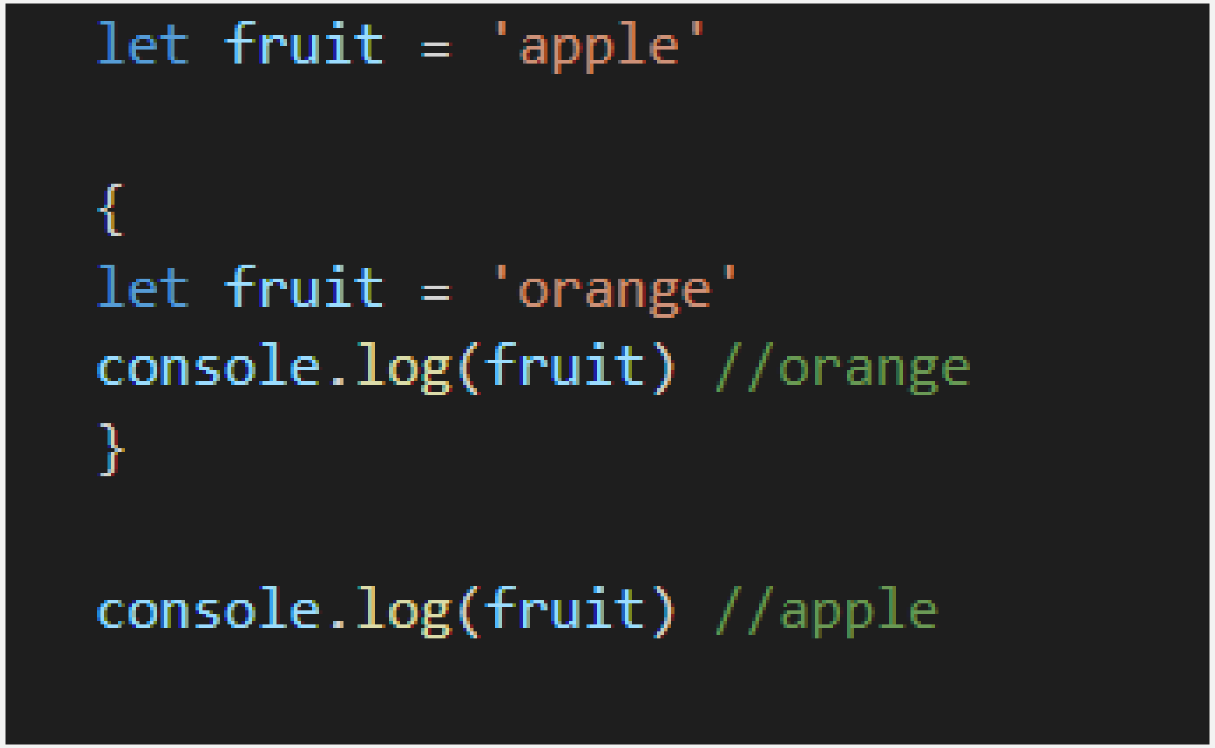
c. Stops executing the function and returns the value

- What is the output of the console.logs below?



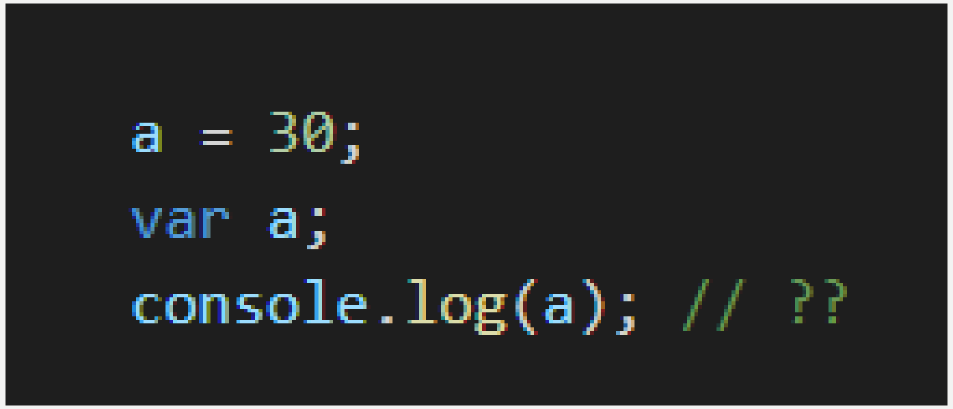
```
let fruit = 'apple'

{
let fruit = 'orange'
console.log(fruit)
}

console.log(fruit)
```

- What is the output of the console.logs below?



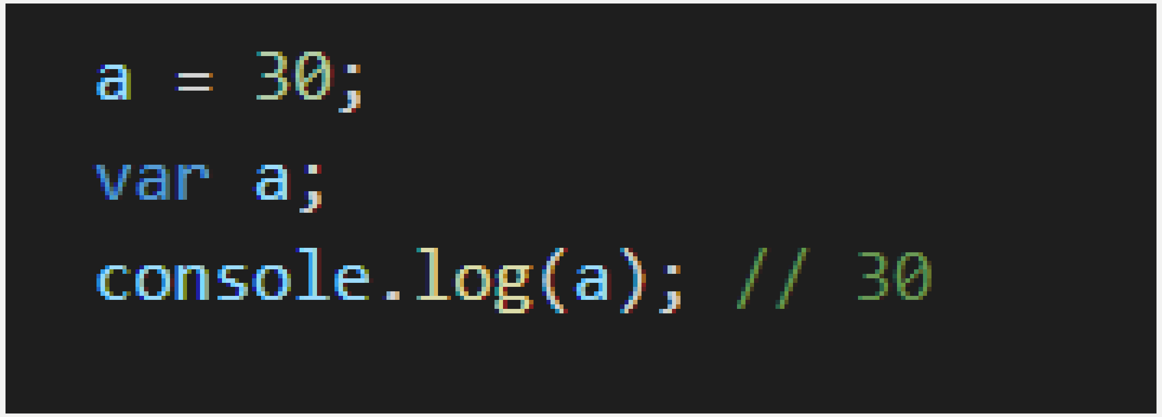
```
let fruit = 'apple'

{
let fruit = 'orange'
console.log(fruit) //orange
}

console.log(fruit) //apple
```

- What is the output of the console.log below?



```
a = 30;
var a;
console.log(a); // ??
```

- What is the output of the console.log below?



```
a = 30;
var a;
console.log(a); // 30
```

JAVASCRIPT HOISTING

- When a **JavaScript engine executes a script** -> it creates the execution context
- The execution context has two phases:
  - creation phase
  - execution phase.

- During the creation phase-> JavaScript engine moves the variable and function **declarations** to the top of the current scope (to the top of the current script or the current function).

JAVASCRIPT HOISTING

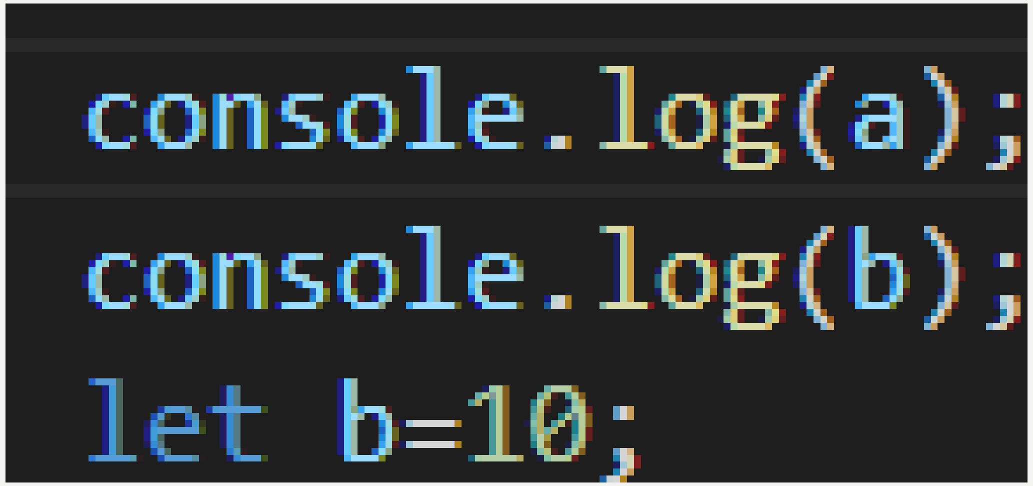
- Hoisting : JavaScript's default behavior of **moving declarations to the top** of current scope(local or global) **before code executions**.

- Hoisting -> allows functions/vars to be safely used in code before they are declared.

JAVASCRIPT HOISTING

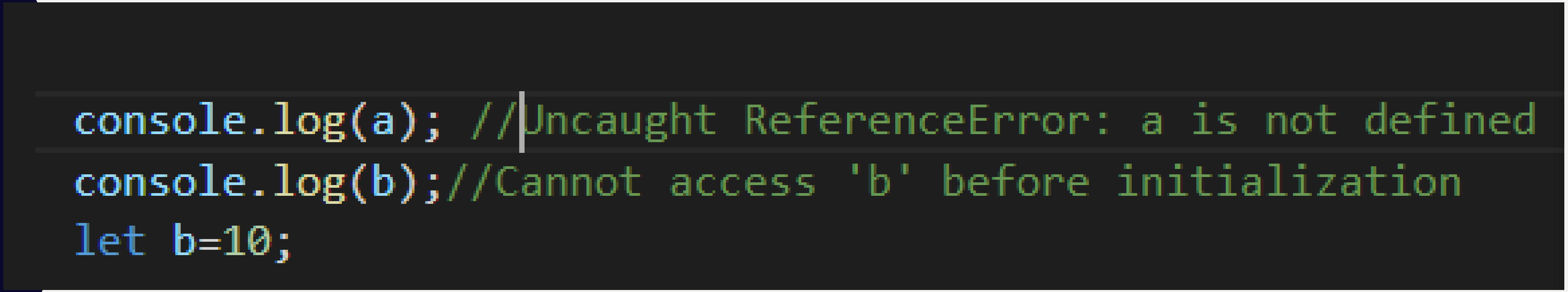
- JavaScript engine hoists the variables declared using the let keyword, **but it doesn’t initialize them** as the variables declared with the var keyword -> it does not work with **let** keyword
- Variables defined with let and const are hoisted to the top of the block, **but not initialized.**
- **Function expressions** and **arrow functions** aren’t hoisted.

- Will we get some errors here? If yes, will these errors be identical?



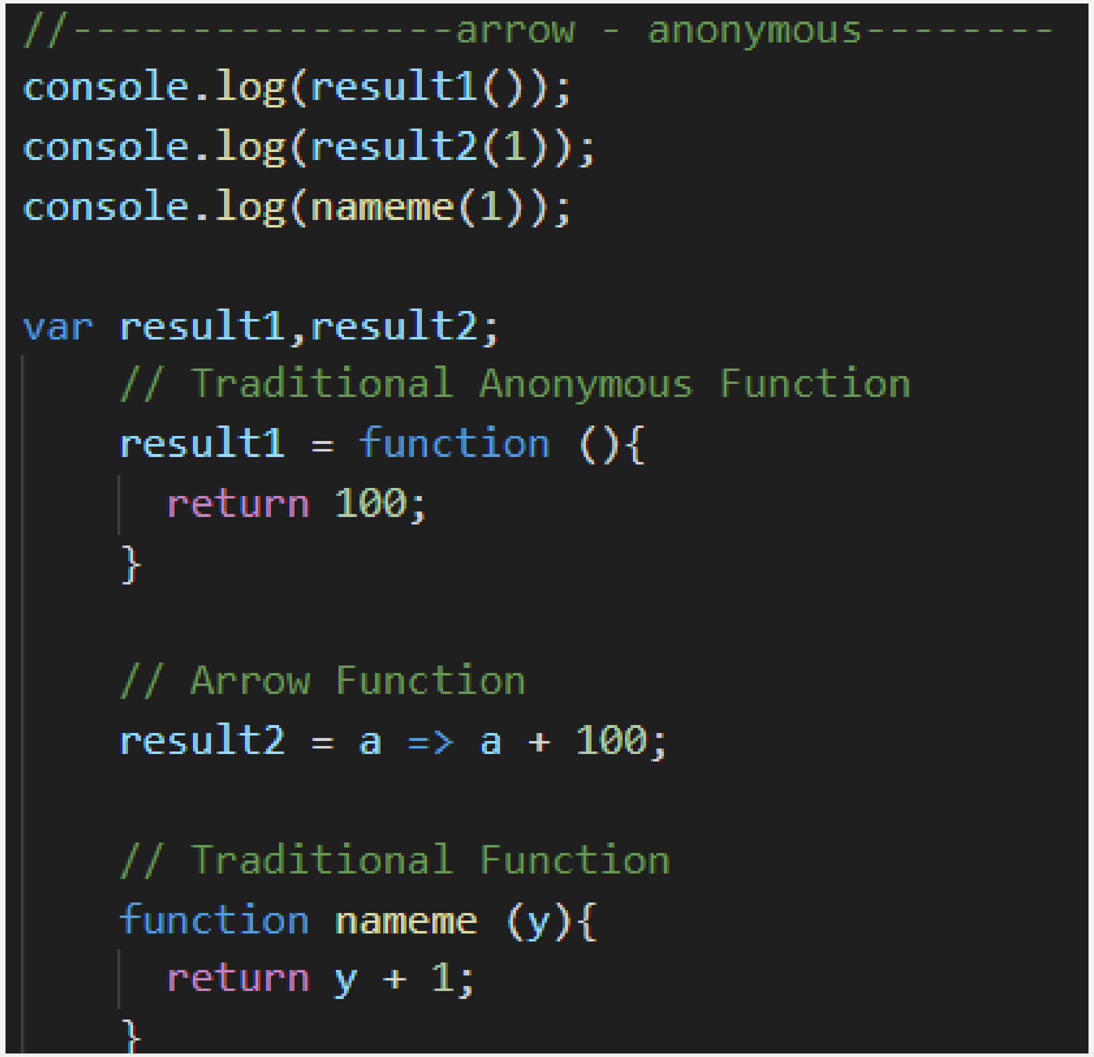
```
console.log(a);
console.log(b);
let b=10;
```

- Will we get some errors here? If yes, will these errors be identical?



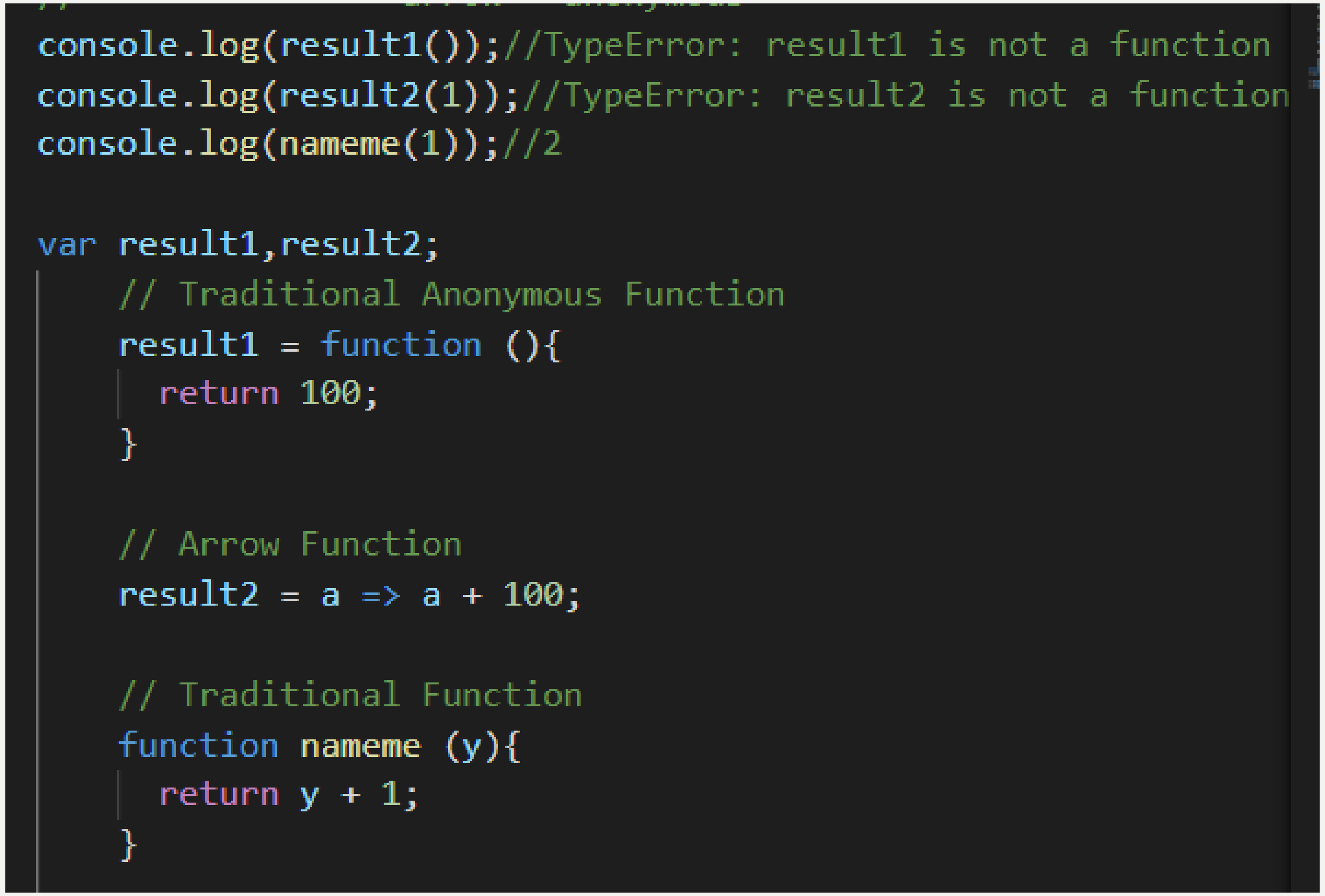
```
console.log(a); //Uncaught ReferenceError: a is not defined
console.log(b);//Cannot access 'b' before initialization
let b=10;
```

- Will we get any errors below?



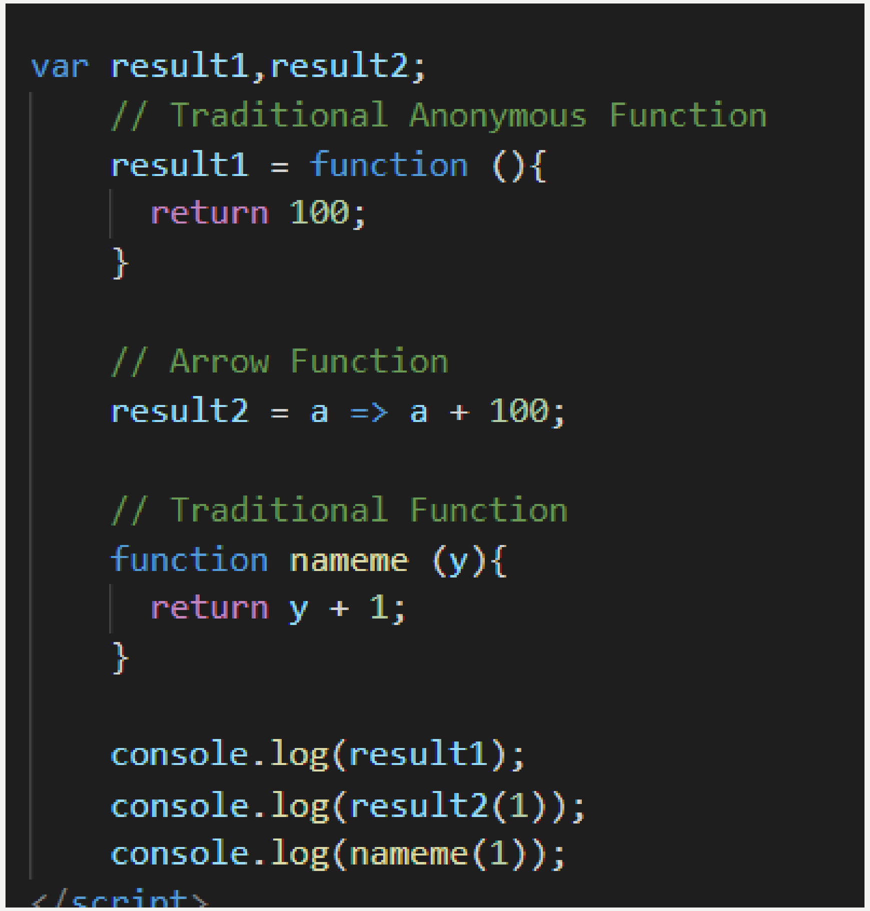
```
//----------------arrow - anonymous--------
console.log(result1());
console.log(result2(1));
console.log(nameme(1));

var result1,result2;
    // Traditional Anonymous Function
    result1 = function (){
      return 100;
    }

    // Arrow Function
    result2 = a => a + 100;

    // Traditional Function
    function nameme (y){
      return y + 1;
    }
```

- Will we get any errors below?



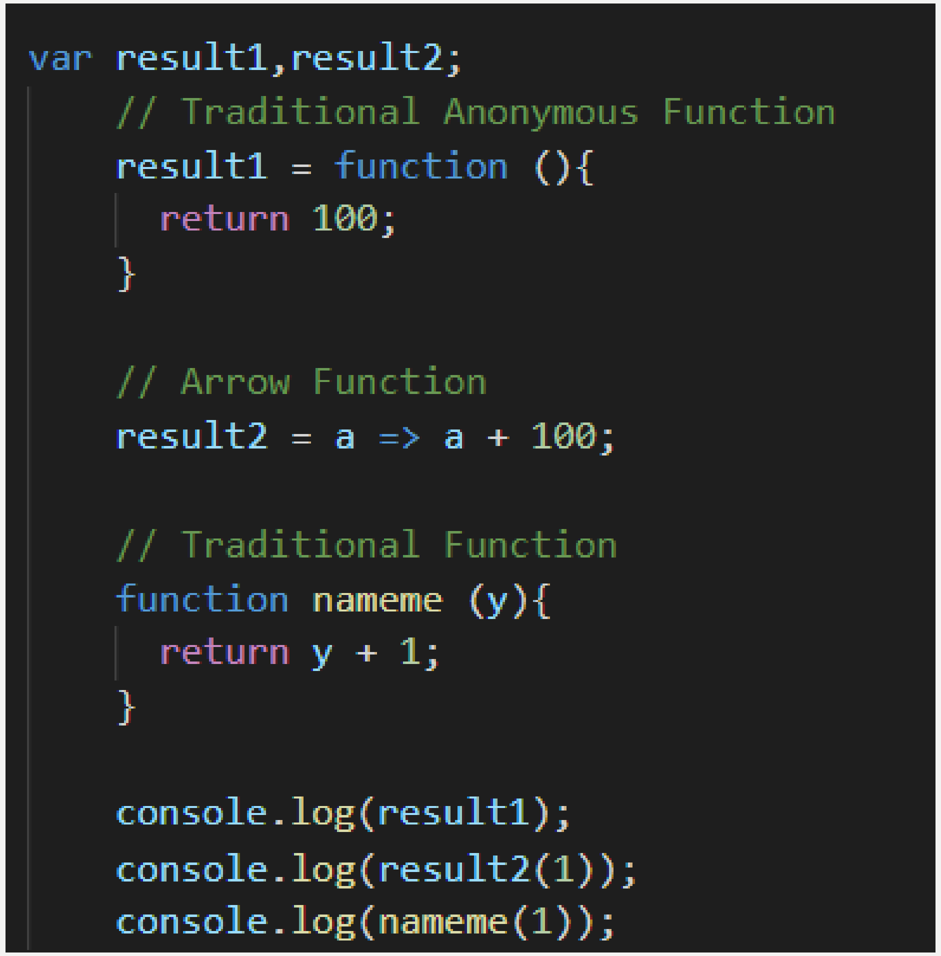
```
console.log(result1());//TypeError: result1 is not a function
console.log(result2(1));//TypeError: result2 is not a function
console.log(nameme(1));//2

var result1,result2;
    // Traditional Anonymous Function
    result1 = function (){
      return 100;
    }

    // Arrow Function
    result2 = a => a + 100;

    // Traditional Function
    function nameme (y){
      return y + 1;
    }
```

- Write down console.logs displayed from code below:

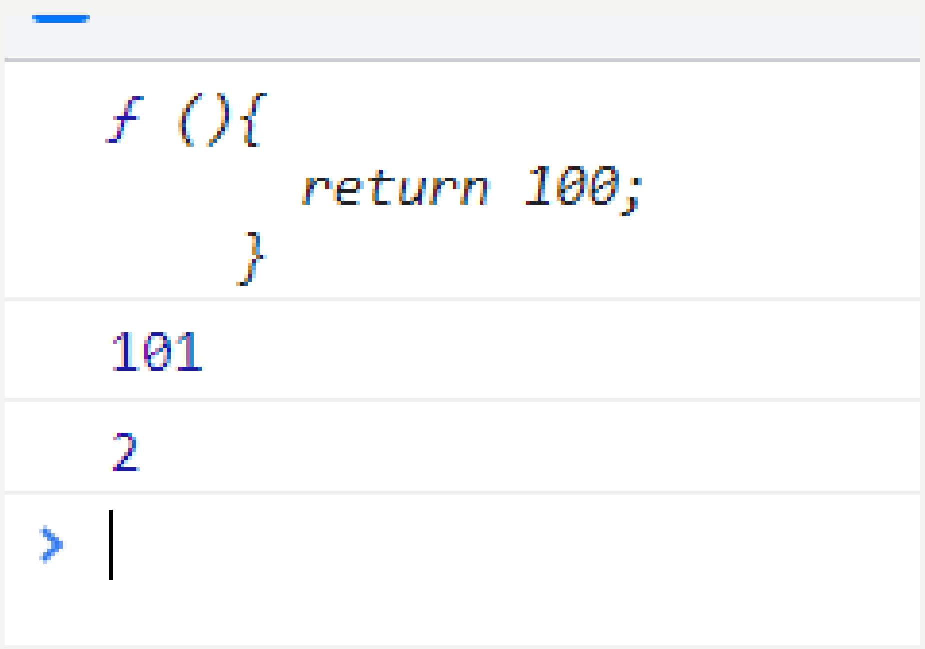


```
var result1,result2;
    // Traditional Anonymous Function
    result1 = function (){
      return 100;
    }

    // Arrow Function
    result2 = a => a + 100;

    // Traditional Function
    function nameme (y){
      return y + 1;
    }

    console.log(result1);
    console.log(result2(1));
    console.log(nameme(1));
```

- Write down console.logs displayed from code below:



```
var result1,result2;
    // Traditional Anonymous Function
    result1 = function (){
      return 100;
    }

    // Arrow Function
    result2 = a => a + 100;

    // Traditional Function
    function nameme (y){
      return y + 1;
    }

    console.log(result1);
    console.log(result2(1));
    console.log(nameme(1));
```



```
ƒ (){
        return 100;
    }
101
2
>
```

- Which built-in method ->removes and returns last element from array?

a. last()

b. shift()

c. pop()

d. None of the above.

- Which built-in method ->removes and returns last element from array?

a. last()

b. shift()

c. pop()

d. None of the above.

- What alert will show below?

- var arr = [1, 2, 3, 4, 5];

- arr.length = 2;
- alert(arr);

- What alert will show below?

- var arr = [1, 2, 3, 4, 5];

- arr.length = 2; // truncate to 2 elements
- alert(arr); // 1,2

- Note that:
- length -> it’s writable
- If we increase it manually, nothing interesting happens
- if we decrease it, the array is **truncated**

- **Try the following code**
- **arr=[1,2,3,4]**
- **arr.length=2**
- **console.log(arr);**
- **arr.length=5;**
- **console.log(arr);**

- const fruit = { name: "apple" };
- const fruitbear = { name: "apple" };

- What will comparison below return, true or false?

a) fruit == fruitbear;
b) fruit === fruitbear;

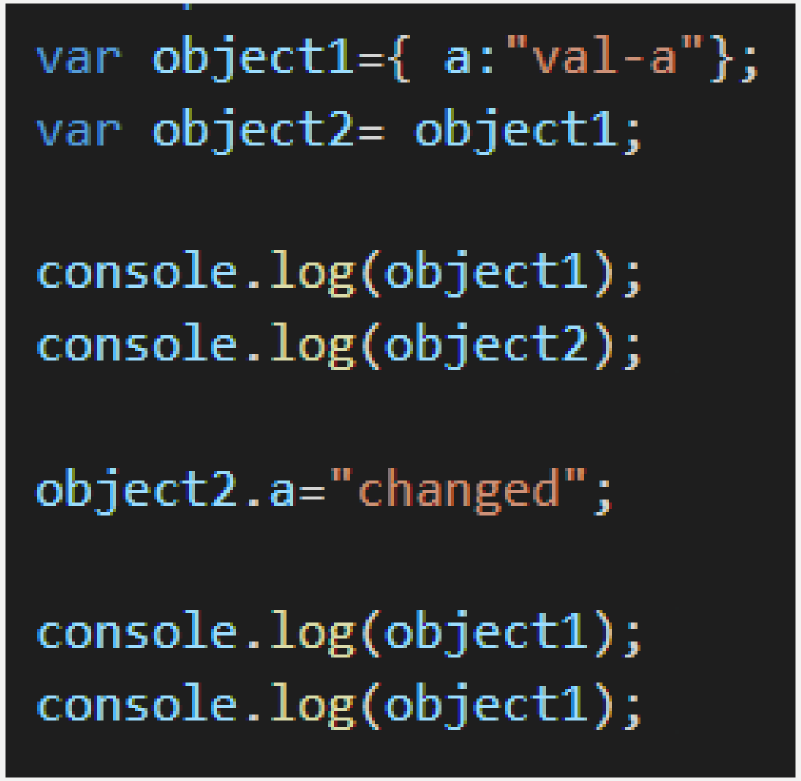
- // Two variables, two distinct objects with the same properties
- const fruit = { name: "apple" };
- const fruitbear = { name: "apple" };

- What will the comparison below return, true or false?

a) fruit == fruitbear; // return false

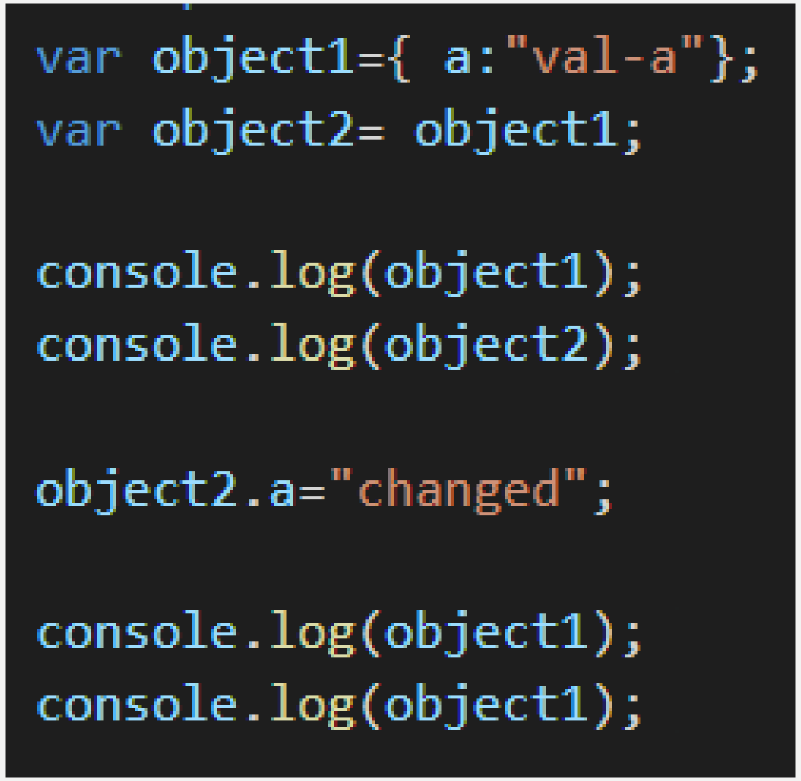
b) fruit === fruitbear; // return false

What will console.logs show below?

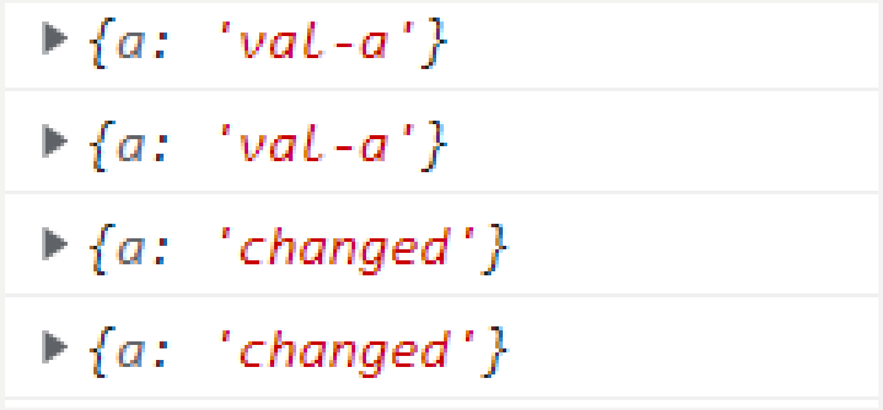


```
var object1={ a:"val-a"};
var object2= object1;

console.log(object1);
console.log(object2);

object2.a="changed";

console.log(object1);
console.log(object1);
```

What will console.logs show below?



```
var object1={ a:"val-a"};
var object2= object1;

console.log(object1);
console.log(object2);

object2.a="changed";

console.log(object1);
console.log(object1);
```

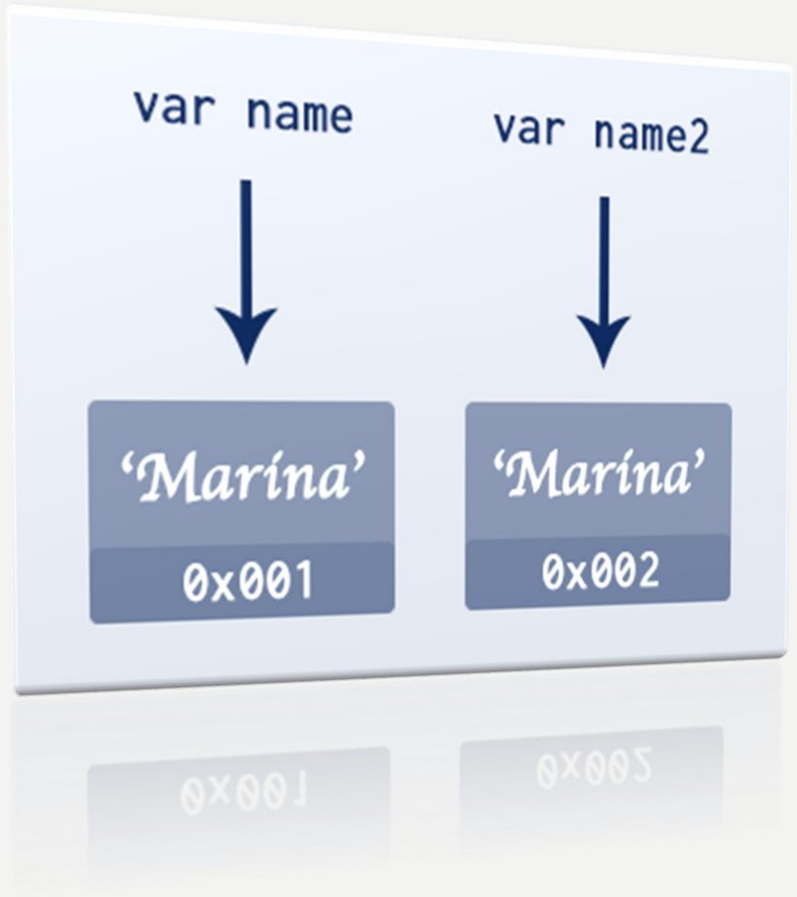


```
▶ {a: 'val-a'}
▶ {a: 'val-a'}
▶ {a: 'changed'}
▶ {a: 'changed'}
```

When we **pass** an object (or array) it is possible to modify the contents of that object.

Here a reference to *object1* is assigned to *object2*.
Think of it like the same object is accessible by two names.

- let name = ‘Marina’;let name2 = name;



- **Objects** in JavaScript **are passed by reference.**
- When more than one variable is set to store either an object, array or function, those variables **will point to the same allocated space** in the memory.

- const animals = ['Cat', 'Dog', 'Horse', 'Snake’];
- let animals2 = animals

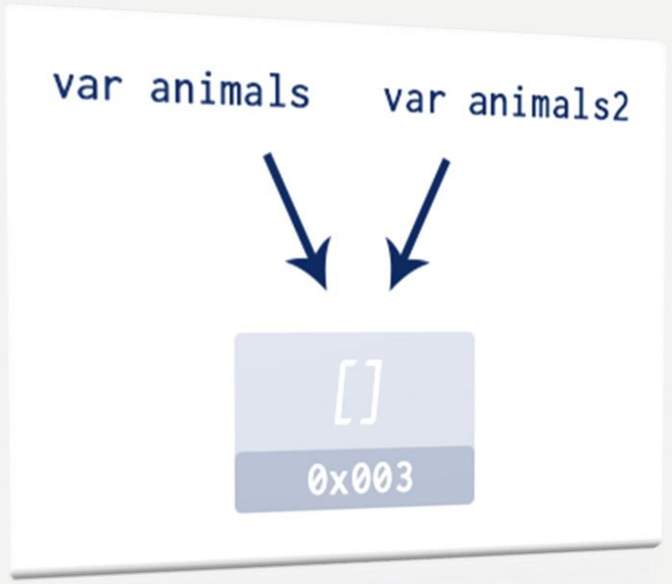
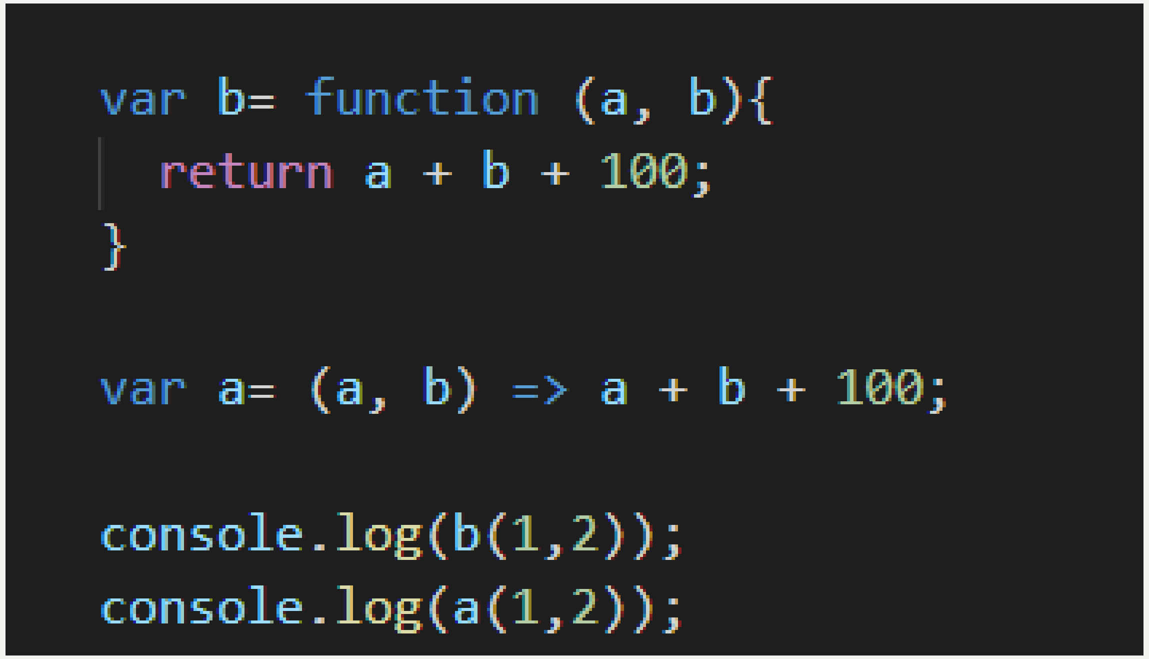


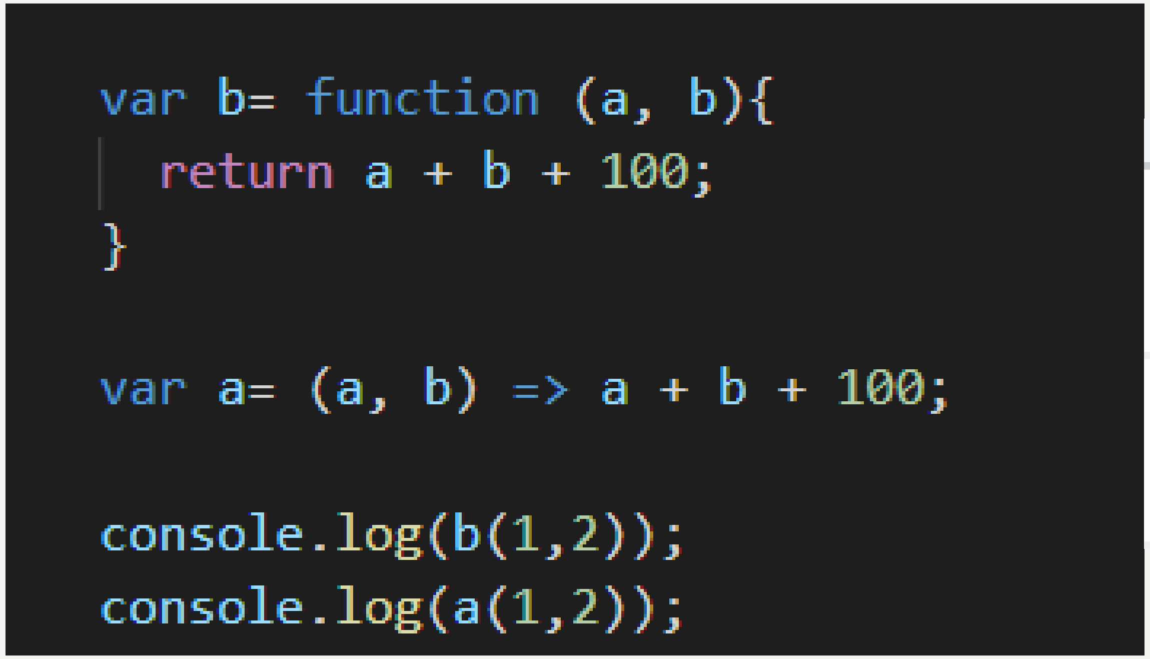
Image sources https://www.freecodecamp.org/news/how-to-get-a-grip-on-reference-vs-value-in-javascript-cba3f86da223/

- Will a and b return the same result?

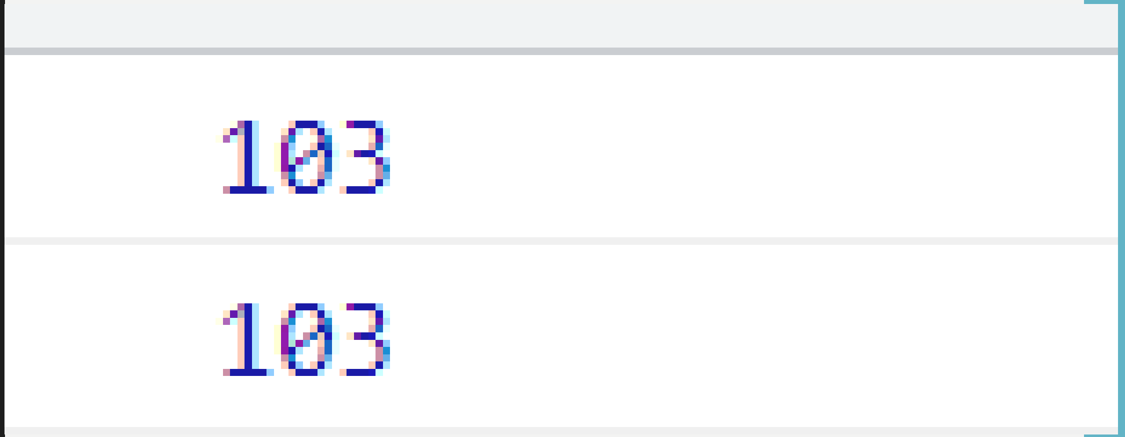


```
var b= function (a, b){
  return a + b + 100;
}

var a= (a, b) => a + b + 100;

console.log(b(1,2));
console.log(a(1,2));
```

- Will a and b return the same result? Yeap

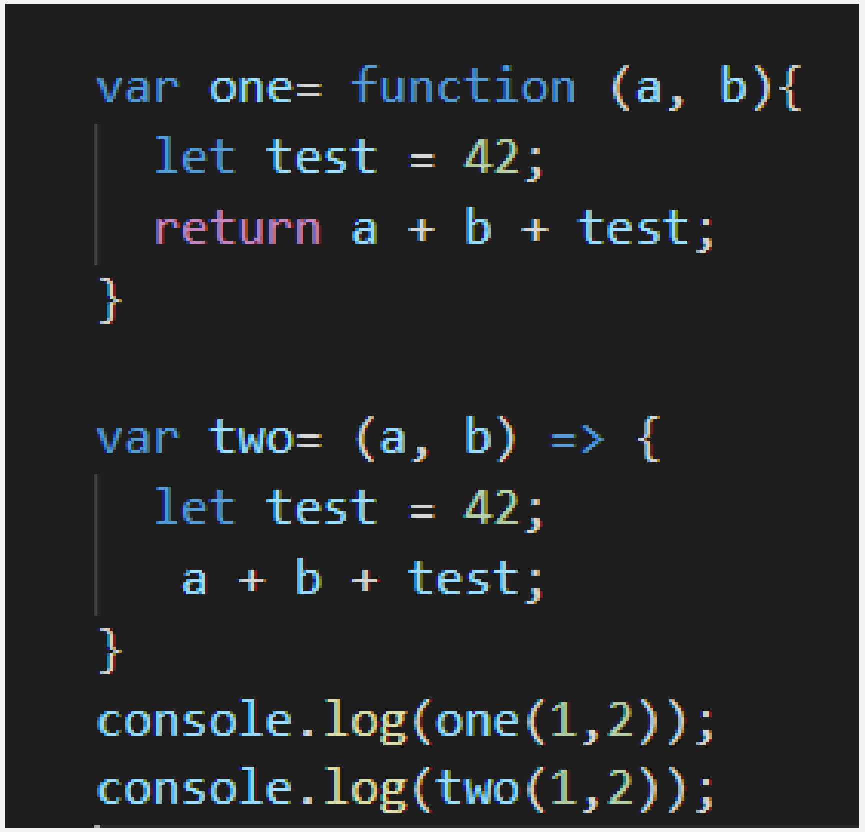


```
var b= function (a, b){
  return a + b + 100;
}

var a= (a, b) => a + b + 100;

console.log(b(1,2));
console.log(a(1,2));
```



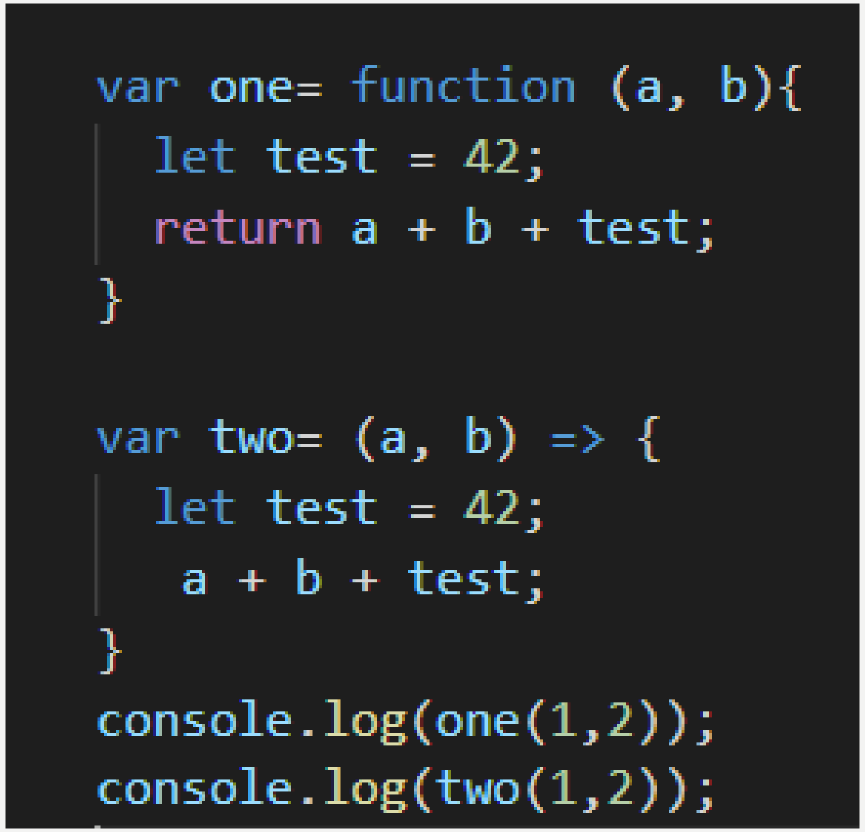
```
103
103
```

- Will one and two return the same result?

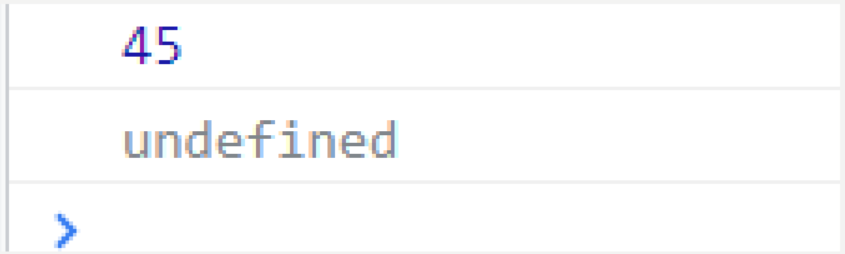


```
var one= function (a, b){
  let test = 42;
  return a + b + test;
}

var two= (a, b) => {
  let test = 42;
   a + b + test;
}
console.log(one(1,2));
console.log(two(1,2));
```

- No: we need "return" in the arrow function: it can not magically guess what we want to "return"



```
var one= function (a, b){
  let test = 42;
  return a + b + test;
}

var two= (a, b) => {
  let test = 42;
   a + b + test;
}
console.log(one(1,2));
console.log(two(1,2));
```



```
45
undefined
>
```