# Coding Questions

**Passing values by reference vs by value**

For a JS developer, it's crucially important to understand which values are passed by reference, and which ones are passed by value.

Remember that objects, including arrays are passed by reference while strings, booleans and numbers are passed by value.

### 1. What would be the output of following code?

var strA = "hi there";

var strB = strA;

strB="bye there!";

console.log (strA)

The output will 'hi there' because we're dealing with strings here. Strings are passed by value, that is, copied.

### 2. What would be the output of following code?

var objA = {prop1: 42};

var objB = objA;

objB.prop1 = 90;

console.log(objA)

The output will {prop1: 90} because we're dealing with objects here. Objects are passed by reference, that is, objA and objB point to the same object in memory.

### 3. What would be the output of following code?

var objA = {prop1: 42};

var objB = objA;

objB = {};

console.log(objA)

The output will {prop1: 42}.

When we assign objA to objB, the objB variable will point to the same object as the objB variable.

However, when we reassign objB to an empty object, we simply change where objB variable references to. This doesn't affect where objA variable references to.

### 4. What would be the output of following code?

var arrA = [0,1,2,3,4,5];

var arrB = arrA;

arrB[0]=42;

console.log(arrA)

The output will be [42,1,2,3,4,5].

Arrays are object in JavaScript and they are passed and assigned by reference. This is why both arrA and arrB point to the same array [0,1,2,3,4,5]. That's why changing the first element of the arrB will also modify arrA: it's the same array in the memory.

### 5. What would be the output of following code?

var arrA = [0,1,2,3,4,5];

var arrB = arrA.slice();

arrB[0]=42;

console.log(arrA)

The output will be [0,1,2,3,4,5].

The slice function copies all the elements of the array returning the new array. That's why arrA and arrB reference two completely different arrays.

### 6. What would be the output of following code?

var arrA = [{prop1: "value of array A!!"}, {someProp: "also value of array A!"}, 3,4,5];

var arrB = arrA;

arrB[0].prop1=42;

console.log(arrA);

The output will be [{prop1: 42}, {someProp: "also value of array A!"}, 3,4,5].

Arrays are object in JS, so both varaibles arrA and arrB point to the same array. Changing arrB[0] is the same as changing arrA[0]

### 7. What would be the output of following code?

var arrA = [{prop1: "value of array A!!"}, {someProp: "also value of array A!"},3,4,5];

var arrB = arrA.slice();

arrB[0].prop1=42;

arrB[3] = 20;

console.log(arrA);

The output will be [{prop1: 42}, {someProp: "also value of array A!"}, 3,4,5].

The slice function copies all the elements of the array returning the new array. However, it doesn't do deep copying. Instead it does shallow copying. You can imagine slice implemented like this:

function slice(arr) {

var result = [];

for (i = 0; i< arr.length; i++) {

result.push(arr[i]);

}

return result;

}

Look at the line with result.push(arr[i]). If arr[i] happens to be a number or string, it will be passed by value, in other words, copied. If arr[i] is an object, it will be passed by reference.

In case of our array arr[0] is an object {prop1: "value of array A!!"}. Only the reference to this object will be copied. This effectively means that arrays arrA and arrB share first two elements.

This is why changing the property of arrB[0] in arrB will also change the arrA[0].

## Hoisting

### 1. console.log(employeeId);

1. Some Value
2. Undefined
3. Type Error
4. ReferenceError: employeeId is not defined

Answer: 4) ReferenceError: employeeId is not defined

### 2. What would be the output of following code?

console.log(employeeId);

var employeeId = '19000';

1. Some Value
2. undefined
3. Type Error
4. ReferenceError: employeeId is not defined

Answer: 2) undefined

### 3. What would be the output of following code?

var employeeId = '1234abe';

(function(){

console.log(employeeId);

var employeeId = '122345';

})();

1. '122345'
2. undefined
3. Type Error
4. ReferenceError: employeeId is not defined

Answer: 2) undefined

### 4. What would be the output of following code?

var employeeId = '1234abe';

(function() {

console.log(employeeId);

var employeeId = '122345';

(function() {

var employeeId = 'abc1234';

}());

}());

1. '122345'
2. undefined
3. '1234abe'
4. ReferenceError: employeeId is not defined

Answer: 2) undefined

### 5. What would be the output of following code?

(function() {

console.log(typeof displayFunc);

var displayFunc = function(){

console.log("Hi I am inside displayFunc");

}

}());

1. undefined
2. function
3. 'Hi I am inside displayFunc'
4. ReferenceError: displayFunc is not defined

Answer: 1) undefined

### 6. What would be the output of following code?

var employeeId = 'abc123';

function foo(){

employeeId = '123bcd';

return;

}

foo();

console.log(employeeId);

1. undefined
2. '123bcd'
3. 'abc123'
4. ReferenceError: employeeId is not defined

Answer: 2) '123bcd'