**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]

**What is IIFE(Immediately Invoked Function Expression)?**

IIFE (Immediately Invoked Function Expression) is a JavaScript function that runs as soon as it is defined.   
The signature of it would be as below,

(function ()

{

// logic here

}

)

();

The primary reason to use an **IIFE is to obtain data privacy because any variables declared within the IIFE cannot be accessed by the outside world**.

i.e, If you try to access variables with IIFE then it throws an error as below,

(function ()

{

var message = "IIFE";

console.log(message);

}

)

();

console.log(message); //Error: message is not defined