**Differences between Copy by Value and Copy by Reference**

**Copy by Value:**

When a primitive data type, namely Number ,String ,Boolean (also undefined and null) is initialised to a variable, the **value** of it is copied into the memory location of the variable.

For eg:

*let a =25;*

*let b= “Hello”;*

*let c= true;*

*console.log(a,b,c); // Output will be 25 Hello true*

*let x= a;*

*let y= b;*

*let z= c;*

*a=55;*

*b= “World”;*

*c= false*

*console.log(x,y,z) ; //Output will be 25 Hello true*

Changing value of **a** ,will not change the value of **x** and likewise for **b** and **c**, because it is the **value** of **a** which is copied,not the address of **a.** This is called Copy by Value

**Copy by Reference**

When a variable is initialised to a **non-primitive data type** , namely **Object** or an **Array** ,the **reference** to this value is stored in the variable. i.e. the variable will hold the address of the memory location of the Object or array, rather than the value itself.

For eg:

*let a= {name:”Balaji”, age:24};*

*let b= [1,2,3];*

*let x =a;*

*let y=b;*

*console.log(x,y); // {name:”Balaji” , age:24} [1,2,3]*

*a= {name:”Ganesh”, age:42};*

*b= [4,5,6];*

*console.log(x,y); //{name:”Ganesh”, age:42} [4,5,6]*

**What happens?**

1.Let the address of the object *{name:”Balaji”, age:24}* be #888 . A **reference** to what is at #888is stored in **a**. So now **a** contains whatever is at address #888.

2. Likewise, let the address of the array [1,2,3] be #555 .**b** holds **reference** to whatever is at memory location #555.

3. Since **x** is initialised to **a**, **x** will hold a reference to **a** . So value of x now points to *{name:”Balaji”, age:24}*

4.Likewise **y** is initialised to **b.** So **y** holds a reference to **b.** Thus,yalso points to [4,5,6]

5.Now **a** is re-initialised to *{name:”Ganesh”, age:42} .*Now **a** will hold reference to this object.

6.Likewise **b** is re-initialised to [4,5,6]. Now **b** will hold reference to this array.

7.Now **x** and **y** also would point to the newly initialised values, because they are **called by reference**