JavaScript: Primitives/References

In JS, there are primitive data types and reference data types.

Primitive data types are referenced by **value** while the non-primitive/reference data types point to **memory addresses.**

Primitive data types are:

* Boolean
* Number
* String
* Null
* Undefined
* Symbol

Reference data types are:

* Objects
* Arrays

When we write a primitive data type like this:

let one = 1;

On the call stack, the one variable directly points to the value 1:

Call Stack  
#000 one -> | 1 |  
#001 | |  
#002 | |  
#003 | |

And if we change the value of the one variable

let one = 1  
one = 3

The one memory address #000 which holds the value 1 is directly changed to 3.

But, if we write a reference data type like this:

let arr = {  
 one: 1  
}

OR

let arr = new Object()  
arr.one = 1

JS will create the object in the Heap memory and,

store the memory address of the object in arr call stack location:

Call Stack Heap  
#000 arr -> | #101 | #101 | one: 1 |  
#001 | | #102 | |  
#002 | | #103 | |  
#003 | | #104 | |

You see arr doesn't store the object directly but points to the memory location(#101) of the object. Unlike primitive data types that hold its value directly.

let arr = { one: 1 }  
// arr holds the memory location of the object {one: 1}  
// `arr` == #101

let one = 1;  
// `one` a primitive data type holds the value `1`  
// one == 1

If we change the property in arr like this:

arr.one = 2

We are basically telling the program to change the property **value** of the object pointed to by arr.

***Because reference data types hold memory address any change to it affects the memory it points to.***

If we passed in a primitive data type:

function chg(arg) {  
 arg++  
}

let one = 1; // primitive data types holds the actual value of the variable.

log(one) // 1

chg(one /\* 1 \*/)  
// the value of `one` is passed in.

log(one) // one is still `1`. No change because primitives only hold the value

Similarly, if we pass the property of an object instead of object itself, it will be the value of that property which is passed to the function,

function chg(arg) {

arg++;

}

let obj = {a: 1};

console.log(obj) // {a: 1}

chg(obj.a) // here we are passing the ACTUAL VALUE stored for key “a”

console.log(obj) //{a: 1}

}

In simple terms,

Passing reference – means passing the address.

Passing value – means passing the value, which might be stored at some address in memory.