**Data types and its way of copying of the values**

**DATATYPES-**

Datatypes in javascript they are categorized into following classification they are,

1. Primitive -number, strings, Boolean and bigint.
2. Trivial -null and undefined.
3. Composite -arrays, functions and objects

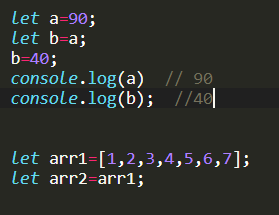
We are now going to see how they are copied when they are assigned to a new variable or passed into the function. We have two major types that are to be explained.

1. Copy be value –

We have primitive datatypes following copy by value where as composite datatypes follow copy by value.

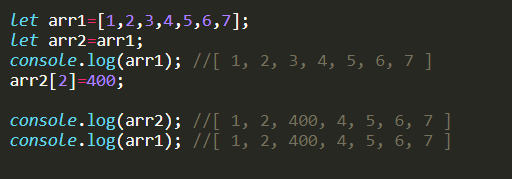
When a variable is assigned with a other variable whose data type is primitive, then the value of the variable assigned is as same as the value of the passed variable. There is a new space is created in the variable name and the value of passed variable is stored in the space.

This by which any changes made to the new variable will only affect the new variable and not the passed one. The value of the passed variable remains un changed as it was only copy beign passed and copied to new variable and not its whole address location.



1. Copy by reference-

When we see if the variable of data type array or object of an composite data type is passed to the new variable then only the reference of the old variable is copied and not the values of that old variable.



**Reason for copy by reference over copy by value-**

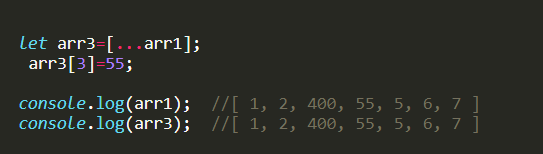
This happens in a way that both of the variables pointing to same location in memory and changes in any one of them affect each other. This type of copy by reference is adapted for composite data types since they occupy a large memory space and when there is a need of copy of them copying by value might take larger memory space and tedious time consumption in realtime.

Hence javascript adapts a shallow copy method by which only reference of the location of the objects in memory is copied.

**Copy by value can be applied on composite datatypes-**

We could do a copy by value operation on the objects when we need to retain the old variable and manipulate only their copies for certain functionalities so we go for the following methods.

* **Using spread operator-**

****

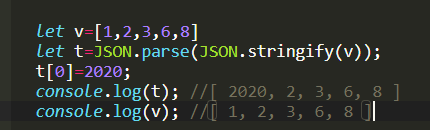
While we use spread operator we actually iterate the whole array into it we could change the values but when we use shift opration the copied array changes the main old array. So it does not hold good for all operations.

* **Using concatenation method - concat()**

When we use concatenation with other array and change the new array the old array values do not get affected and it holds good for all operatons.

* **Using slice method – slice()**
* **We could use for or while loop to iterate through the length of the array and have it copie by each value.**
* **Using for each**
* **Using object.assign() method**
* **Using map() method-**
* **Using JSON.parse(JSON.stringiy()) method-**

Stingify the value and then parse the old array to form proper JSON object by which we could copy by value of the whole array.

****

There are also so many other ways to iterate through objects unlike array and also perform copy by value some of the conventional ways are discussed here.