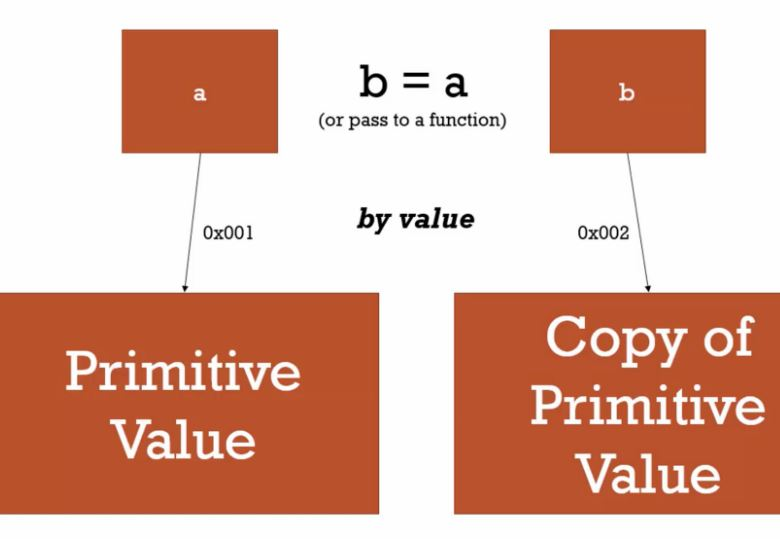
**TASK 2**

Difference between copy by value and copy by reference

**COPY BY VALUE:**

In javascript copy by value passes the primitives like undefined, null, boolean, string, and numbers. When we assign a value to the variable ‘a’ using equals to operator, it assumes that the value is a primitive type and setup address in the memory. It stores information and points to a variable in that memory. After that when you create another variable ‘b’ and assign ‘a’ to ‘b’, then ‘b’ creates another location in the memory and copies the value of ‘a’.



So, by value copies the value of the original variable (a) into two separate spots in memory. Being unaware of this can cause some interesting interactions, let us take a look below:

Let a=5;

Let b=a;

Console.log(a); // prints 5

Console.log(b); // prints 5

Let a=1;

Let b=a;

Console.log(a); // prints 1

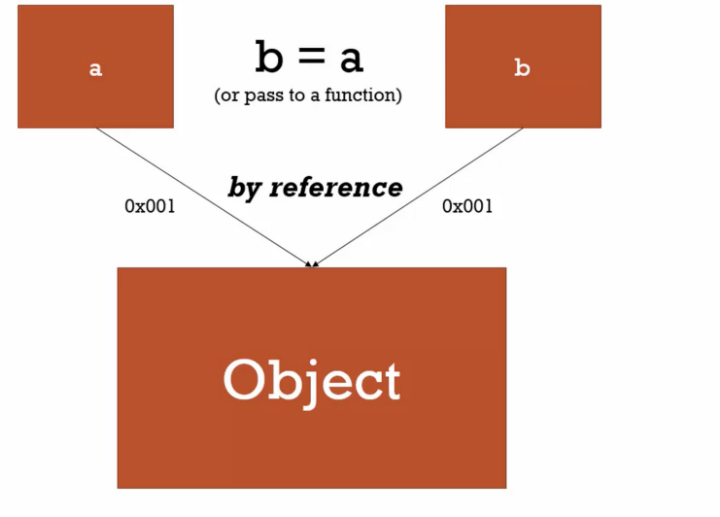
Console.log(b); // prints 5

So if you change the value to ‘a’, it prints the changed value whereas the ‘b’ value is not changed as it still points to the same location that assigned in the memory. Because ‘b’ creates its own spot in the memory.

In sum, by value copies the value into two separate spots in memory effectively making them entirely separate entities despite one initially being set equal to the other.

**Copy by reference**:

In copy by reference ,passing by reference relates to objects in JS including functions. When we assign an object to a variable ‘a’, equals operator understands that it is an object and creates a location(address) in the memory. After that we create another variable ‘b’ and assign ‘a’ to it. Then the equals operator finds that it is an object so it won’t create any address and it points the same address as ‘a’.



|  |
| --- |
|  |
| Let | a={language:”Javascript”};  let b = a |
|  |  |
|  | console.log(a) ; // => {language: "Javascript"} |
|  | console.log(b) ; //=> {language: "Javascript"} |
|  |  |
|  | a.language = "Ruby"; |
|  |  |
|  | console.log(a); // => {language: "Ruby"} |
|  | console.log(b) ; // => {language: "Ruby"} |

In above example we assigned an object to the variable ‘a’ so the location is created in the memory and ‘a’ assigned to the variable ‘b’. Both prints the same in this case as b also points to the same location assuming it as an object. So, when I mutate the the value of the key in variable (a) (in this case changing “Javascript” to “Ruby”) when I console.log the results 0f (a) and (b) they are the same since (b) points to the same location that (a) does.

**Copy by value a composite data type in JavaScript?**

Composite data types are objects, arrays and functions in javascript. In composite data types we cannot clone the original value as objects are passed through reference. It points to the same location whenever we change the value its original value also changed. So, to do that we use

1. Using spread operator (…):The spread operator (…) does a shallow copy of the variable data. It copies the values present that is the nonprimitive data type.

Example:

var a =[2,5,6];

console.log(a); //[2,5,6]

var b=[…a];

var a[0]=1;

console.log(a); //[1,5,6]

console.log(b); //[2,5,6]

In above case b vale is same as the original value even if it is changed.

2)Using JSON.parse() and JSON.stringify() methods: The JSON object is available in all the browsers, this has two methods for JSON data i.e the parse and the stringify. Parse takes a JSON string and transforms it into JS Object and Stringify takes JS Object and transforms into a JSON Object.

Example:

var a=[3,4,5];

var b=JSON.parse(JSON.stringify(a));

console.log(a); //[3,4,5]

var a[0]=20;

console.log(a); //[20,4,5]

console.log(b); //[3,4,5]

3)Using Object.assign():The Object.assign() method copies all properties from one source object to a target object. It returns the target object. This does the shallow copy of the original object.

Example:

var a=[1,4,5];

var b= Object.assign( [] ,a);

console.log(a); // [1,4,5]

var a[0]=6;

console.log(a); //[6,4,5]

console.log(b); //[1,4,5]

In above example, b copies the original value even if it is changed.

**Html and script.js file and run a for loop on the data and print all the country names in the console.**

**Try the rest countries api. Extract and print the total population of all the countries in the console. use the html template.**

