## Copy by Value vs Copy by Reference

When a primitive data type is defined it is stored in a location in memory which is labeled with the data type name. So two variables with equal values are two different entities in memory. Changing the value of one variable will not affect the value of the other. This is known as copy by value.

On the other hand when a non-primitive data type like an object or an array is defined it creates a label with the data type name which stores the location of the value of the object. So the data type name doesn’t store the value of the object rather it stores the memory address of the value.

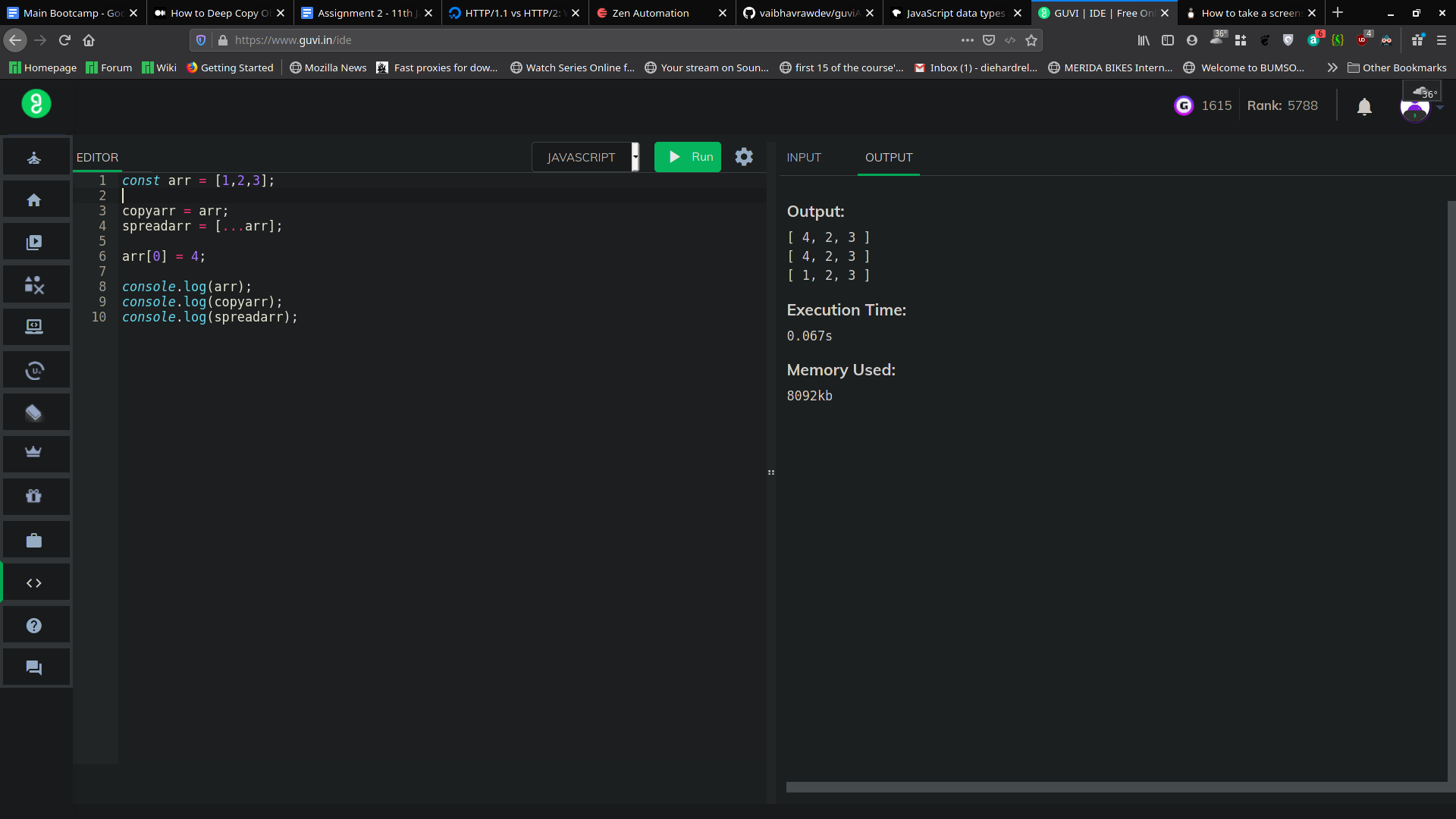
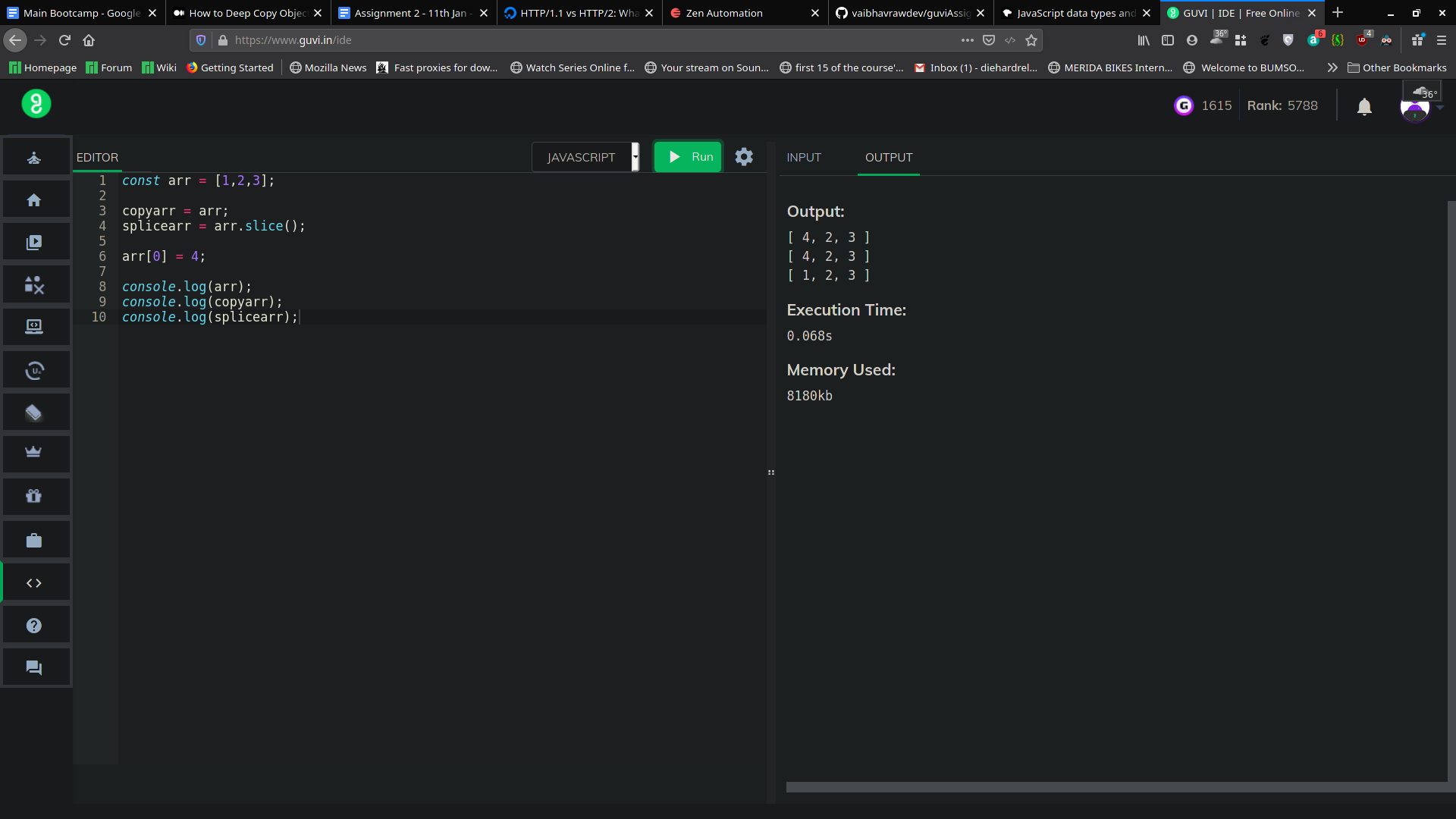
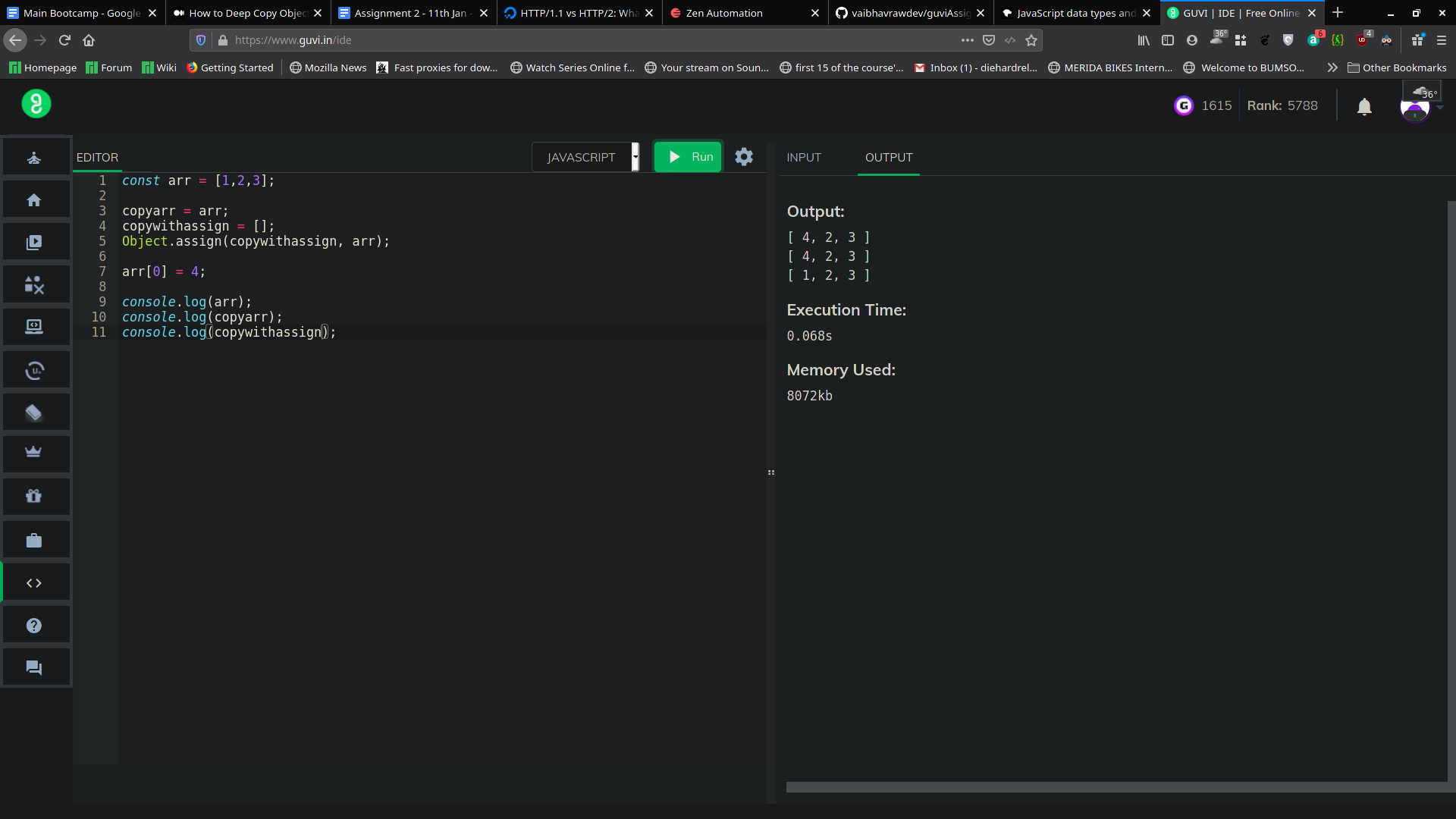
Thereby when an object is copied to create another, they’re pointing to the same value in memory, which in turn means that changing one object will also change the other. This is known as copy by reference.

## Copy by Value of Composite Data Types

Composite Data Types are data types built from primitive data types and other composite data types. Composite data types can hold collections of values and more complex entities.

For composite data types, making a normal copy with the assignment operator makes a copy by reference by default.

So to circumvent this and to use copy by value for composite data types, the following methods are used:

1. The Spread Operator(...) - 
2. Slice Method (.slice) - 
3. Copy by Assign (Object.assign()) - 
4. Copy by Array.from() - 