**Stack Heap**

username: “Babita",

useremail:"userone2gamil.com",

userTwo

userOne

myWhatsname

anotherName

myWhatsname

**Stack**

**1.let myWhatsappname = "FurryBoy";:**

This line creates a variable called myWhatsappname and assigns it the string value "FurryBoy".

So, myWhatsappname now holds "FurryBoy" in memory.

**2.let anothername = myWhatsappname;**:

* + This creates a new variable called anothername.
  + anothername is assigned the **value** of myWhatsappname, which is "FurryBoy".
  + Since strings are **primitive values**, JavaScript makes a **copy** of "FurryBoy" and stores it in anothername. So, now we have:
    - myWhatsappname holding "FurryBoy"
    - anothername also holding a separate "FurryBoy".

1. **console.log(anothername); // FurryBoy**:
   * This line prints the value of anothername, which is "FurryBoy".
   * Since we haven’t changed anothername yet, it still holds the initial copied value "FurryBoy".

**anothername = "Coco";**

* Here, we assign a new value, "Coco", to anothername.
* This **only changes anothername** and doesn’t affect myWhatsappname.
* Now we have:
  + myWhatsappname still holding "FurryBoy"
  + anothername now holding "Coco"

Heap

In the code, userone is an object with properties username and useremail. When you assign usertwo = userone, both variables refer to the same object in memory. So, when you change usertwo. useremail, it also changes userone. useremail because they point to the same object.

Both userone and usertwo will show the same email (usertwo@gmail.com) because they are referring to the same object.