let obj = {

name: "Rohit",
  age:30,
  amount:420,
}

// const {name , amount} = obj;
// console.log(name,amount);
// Homework pass by value or pass by reference

When you use object destructuring in JavaScript like this:

```
const obj = {
    name: "Rohit",
    age: 30,
    amount: 420,
};

const { name, amount } = obj;
console.log(name, amount);
```

- **Destructuring on primitive values:** For primitive types (like strings, numbers, booleans), destructuring creates a *copy* of the value, not a reference.
- **Destructuring on objects or arrays:** If you destructure a property that is itself an object or array, the *reference* to that object/array is copied (not the object's content). Any changes you make to the referenced object or array *will affect* the original object.[1] [2]

For your example:

- `name` and `amount` are strings and numbers (primitives). So the variables `name` and `amount` get *copied* values, not references.
- If your object had a property that was itself an object (like `address: { city: "Pune" }`), destructuring `const { address } = obj;` would copy the *reference* to the `address` object.

Summary Table

Type	Destructured Variable	Pass by Value	Pass by Reference
Primitive (number, string, boolean)	Yes	✔	
Object/Array	Reference copied		✔

Key point:

- Primitives are always copied by value.
- Objects/arrays are copied by reference (their reference is copied, not deep-cloned; so changing their content changes the original).[2] [3] [1]

If you only extract properties like `name` and `amount` (both primitives), changing `name` or `amount` after destructuring does *not* affect the original object. If you destructure an object or array property, changes to that property via the new variable *will* affect the original object.

Example with an object property:

```
const obj = { data: { x: 1 } };
const { data } = obj;
data.x = 2;
console.log(obj.data.x); // 2 (changed)
```

Example with a primitive property:

```
const obj = { val: 5 };
const { val } = obj;
val = 10;
console.log(obj.val); // 5 (unchanged)
```

So, *destructuring itself does not mutate the original object*. The rules about "pass by value" or "pass by reference" follow normal JavaScript assignment rules based on the property type.[3] [1] [2]

: Destructuring assigns references for objects/arrays and copies for primitive values.[1] [2]
: Destructuring does not change (mutate) the original object.[3]

⁂

1. https://stackoverflow.com/questions/38797417/are-any-aspects-of-object-destructuring-assignments-by-reference
2. https://lucamezzalira.com/2015/09/25/es2015-destructuring-assignment-object-and-array/
3. https://www.w3schools.com/js/js_destructuring.asp