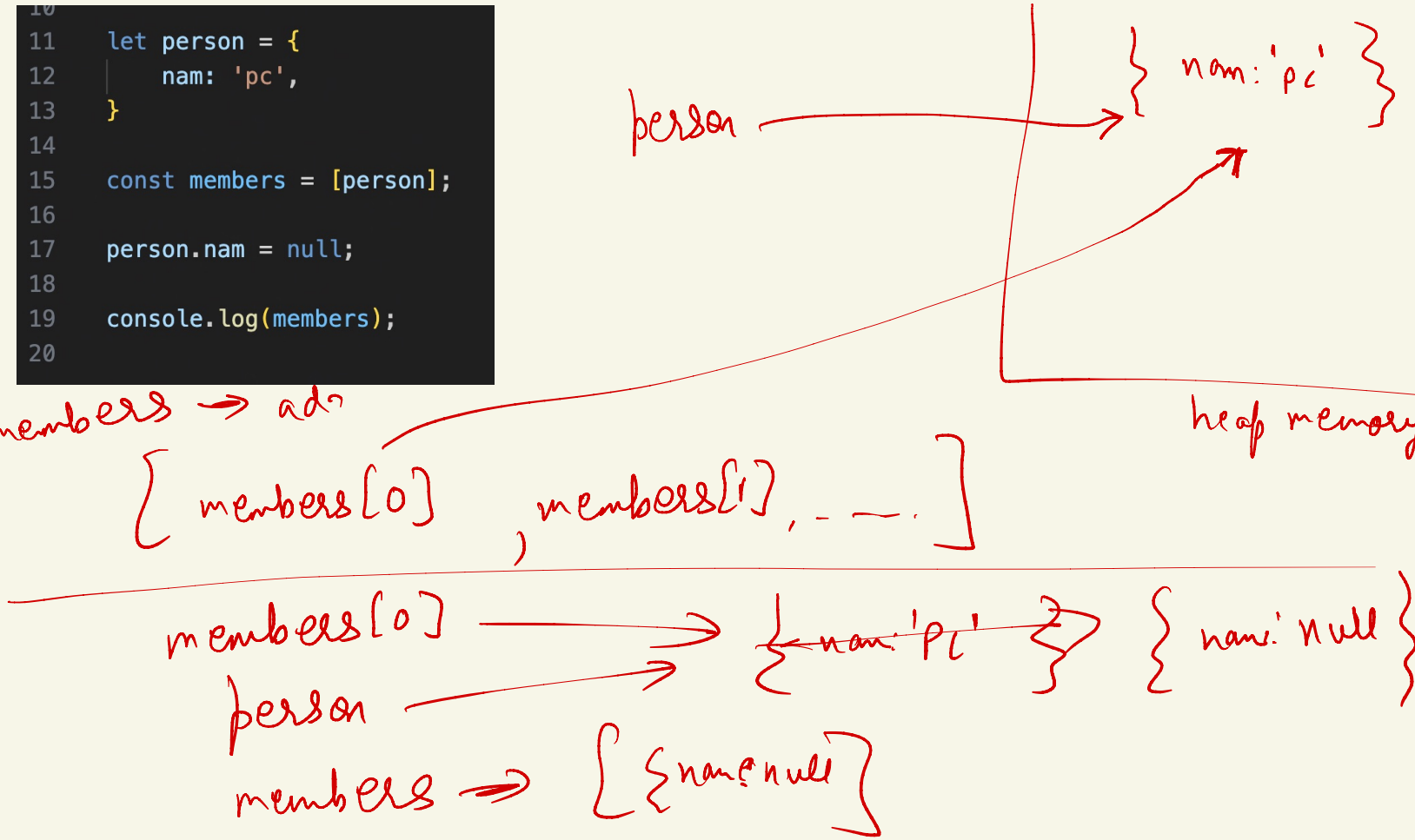
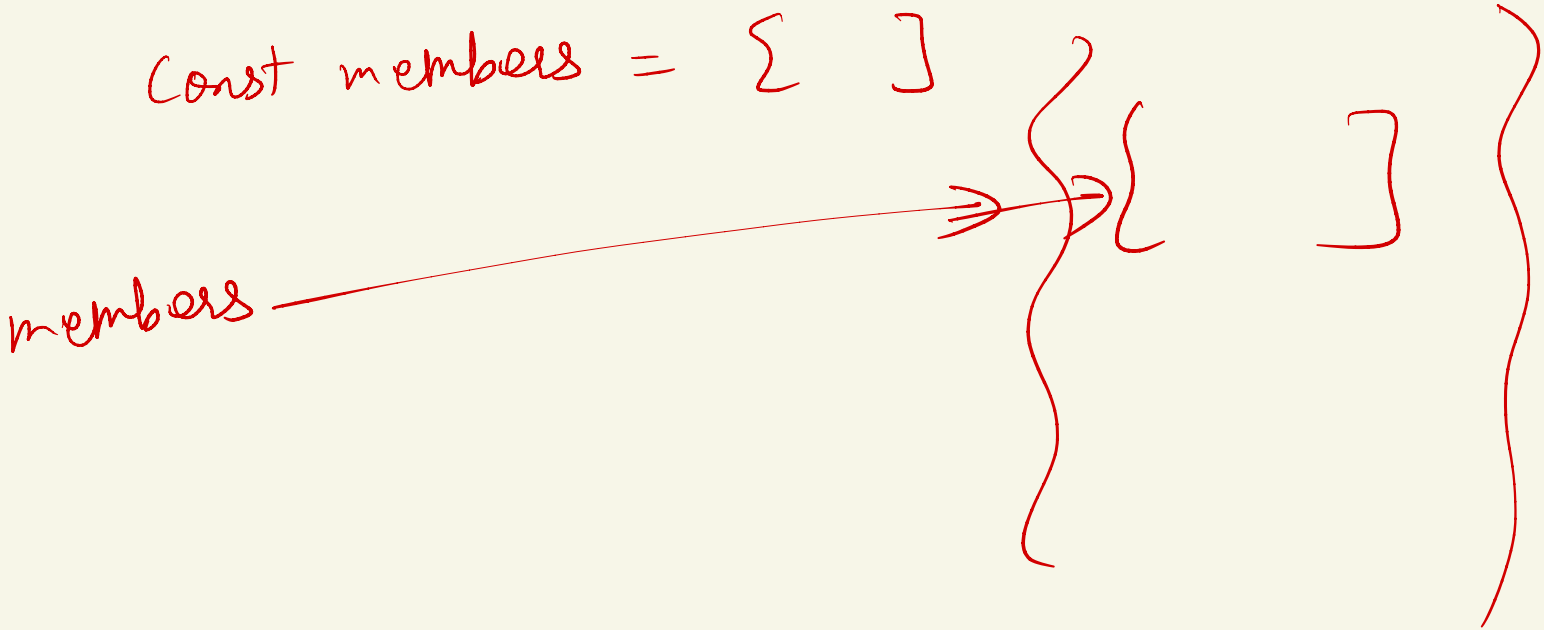
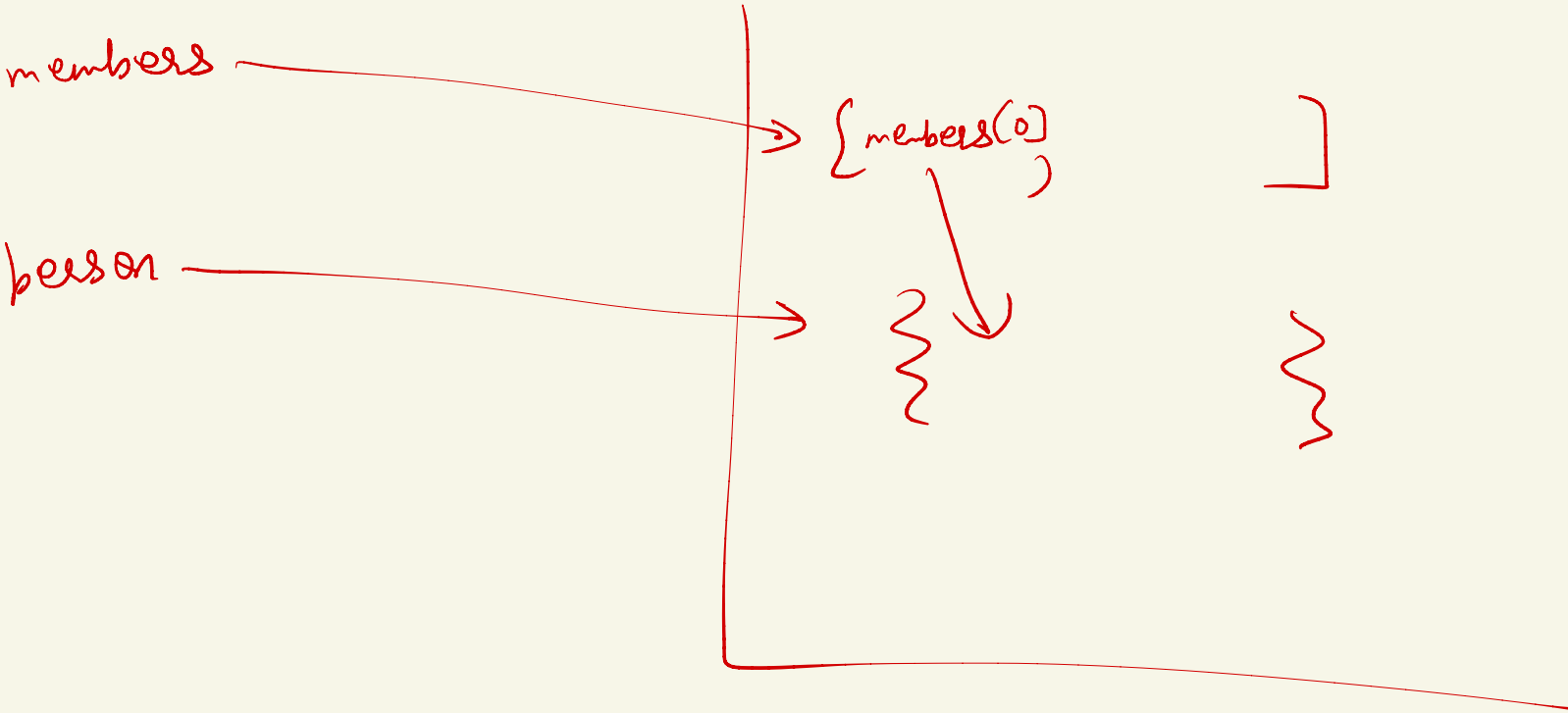
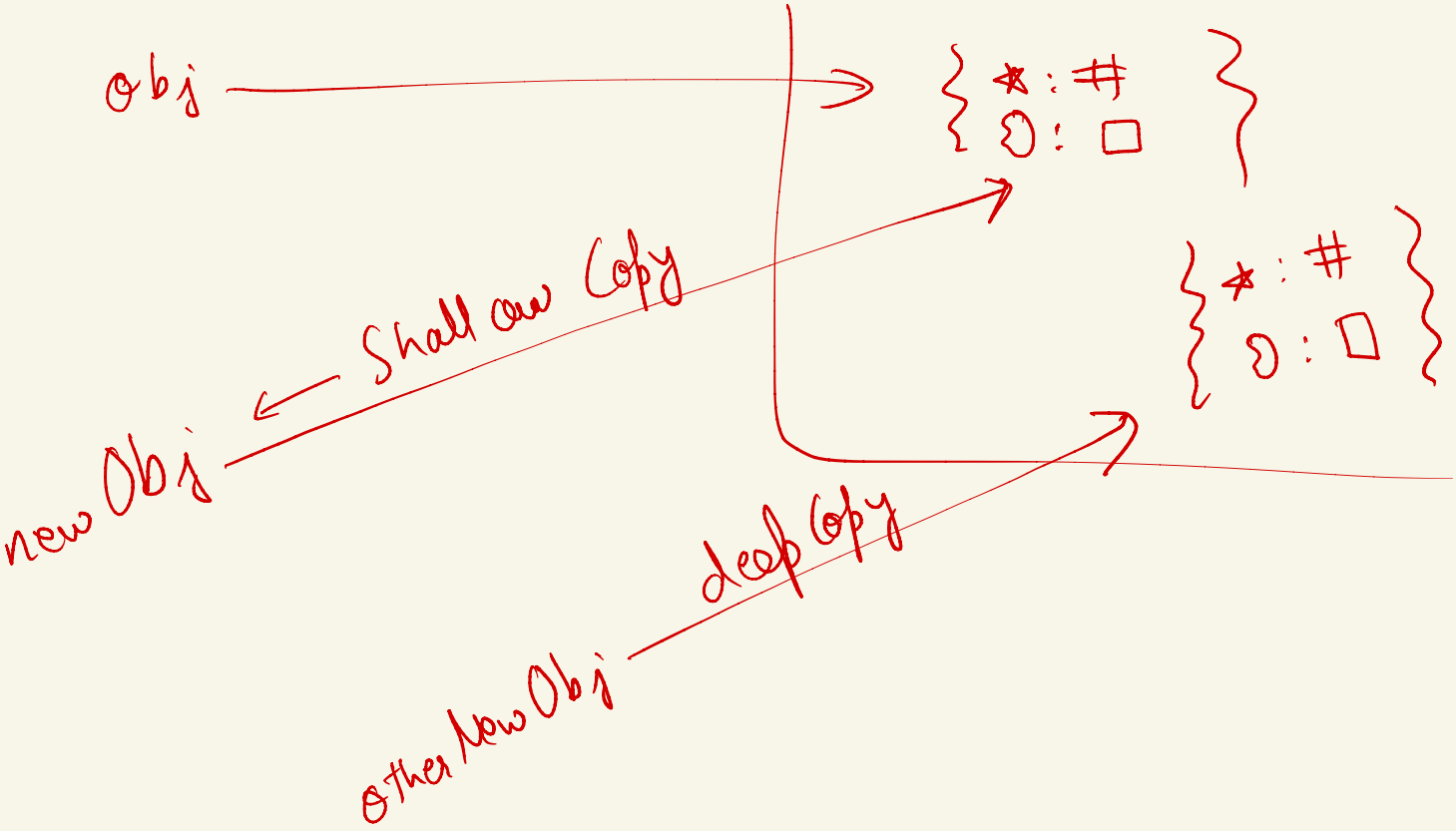
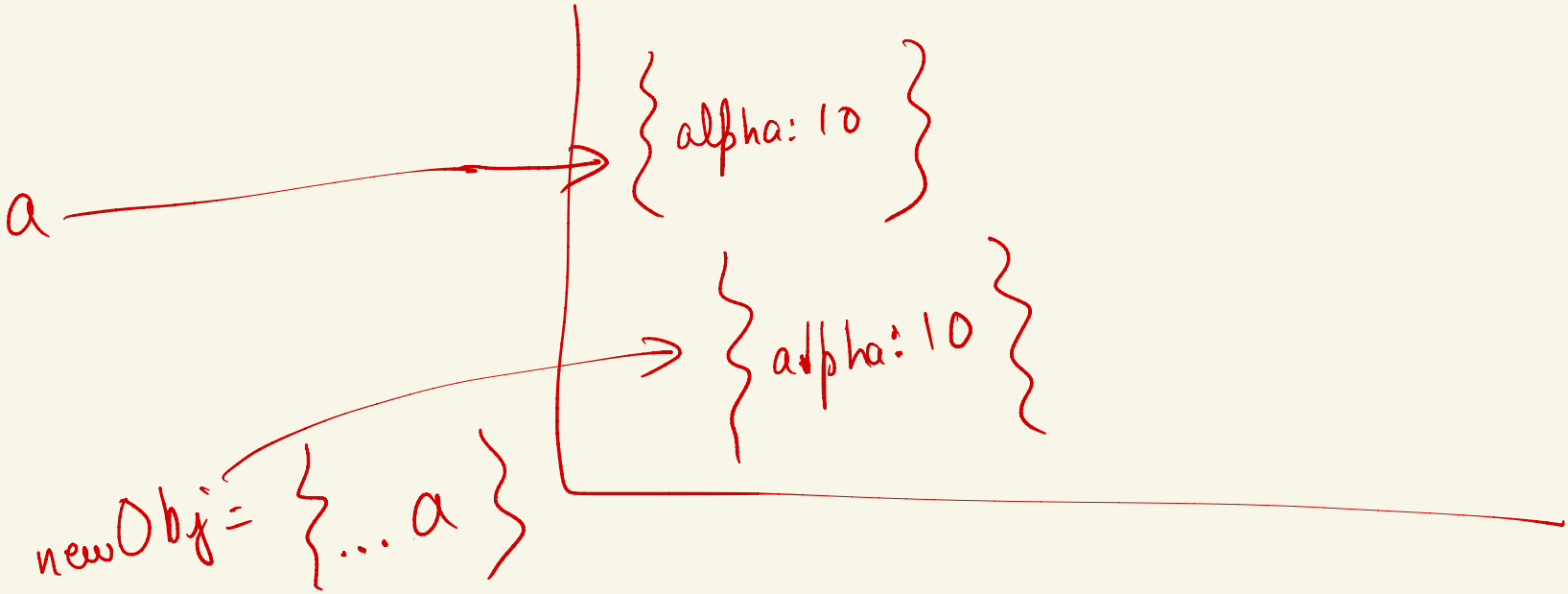
```
11  let person = {
12      nam: 'pc',
13  }
14
15  const members = [person];
16
17  person.nam = null;
18
19  console.log(members);
20
```

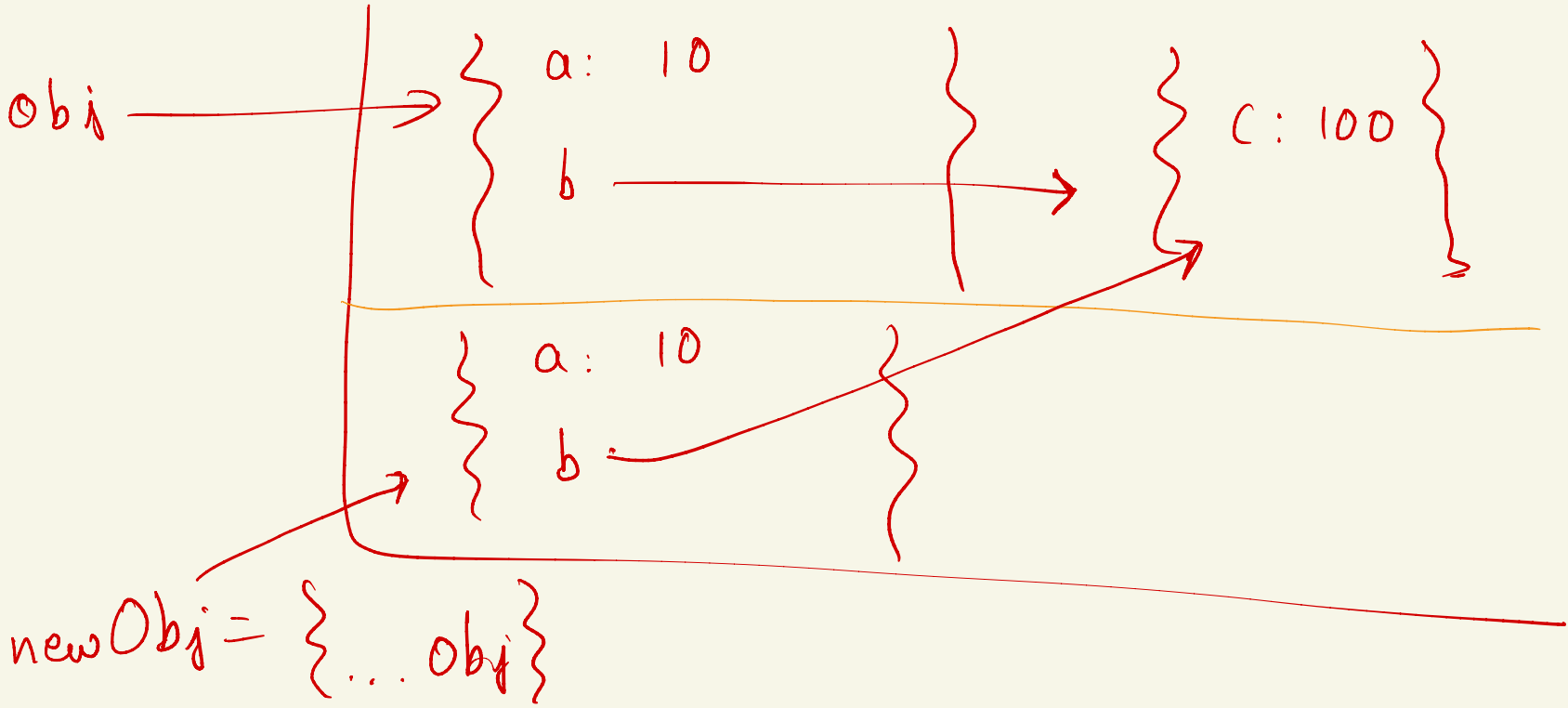


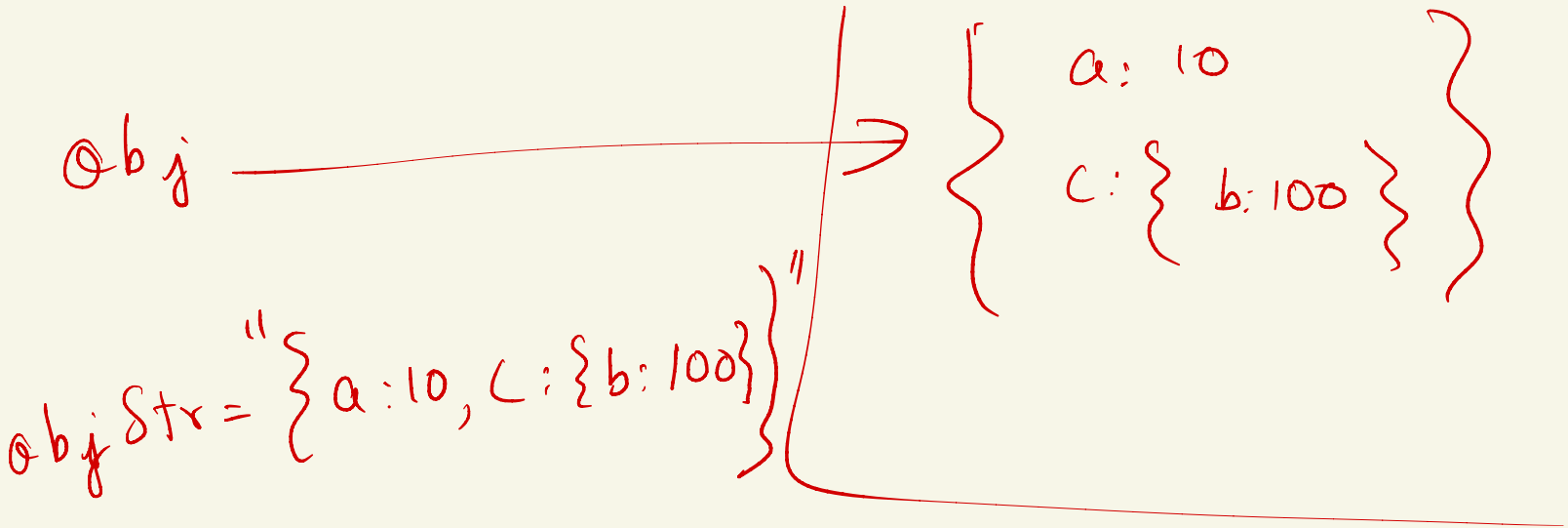

Const members = [ ]
members


members
person
members[0]


obj
new Obj
Shallow Copy
other New Obj
deep Copy

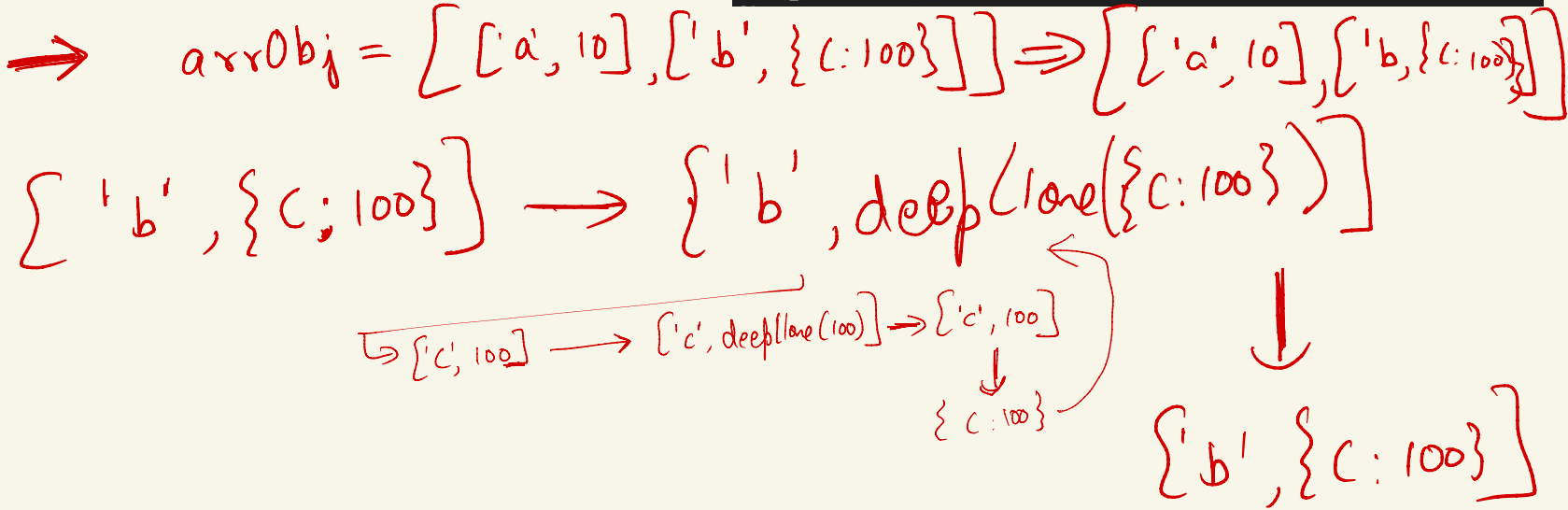

a
{ alpha: 10 }
{ alpha: 10 }
newObj = { ... a }


obj
a: 10
b
c: 100
a: 10
b
newObj = {...obj}


obj
{ a: 10
c: { b: 100 } }
objStr = "{a:10, c:{b:100}}"

```
35
36  let obj1 = {
37      a:10,
38      b:{
39          c:100,
40      }
41  }
```

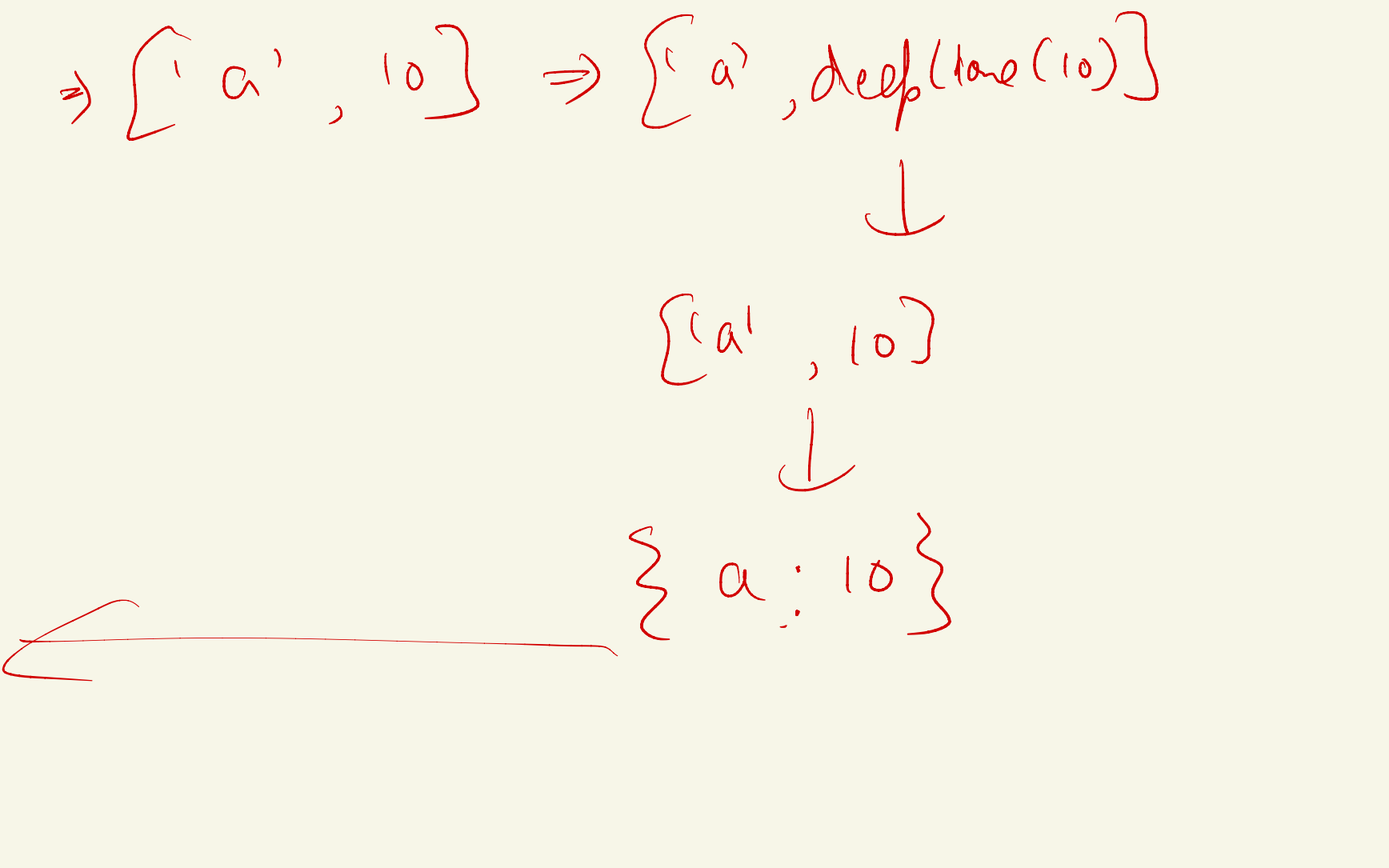
```
5   const deepClone = (obj) => {
6       const type = typeof obj;
7       // i don't want to proceed with
8       // 'null', 'undefined' numbers & string
9       if (type !== 'object' || !obj) return obj;
10
11      // this is also array
12      let arrObj = Object.entries(obj);
13
14      // this is array
15      let deepCloneArrOfObj = arrObj.map(([key, value]) => [key, deepClone(value)]);
16
17      let finalObj = Object.fromEntries(deepCloneArrOfObj);
18      return finalObj;
19      // returing deepCloned version of obj
20  }
```



obj = [{a: 10}, 'abc', 123];

=> [deepClone({a: 10}), deepClone('abc'), deepClone(123)]

[{a: 10}, abc, 123]


⇒ ['a', 10] ⇒ ['a', deep(one(10)]
['a', 10]
{a: 10}