

# Array notation explanation



```
const users = [
  {name: "A", age: 20},
  {name: "B", age: 25},
  {name: "C", age: 20},
];

const result = users?.reduce((acc, crnt) => {
  if(acc[crnt.age]) {
    acc[crnt.age] = [...acc[crnt.age], crnt.name]
  } else {
    acc[crnt.age] = [crnt.name]
  }
  return acc;
}, {})

console.log("result-->>", result);
console.log("users:", users);

in this explain me one thing that
acc[crnt.age] = [crnt.name]

by doing this way:
[crnt.name]
```

Let's explain **exactly what [crnt.name] means** in your code in a simple, beginner-friendly way.

## ✓ Your code

```
js

acc[crnt.age] = [crnt
```



my question is that by this i am assigning the reference to new array of name or copying independently it to it and later if the name value change it won't effect our one

Great question — and **you are thinking exactly like a good JavaScript developer now.**  
Let's clear this completely.

## ✓ Short Answer

[`crnt.name`] **creates a brand-new array**, and inside it, it stores the *value* of `crnt.name`.

So:

- It **does NOT store reference of an existing array** → it is a **new array**.
- It **does NOT link to the original object** → changing `crnt.name` later **will NOT change** the array value.
- It **stores the current value**, not a live reference.

## ✓ Deep Explanation (Very Simple)

### ★ When you do:

```
js
acc[crnt.age] = [crnt.name];
```

### What happens?

- A **brand new array** is created → e.g. `["A"]`
- Inside that new array, the **value "A"** is copied.
- There is **no reference** to `crnt` object here.

## 💡 Proof Example

Look at this:

```
js
const user = { name: "A" };
const arr = [user.name]; // stores "A"

user.name = "Z"; // change original object's name
console.log(arr); // still ["A"]
```

### Output:

```
css
["A"]
```

Because `[user.name]` stored the value "A", not a reference to `user.name`.

## ! When would reference happen?

Only if you do this:

```
js  
const arr = [user]; // user object reference stored
```

Now if you change `user.name`, it **will** reflect inside the array.

But in your case you are doing:

```
js  
[user.name] // primitive value (string), not reference
```

So no link exists.

## ✓ Final Conclusion

`acc[crnt.age] = [crnt.name]:`

- ✓ Creates a **new independent array**
- ✓ Stores **current value of crnt.name**
- ✓ Changing `crnt.name` later **will NOT affect** the stored value
- ✓ No reference to original object

If you want, I can also explain this using memory diagrams (boxes & arrows style) to make it 1000% clear.