but copy and original addressin memory what about that? where array store in js in stack or heap?

Excellent question, Yasser 👨‍💻—this dives into **JavaScript's memory model**, and it's crucial for understanding copy behavior.

🧠 Where Are Arrays Stored in JavaScript?

🔹 Stack vs Heap

Memory Type	What It Stores	Speed
Stack	Primitive values, references (pointers)	Fast ⚡
Heap	Complex objects (arrays, functions, etc.)	Slower 🐢

✅ So:

- The **array itself** (its contents) is stored in the **heap**.
- The **reference to the array** is stored in the **stack**.

🔍 What Happens During a Shallow Copy?

Javascript Copy

```
let original = [1, 2, 3];
let copy = original.slice(); // or [...original]
```

- `original` and `copy` are **two different references** in the stack.
- But they point to **two different arrays** in the heap (because primitives are copied by value).

✅ With Primitives:

- Each value is duplicated.
- No shared memory between `original` and `copy`.

🔥 With Objects (Non-Primitives):

Javascript Copy

```javascript
let objArr = [{ name: "ahmed" }];
let shallow = objArr.slice();

shallow[0].name = "ibrahim";

console.log(objArr); // [{ name: "ibrahim" }] ❌ mutation affects original
```

- `objArr[0]` and `shallow[0]` point to the **same object in the heap**.
- Because only the **reference** was copied, not the object itself.

🧠 Visual Analogy

Text Copy

```
Stack:
  original → → [1, 2, 3] (in heap)
  copy     → → [1, 2, 3] (new array in heap)

Heap:
  [1, 2, 3] ← original
  [1, 2, 3] ← copy
```

For primitives, both arrays are **independent** in the heap.