perplexity

can u recheck cuz my sir said in js new memory gets allocated as its a dynamic lang unlike cpp

Your sir's statement reflects a common misunderstanding about how memory works with primitives in JavaScript compared to C++.

In JavaScript, reassigning a primitive variable (like `let c = 10; c = 50;`) does NOT allocate new memory each time you reassign it—just as in C++, the single variable keeps its slot in memory, and its value is simply overwritten [1] [2]. JavaScript is "dynamic" in that variable types can change, but for primitives (numbers, booleans, etc.), reassignments replace the value in the same memory location (on the stack) [1] [2] [3].

- **Primitives in JS**: Numbers, strings, booleans, etc., are stored directly in the stack. When you do:

  ```
  let c = 10;  // allocates memory for c, stores 10
  c = 50;      // updates c's value in the same slot; does NOT allocate new space
  ```

  The variable `c` still uses the same memory location—only the value changes, no memory wastage [1] [2] [3].

- **Reference Types (objects, arrays, etc.)**: When you assign a new object or array, a new spot in heap memory is allocated, and the variable's reference changes to point to it. If the old reference isn't used, it will later be garbage collected [1] [2].

- **Why the confusion?** JavaScript is called "dynamic" because:
  - You can change the type of a variable (`let c = 10; c = 'hello';`), which may cause the engine to handle it differently behind the scenes.
  - For objects/arrays, new memory is indeed allocated for each new reference or reassignment [2] [3].

Summary:

- *Primitives* (like your example `c = 10; c = 50;`) do not cause new memory allocations with each reassignment—JS just updates the value in the same spot, much like C++ [1] [2] [3].
- *Reference types* (objects/arrays) do allocate new memory with each new value.

> For further reading:
>
> - “JavaScript primitive types...are stored in the stack because they are fixed-size data.... Changing the value just updates the variable's value in the stack.” [1]
> - “When you declare `let n1 = 10;`, memory is allocated on the stack. Assigning a new value to `n1` just updates it—no new allocation.” [2]

If your professor has a specific case (perhaps about reference types or "boxed" primitives), please share more details so I can clarify that edge case. But for *basic number/boolean/string variable reassignments*, JavaScript does not waste memory on each assignment.

⁂

1. https://dev.to/chintanonweb/mastering-javascript-memory-a-beginners-guide-to-stack-and-heap-57nj
2. https://www.geeksforgeeks.org/javascript/memory-management-in-javascript/
3. https://vinoo.hashnode.dev/primitives-vs-objects-in-javascript