CHAP 2: How JS Code is executed & Call Stack

```
var n = 2
function square (num) {
    var ans = num * num;
    return ans;
}
var square2 = square (n);
var square4 = square (4);
```

DATE

-> " It happens in two phases - Memory Creation Phase and Code Execution Phase "

• Memory Creation Phase.

Memory	Code
n : undefined square : { ... } (2.1) square2 : undefined square4 : undefined	

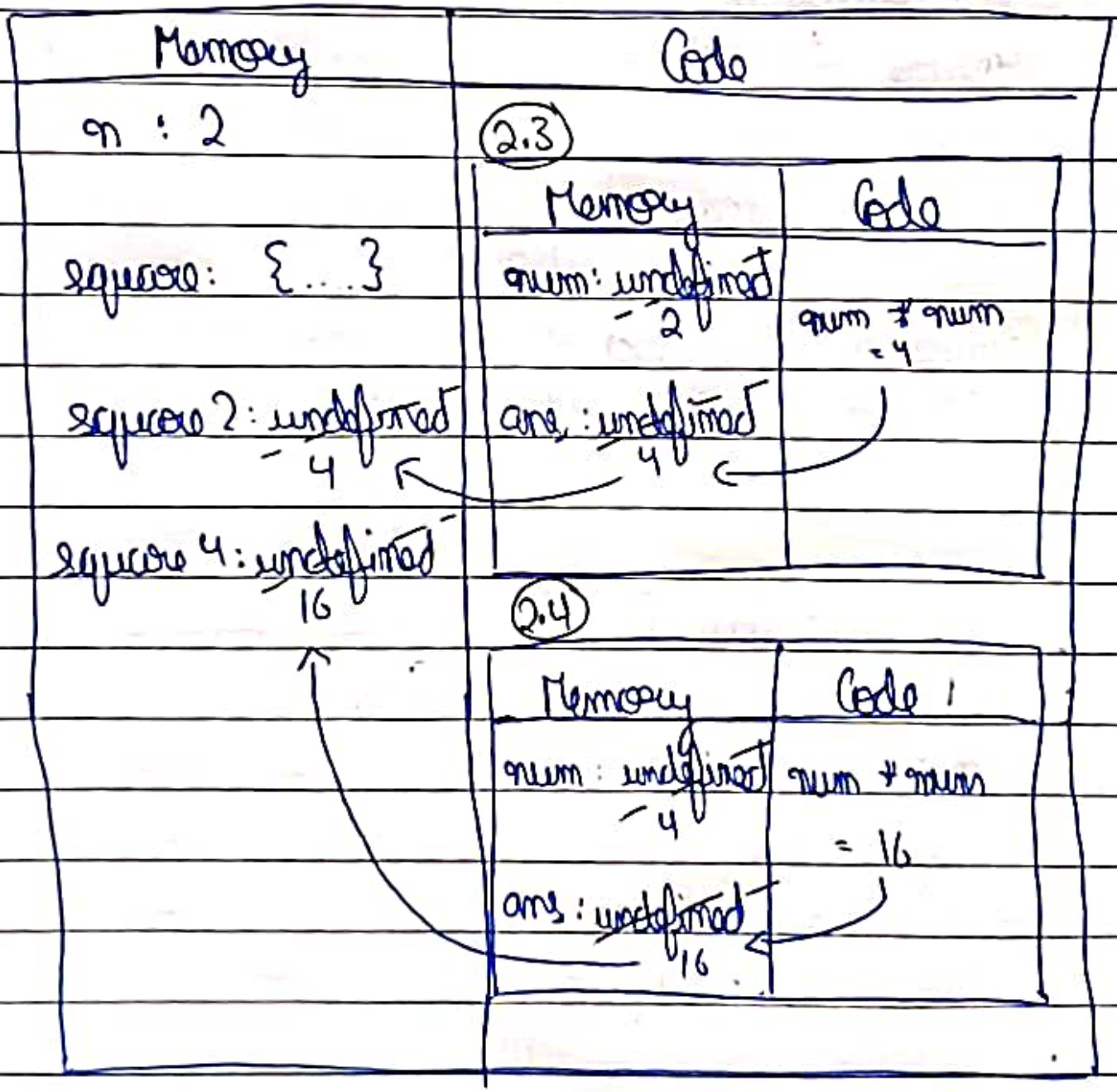
• Code Execution Phase.

Memory	Code
n : 2 square : { ... } square2 : undefined (2.2) square4 = undefined	

DATE

(2.1) The whole code will be shifted here (fⁿ code)

(2.2) Function invokation will happen and another execution context will be created inside 'Code section'

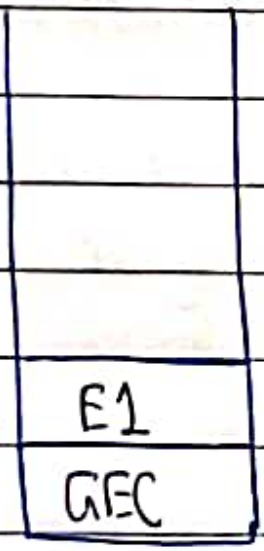


DATE

** Return keyword states that return the control of the program where it was invoked, and the whole execution context will be deleted **

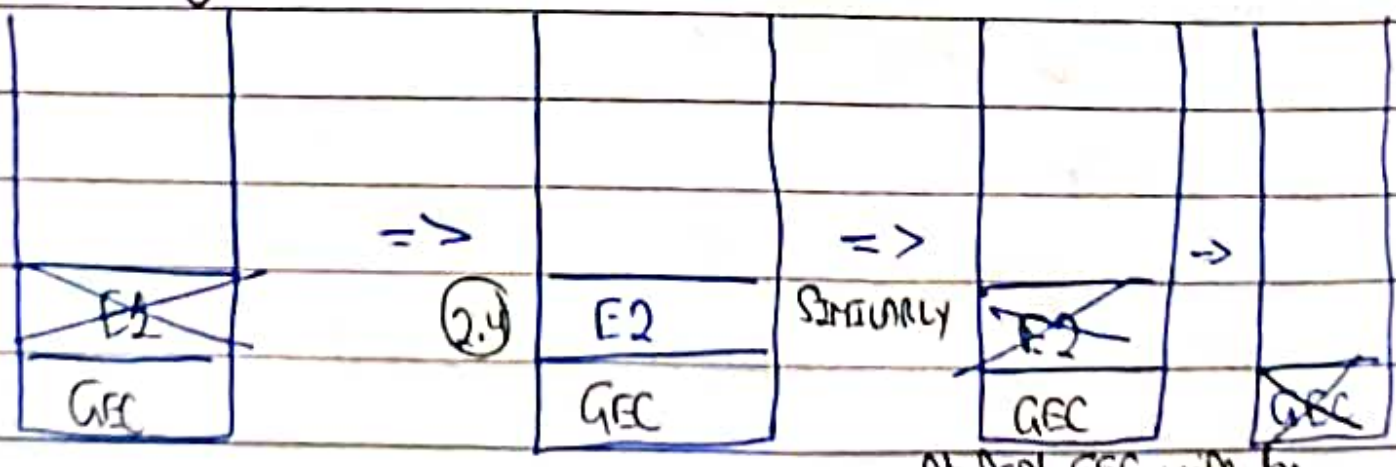
** After the program is finished, whole global execution context will be deleted **

• Call Stack •



* Everytime JS program is executed, GEC is created and pushed into the call stack *

(23) -> After the return statement, E1 will be popped off from the stack.



• At last GEC will be popped out.