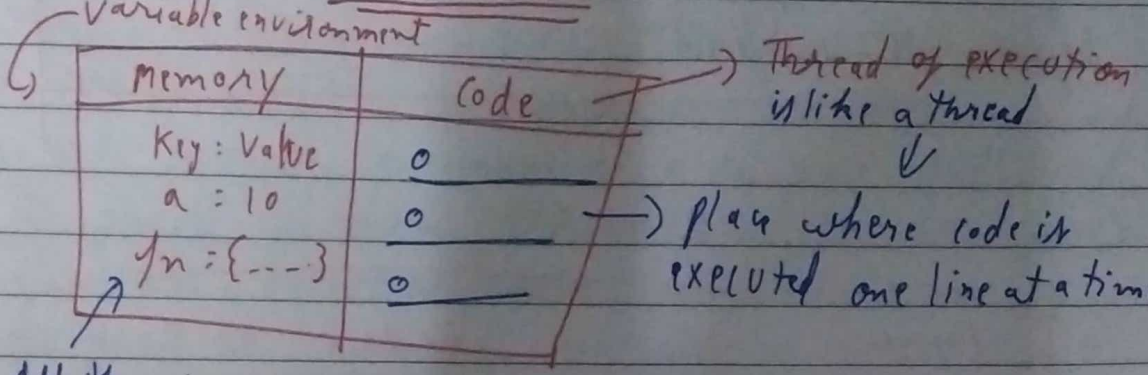
Date .....

How Javascript Works & Execution Context

→ Every thing in Javascript happens inside an execution context

Execution Context



→ Javascript is a synchronous single-threaded language that means Javascript can execute one command at a time in a specific order

How Javascript code is executed & call stack

we have code :-

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

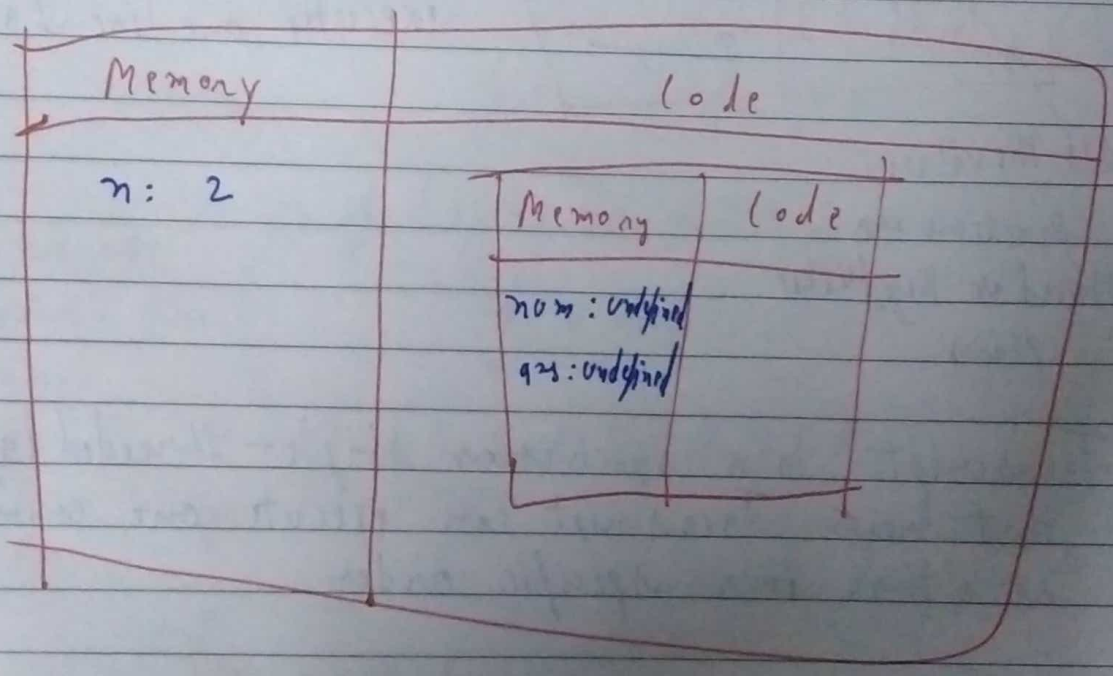
When you run this code, a global execution context is created This execution context is created in two phases 1. memory creation Phase in this phase JS will allocate memory to all the variables & functions

Date .....

Phase 1:

Memory	Code
n : undefined Square : { stores whole cod 3 Square 2 : undefined Square 4 : undefined	

phase 2:



In 1st phase, n is allocated with space & it stores undefined then we go to line two & allocates space for function 'square' & it stores whole code of the function inside its memory space. Then, square 2 & square 4 are variables as well, it allocates memory & stores 'undefined' for them.

Date .....

Now, in 2nd phase, it goes to whole code line by line. As it encounters n=2, it assigns 2 to 'n'. Until now, the value of n was undefined. For function right now there is nothing to execute. now when it encounters ~~n=~~ square 2 it calls our square function & a new execution context is created for our function. Now in this new execution context, in memory creation phase, we allocate memory to num & ans and undefined is placed in them. Now in code execution phase for this execution context, first 2 is assigned to num & var ~~ans~~ ans = 2*2 = 4
After that, 'return ans' returns the control of program back to where our code was ~~executed~~ called from. & our function's execution context will be deleted
Same thing will happen for square 4
And after executing all these code our global execution context will be deleted.

→ Javascript manages code execution context creation & deletion with help of Call Stack.