JavaScript -

→ Everything in JS happens inside an "Execution Context."

Execution Context — is like big box - 2 components in it

Variable Environment	Memory (All the variable & fn are store as a key : value pairs)	Code (Thread of Execution)
	Key : value a : 10 fn : {....}	{Code is executed one line at a time}

↳ JS is a synchronous, threaded (single) language.

Single threaded — Executes one command at a time.

Synchronous single threaded — JS executes one command at a time, and in a specific order.

i.e. It can only go to the next line once the current line has been finished executing.

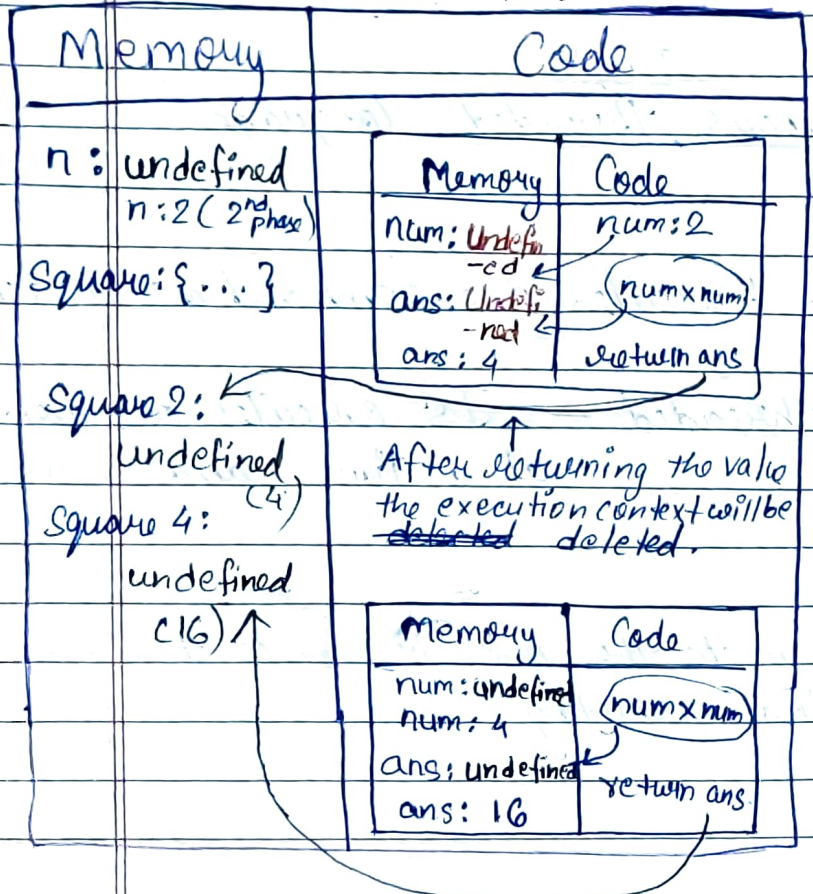
Date __/__/__
Page _____

Q. How JS is executed?

Execution context is created in 2 phases

- Memory creation phase (Critical phase)
  - It allocates memory to all the variables and functions.
- Code Execution phase.

Global Execution Context.



Example :-

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

function invocation

Date __/__/__
Page _____

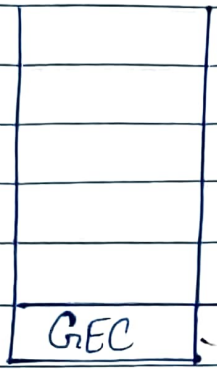
allocates special
Undefined : Basically the value i.e. "undefined" to it.

Square fn — It will have the whole code inside it (i.e. code inside Orange circle)

Return — Return the control of the program, to the place where this function was invoked.

Note:- Once the program is finished the whole global Execution content will be deleted.

→ JS has its own Call Stack



This GEC is poped up when program is finished

Whole Global Execution Content is pushed inside this Stack

→ Call Stack maintains the order of execution of execution content.

→ It is also known as
- Execution content Stack
- Program Stack
- Control Stack
- Runtime Stack
- Machine stack