How JavaScript is Executed

- Everything in javascript happens inside the execution context. When the JS engine scan a script file, it makes an environment called execution context, that handles the entire transformation and execution of the code.

Execution Context

Memory Creation	Code execution
key : value a : 12 fn : {...}	o o o

- Execution Context has two phases first "Memory Creation" also known as "Variable Environment", where all the variable register in key value pairs.
- Second phase is "Code Execution" also known as "Thredd of Execution". Where code is executed one line at a time. Because JS is "synchronous single-threaded" language.
- Let's consider the example below:

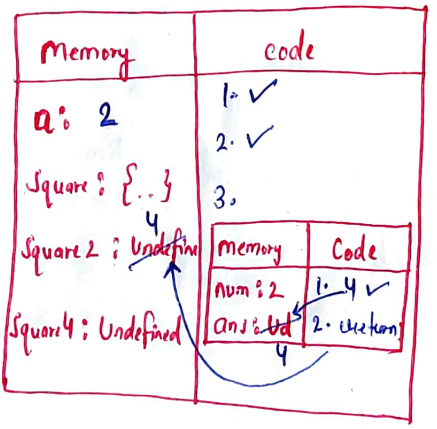
```
1. → Var a = 2;
2. → function square(num) {
        Var ans = num * num; ← 1.
        return ans;          ← 2.
     }
3. → Var square2 = square(n);
4. → Var square4 = square(4);
5. → Console.log(square2, square4);
```

- First execution context has been created, In first phase "memory creation" allocates a memory space for variable and function.

Memory	Code
a : Undefined	
Square : {--}	
Square 2 : Undefined	
Square 4 : Undefined	

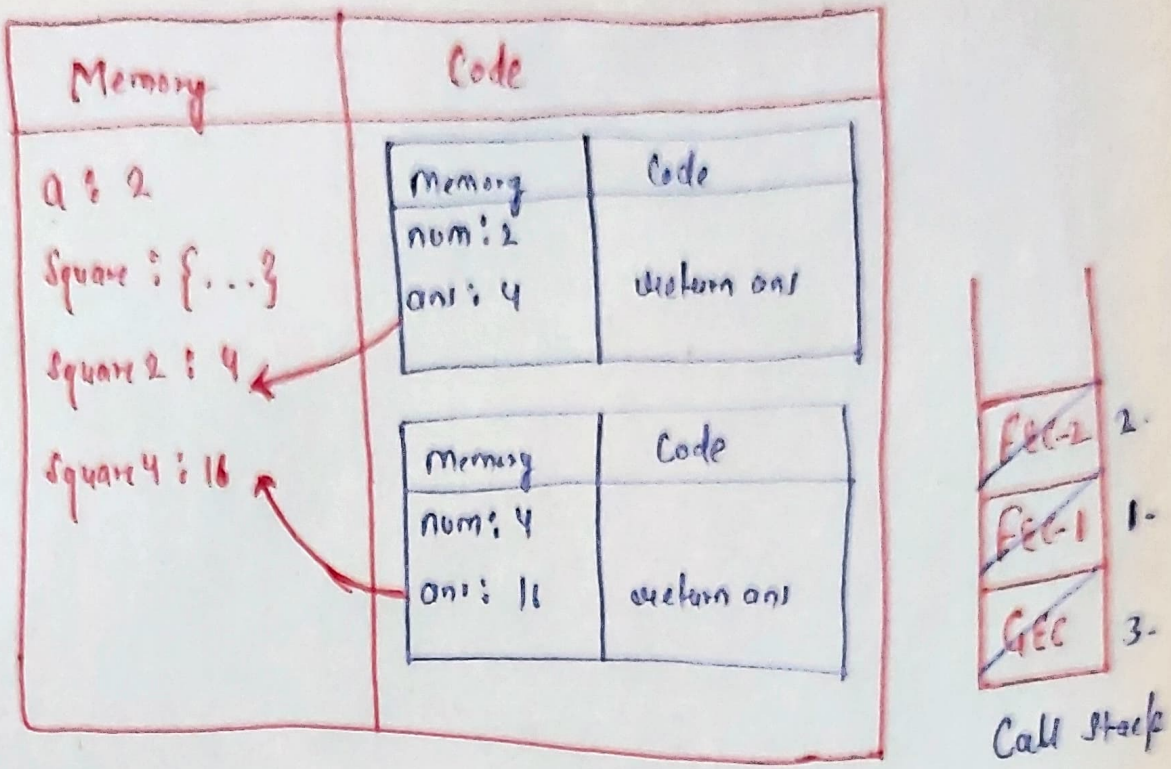
- When allocating memory special value "undefined" registered in variable, for function it stores whole code of the function.

- Second phase "code execution", Run code one line at a time.



- At line 1. 2 assign to variable "a", line 2. already executed a. function register, When line 3. encountered New "Local execution context created for calling a square function.

- Same things happen in "Local execution Context". When line 2 executed "return" keyword is encountered, It returns to the control to the called line which is Line 3. at global execution Context, also local / function execution context is deleted.



- JS manage code execution context creation and deletion with the help of "Call Stack".
- Call Stack is a mechanism to keep track of its place in script that calls multiple function.
- Call Stack maintains the order of execution contexts.