JavaScript

* **Everything in Javascript happens inside an “Execution Context”.**
* Execution Context :

**It is divided into 2 parts:**

**a) Memory (Variable Environment)**

**b) Code (Thread of Execution)**

a) Memory :- Stores values and keys.

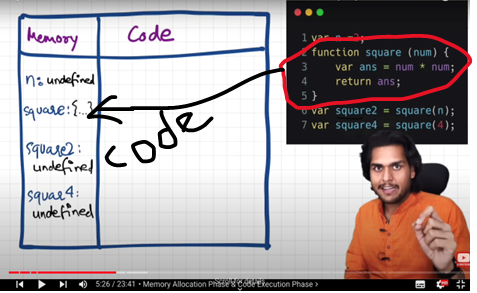
  ex. Key : value –  a : 10

b) Code :- Executes one line at a time in a specific   order.

    -    The main and outer most layer of execution context is called as **“Global execution Context”**.

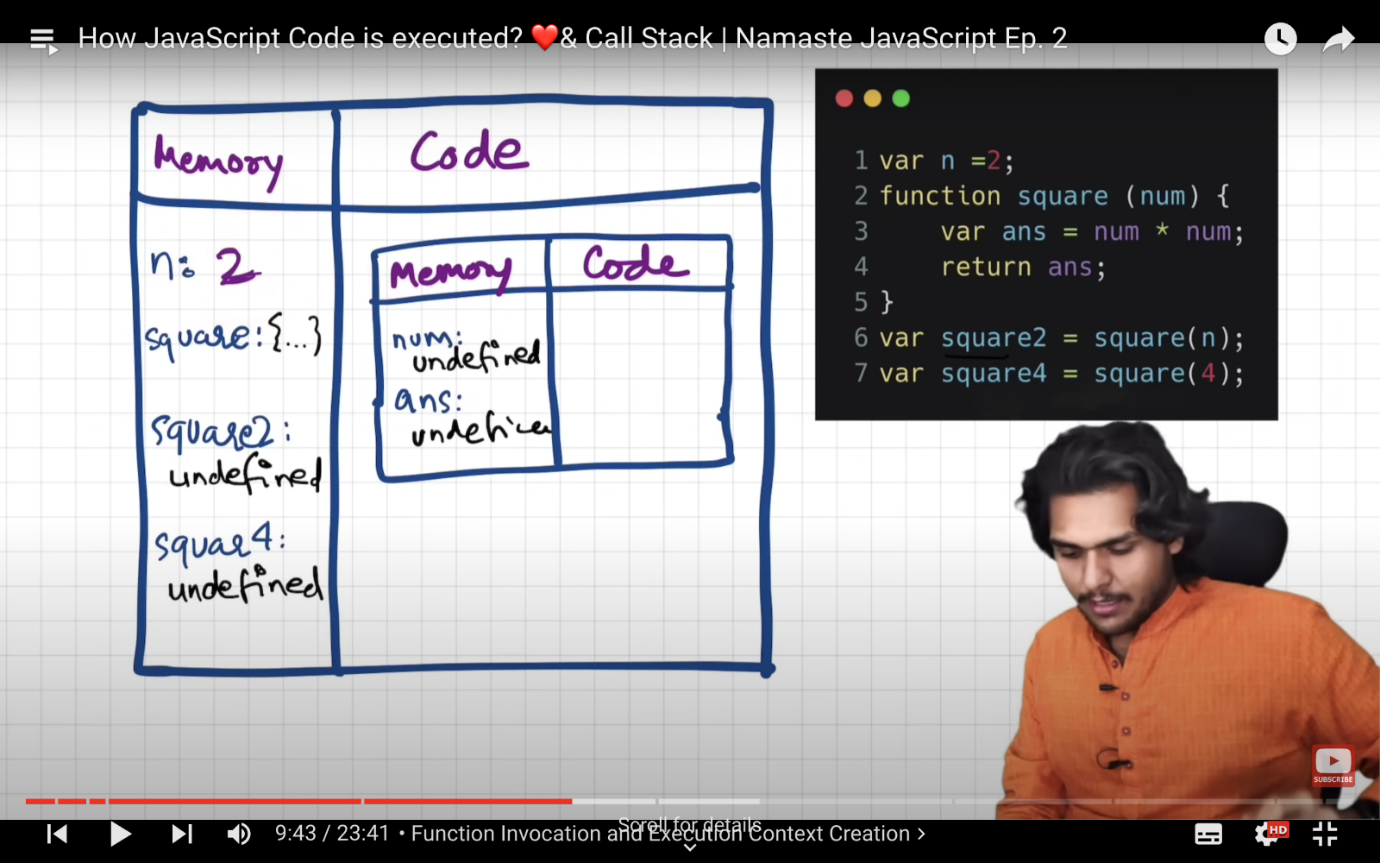
    -    **“JavaScript is a synchronous single-threaded language”.**

-     **Code Execution: 1.  Memory Creation Phase**



-     **Code Execution: 2.  Code Execution  Phase**

* Function Invocation and Execution Context Creation

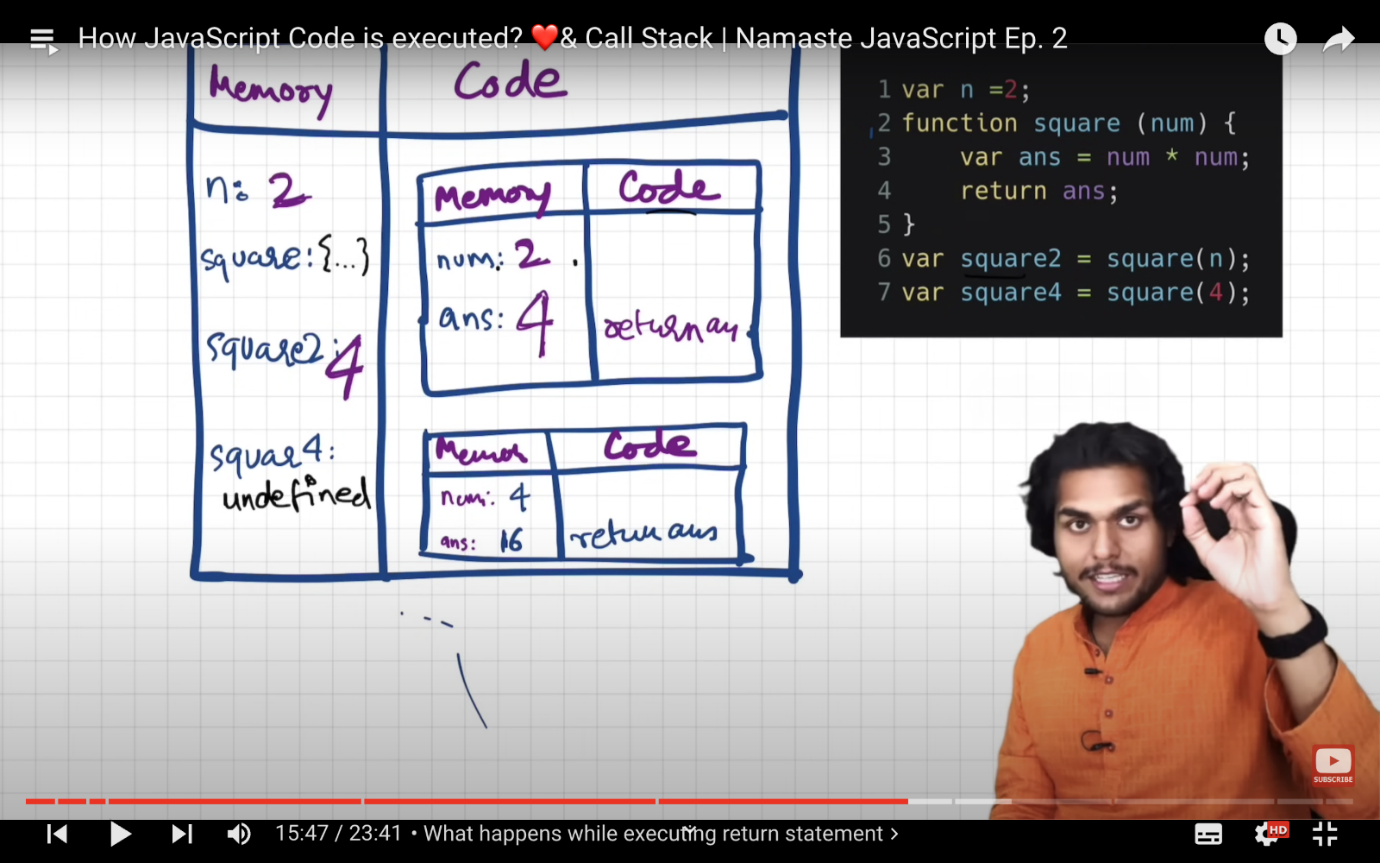


**num is a parameter**

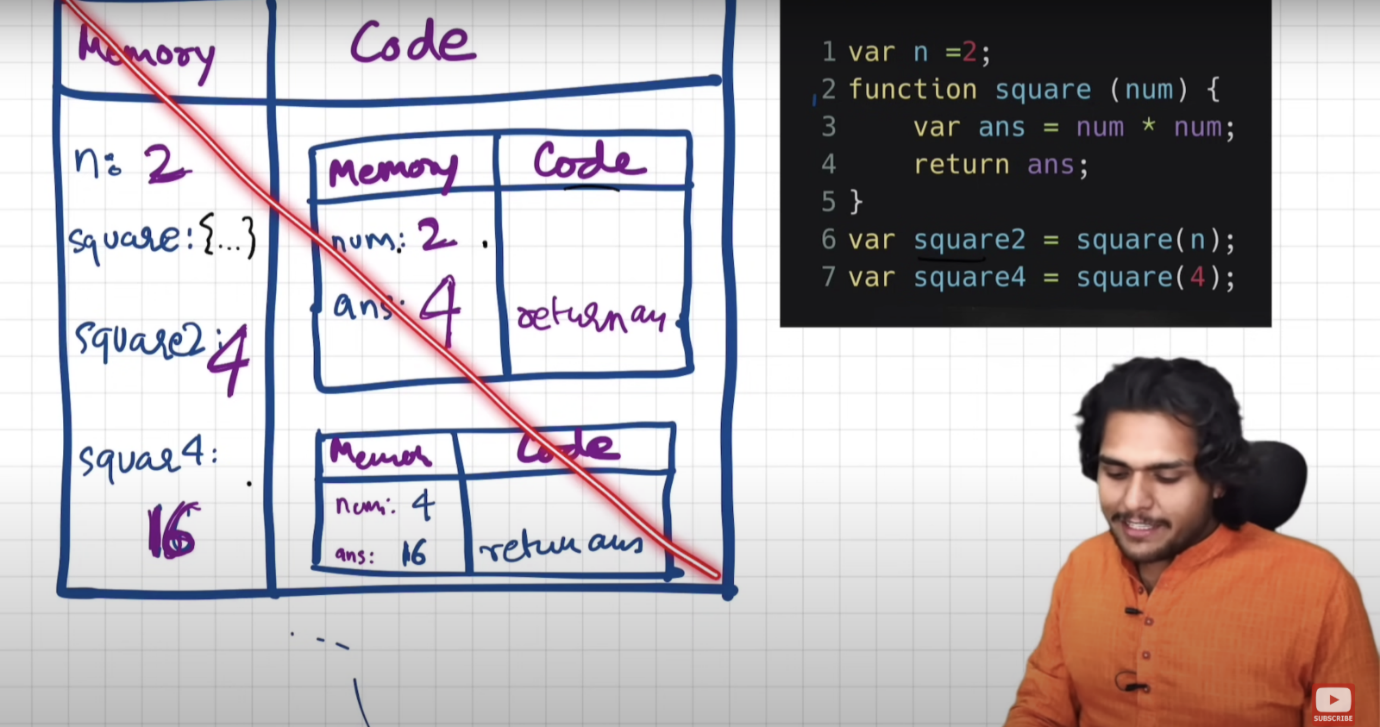
**n is argument**

* What happens while executing return statement

It creates space inside the code part again.. As per requirements. And at first every variable is stored as undifined.



**After running through the whole code the whole execution context is deleted.**



* **The Call Stack in JavaScript**

**“Call Stack maintains the order of execution of execution context.”**

The Global Execution Context is at the very bottom in the    stack. As the new stack comes in it is above that and whenever the Execution Context (EC1) is executed it pops up from the stack. And as the EC inside the EC1’s turns for the execution then the EC1 is executed, and it pops up and then it is the turn of EC2. Like this the stack works in JavaScript.

|  |
| --- |
|  |
|  |
| EC1 |
| GEC (Global Execution Context) |

**After execution of EC1 it pops out and then EC2 gets in. Like this it will continue to ‘n’ number of Execution Context.**

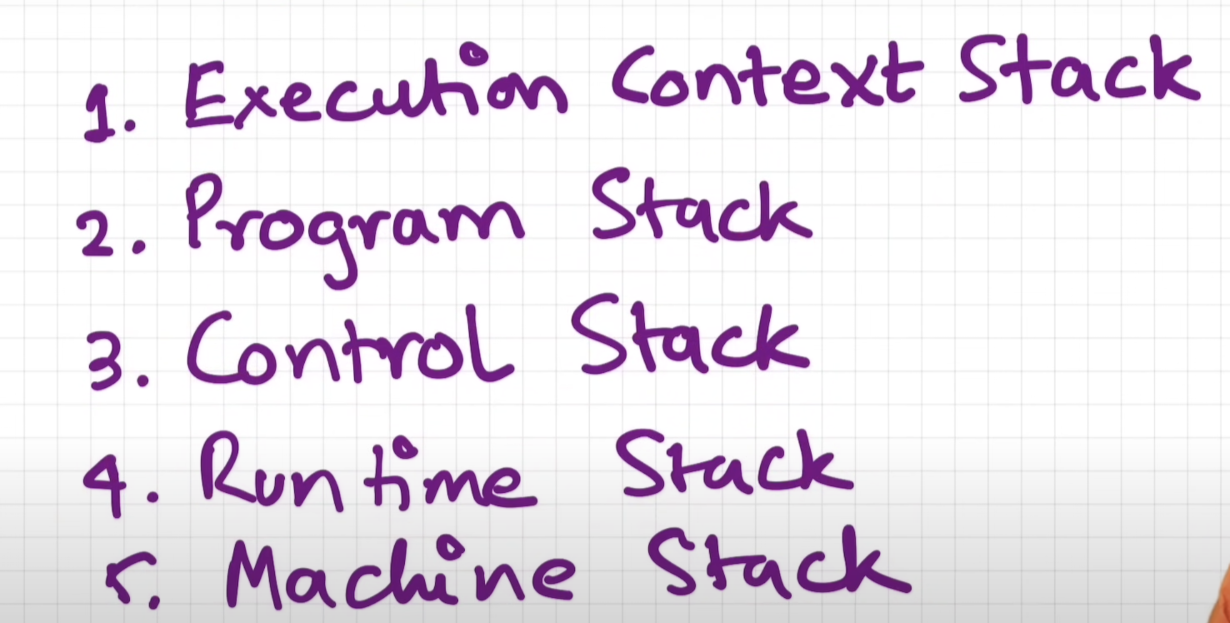
|  |
| --- |
|  |
|  |
|  |
| GEC (Global Execution Context) |

After Execution EC1

|  |
| --- |
|  |
|  |
| EC2 |
| GEC (Global Execution Context) |

Like this all the creation and deletion of EC is managed by JavaScript and at the end GEC is also deleted. And the “Call Stack” is empty.

* **Call Stack has various names:**



         It is all the same thing.