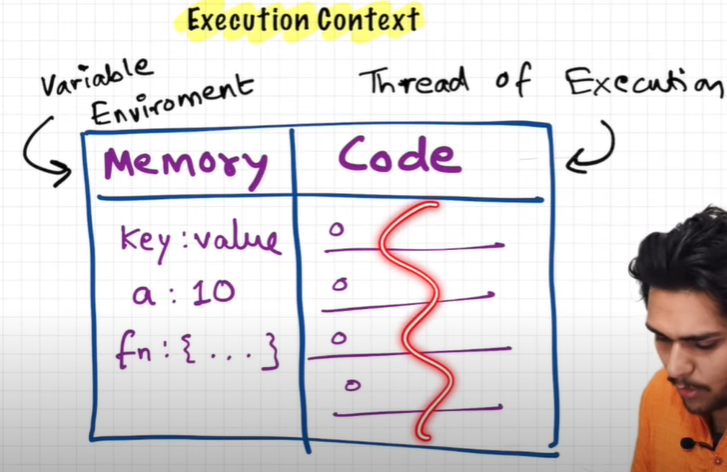
**JS Execution**

* JavaScript is a **synchronous, single-threaded** language.
* Diagram

  Description automatically generated
* Graphical user interface, application

  Description automatically generated
* Diagram

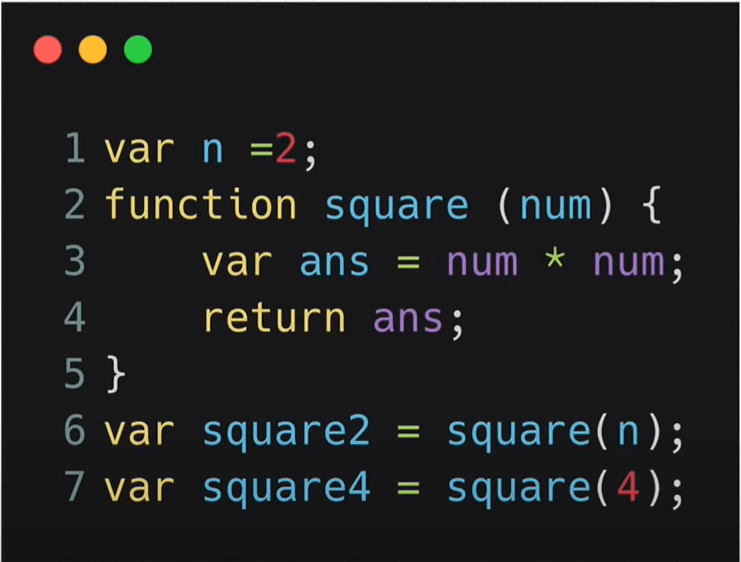
  Description automatically generated
* Everything happens in JS in an **Execution Context**.
* It consists of two parts; one is **memory component** and another one is **code component**.



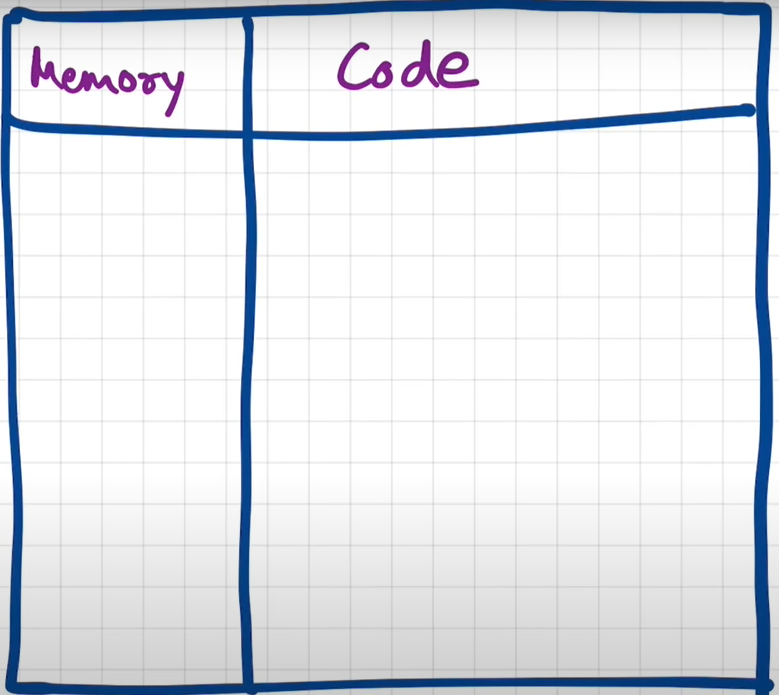
* **Memory component** –
  + It is also known as **Variable environment**.
  + All variables are stored in a **key value** pair.
* **Code component**-
  + It is also known as **Thread of execution**.
  + All code is stored and executed **one line at a time**.

**-) How a JS works in background**

* Taking example of this code

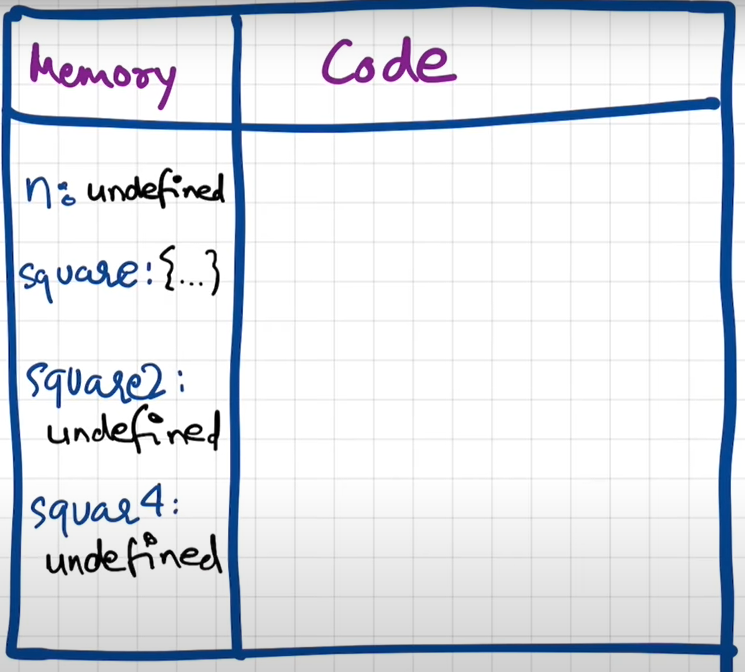


* At the start of the execution of the program a **Global Execution Context(GEC)** is created, and it is pushed inside a **Call stack** and this phase is known as **Memory Creation phase**.

* Phase 1 (Memory allocation phase)
* In this phase JS allocates memory to all the variables in the memory component and it stores a special value to each variable known as **undefined**.
* undefined is like a placeholder for values.
* In case of functions it stores the whole code inside the memory space.
* Text

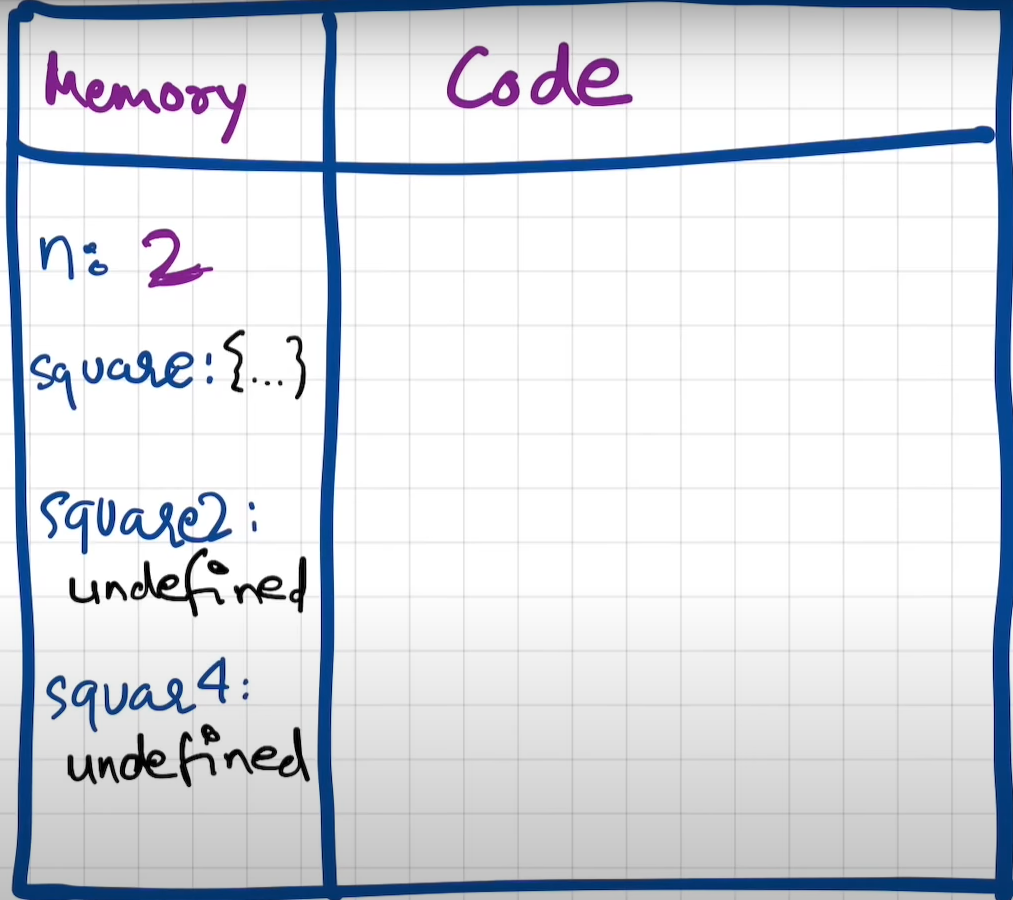
  Description automatically generated



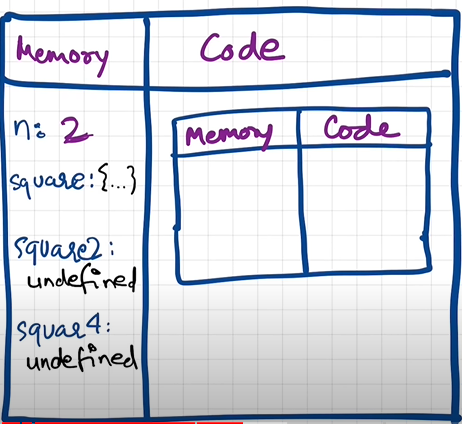
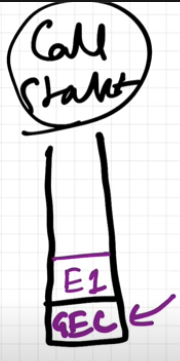
* Phase 2 (Code execution phase)
  + In this phase JS again runs down the code and all the calculations and function execution take in this phase.
  + All the values of the variables are moved to the placeholder.
  + After execution of line 1 of code –

Text

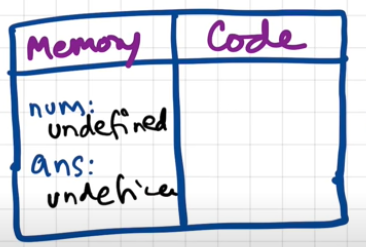
Description automatically generated

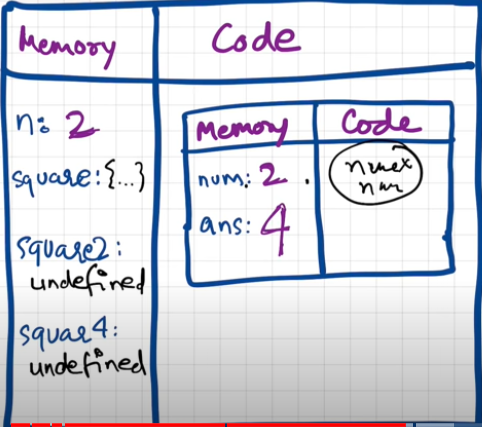


* + The function in line 2-5 has no execution as no one called this function.
  + In line 6 the function is called(invoked) with an argument and here a new execution context(suppose E1) is created inside the GEC and is pushed into Call stack .

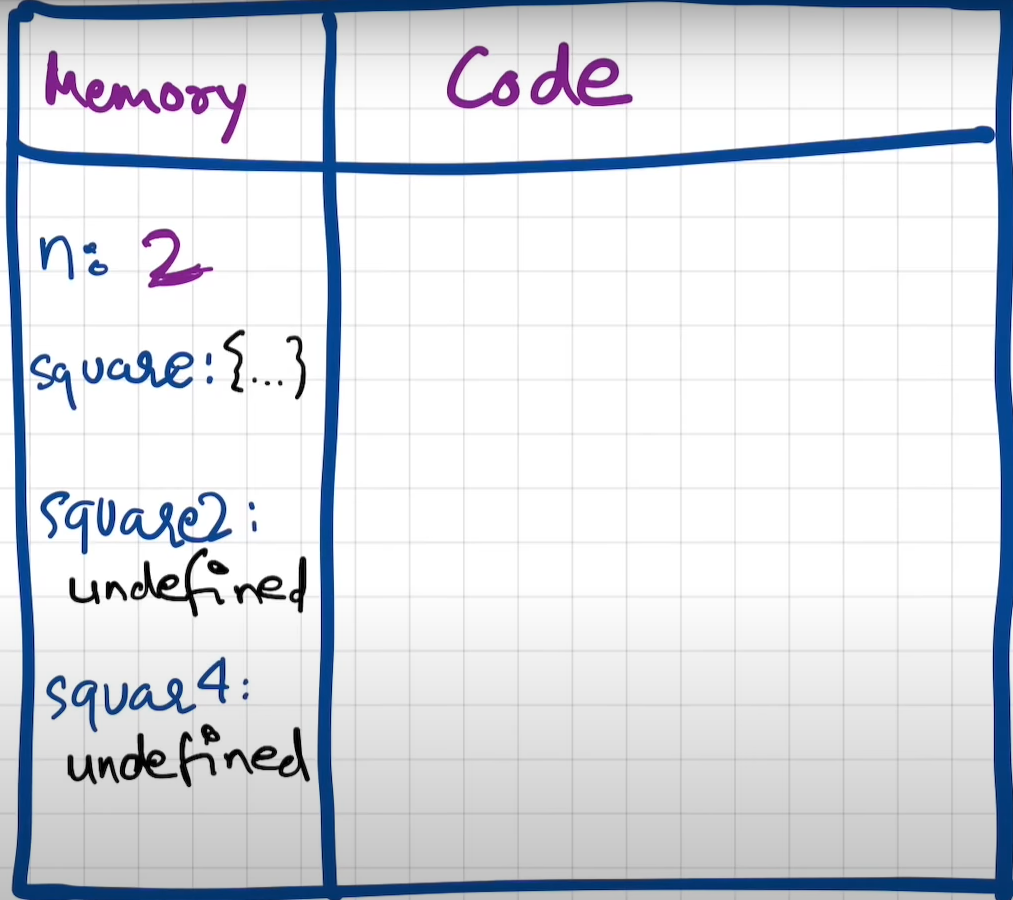
 

* + And same from phase one will start considering this execution context E1 as main program.





* + In line 4, once the return is encountered it states that **return the value of ans to the calling variable by first searching the value of ans in memory component and pass it back**.
  + The value of square2 in GEC is updated to the returned value and **the execution context E1 is completely deleted and removed from the Call Stack**.

In the image above square2 is updated to 4, it is not shown in the image.

* + Same happens to square4 variable.
  + As the whole program is completed the GEC is also removed from the call stack.