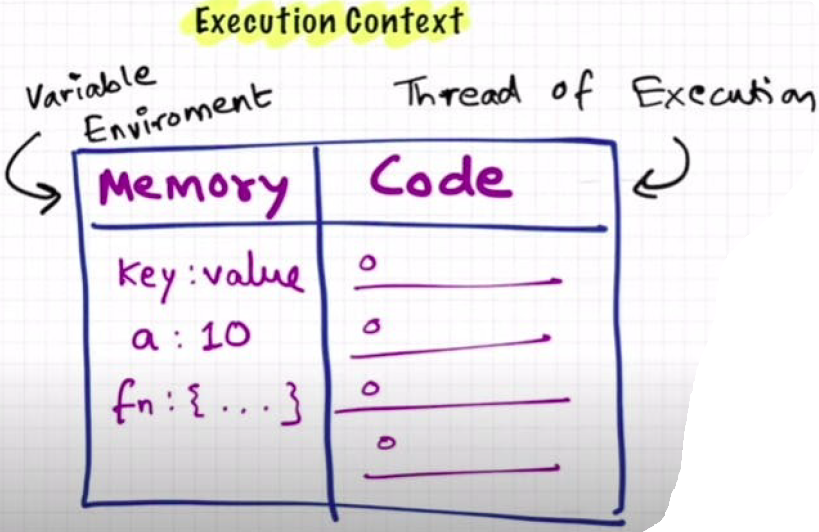
**EXECUTION CONTEXT**



* In the container the 1st component is **memory component** and the 2nd one is **code** **component**.
* Memory component has all the variables and functions in key value pairs. It is also called **Variable environment**.
* Code component is the place where code is executed one line at a time. It is also called the **Thread of Execution**.
* JS is a **synchronous**, **single-threaded** language
  + Synchronous: One command at a time.
  + Single-threaded: In a specific synchronous order.

**JS EXECUTION AND CALL STACK**

* When a JS program is run, a **global execution context** is created.
* The execution context is created in two phases.
  + Memory creation phase - JS will allocate memory to variables and functions.
  + Code execution phase

var n = 2;

function square(num) {

  var ans = num \* num;

  return ans;

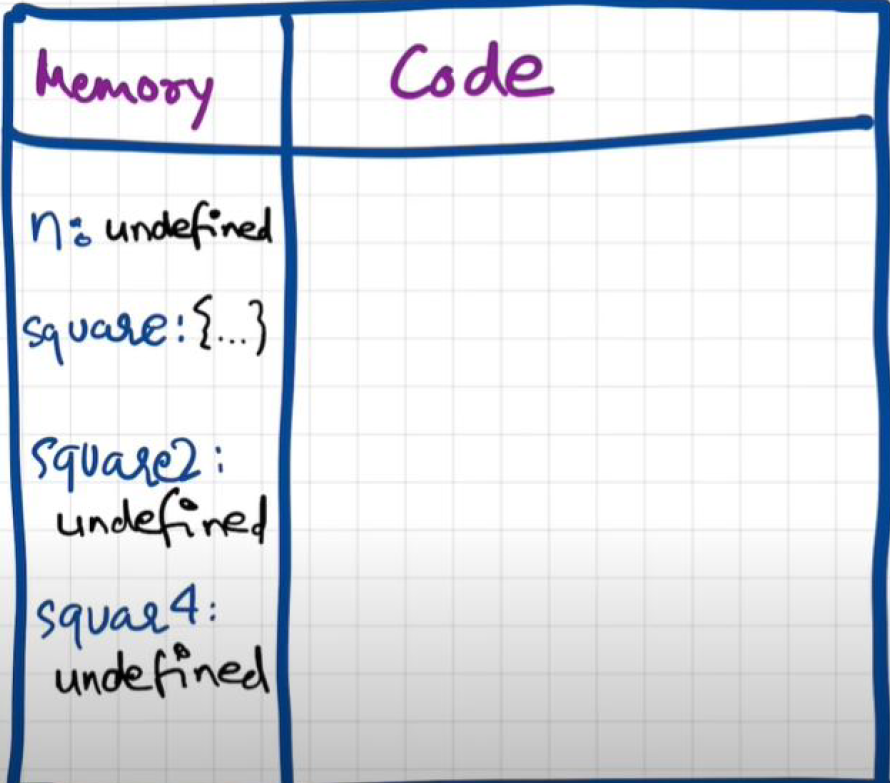
}

var square2 = square(n);

var square4 = square(4);

**PHASE – 1**: **Memory Creation Phase**

* The very first thing which JS does is **memory creation phase**, so it goes to line 1 of above code snippet, and allocates a memory space for variable 'n'.
* It goes to line 2, and **allocates a memory space for function 'square'.**
* When allocating memory for **n it stores 'undefined'**, a special value for 'n'.
* For 'square', it stores the whole code of the function inside its memory space.
* As square2 and square4 are variables as well, it allocates memory and stores 'undefined' for them.
* **The below is the representation:**

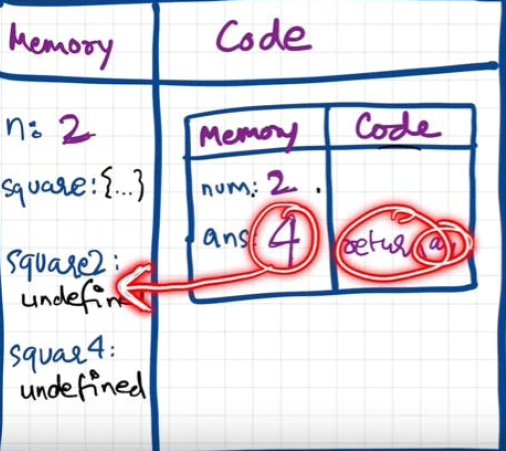


**PHASE – 2: CODE EXECUTION PHASE**

* It starts going through the whole code line by line.
* Now it encounters var n = 2, it assigns 2 to 'n'. And **undefined** value earlier is now replaced by 2.

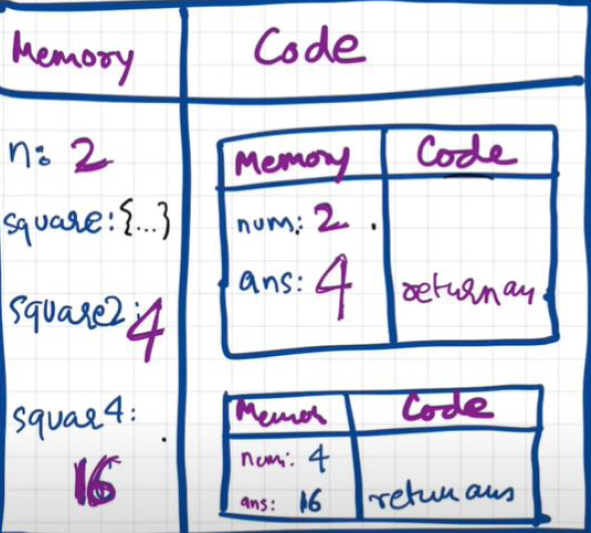
**Line 6 of Code**

* **var square2 = square(n)**
* Here it encounters a **function square** a new execution context is created.
* In memory creation phase, we allocate memory to **num** and **ans** the two variables.
* In Code Execution Phase value 2 is assigned to num. Then var ans = num \* num. will store 4 in ans.
* Once all the execution is completed then the control will be back to where the function was invoked.
* When the **return** keyword is encountered, control returns to the called line and the **execution context is deleted**.



* This process repeats for **var square4 = +square (4)**
* Once the whole code execution is completed the whole global execution context will be destroyed.

**The final diagram will before deletion would look like this**



* **JavaScript** manages 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 of execution contexts. It is also known as
  + **Program Stack,**
  + **Control Stack,**
  + **Runtime stack,**
  + **Machine Stack,**
  + **Execution context stack.**