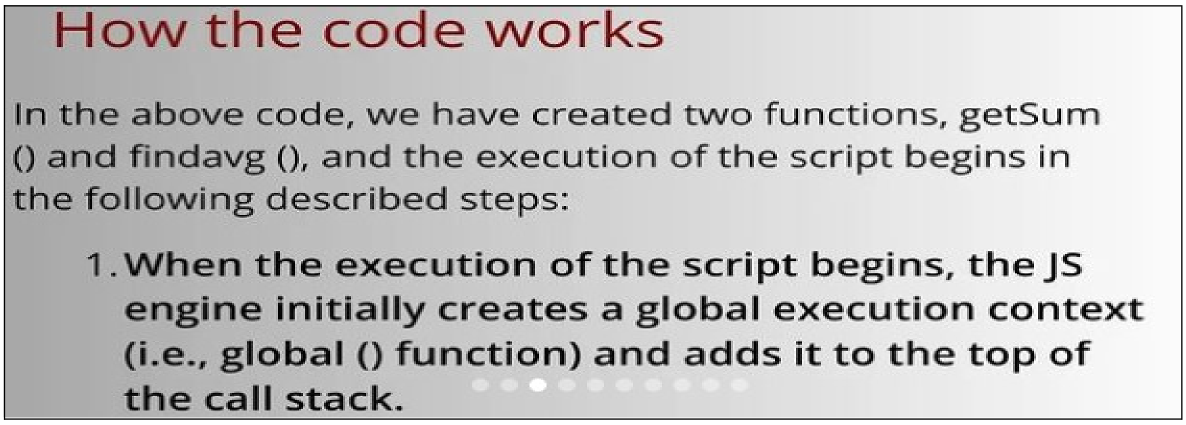
<u>JAVASCRIPT CALL STACK - EXECUTATION CONTEXT</u>

<u>JS Contexts :-</u>
Global Execution Context and Function Execuation Context
They are executed via JS Engine.

<u>Manage JS Contexts:-</u>
In order to manage it, JS uses Call Stack.

<u>Call Stack:-</u>
It is a data structure that keep track information of the functions being called and executed.

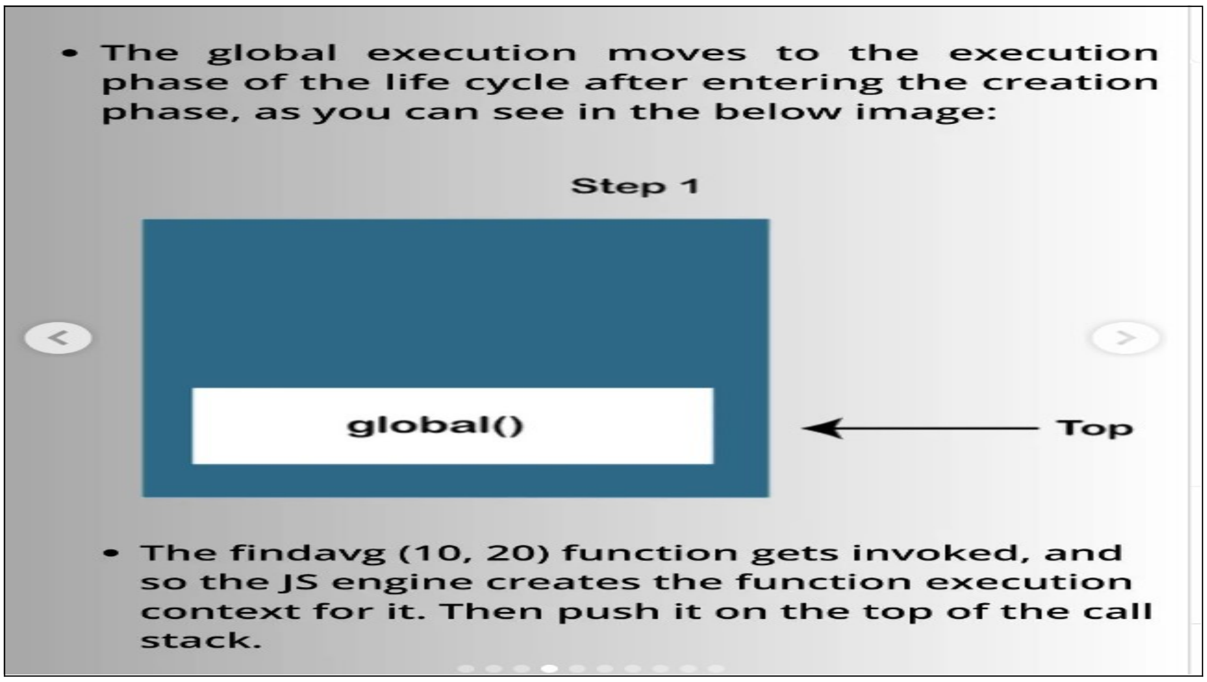
<u>How Code works:-</u>



How the code works

In the above code, we have created two functions, getSum () and findavg (), and the execution of the script begins in the following described steps:

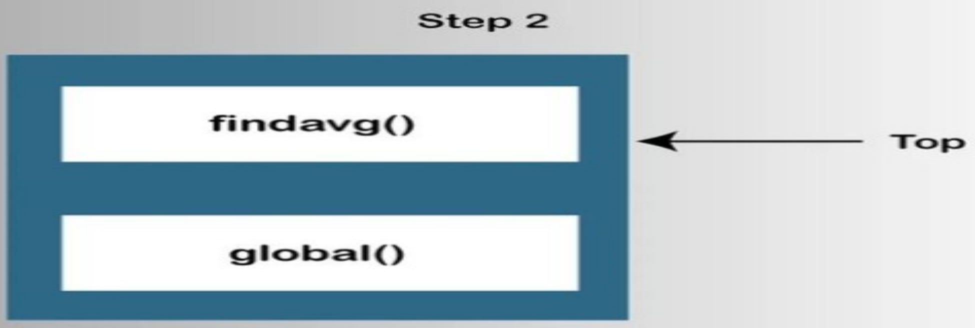
1. When the execution of the script begins, the JS engine initially creates a global execution context (i.e., global () function) and adds it to the top of the call stack.



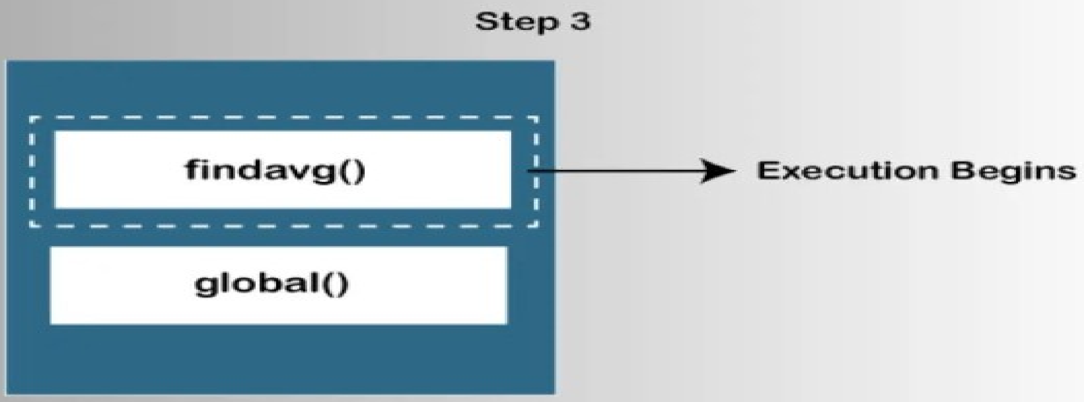
- The global execution moves to the execution phase of the life cycle after entering the creation phase, as you can see in the below image:

- The findavg (10, 20) function gets invoked, and so the JS engine creates the function execution context for it. Then push it on the top of the call stack.

- So, now in the call stack, two functions are pushed, i.e., global () and findavg(), and on the top of the stack, the findavg() function is present, as you can see in the below image:

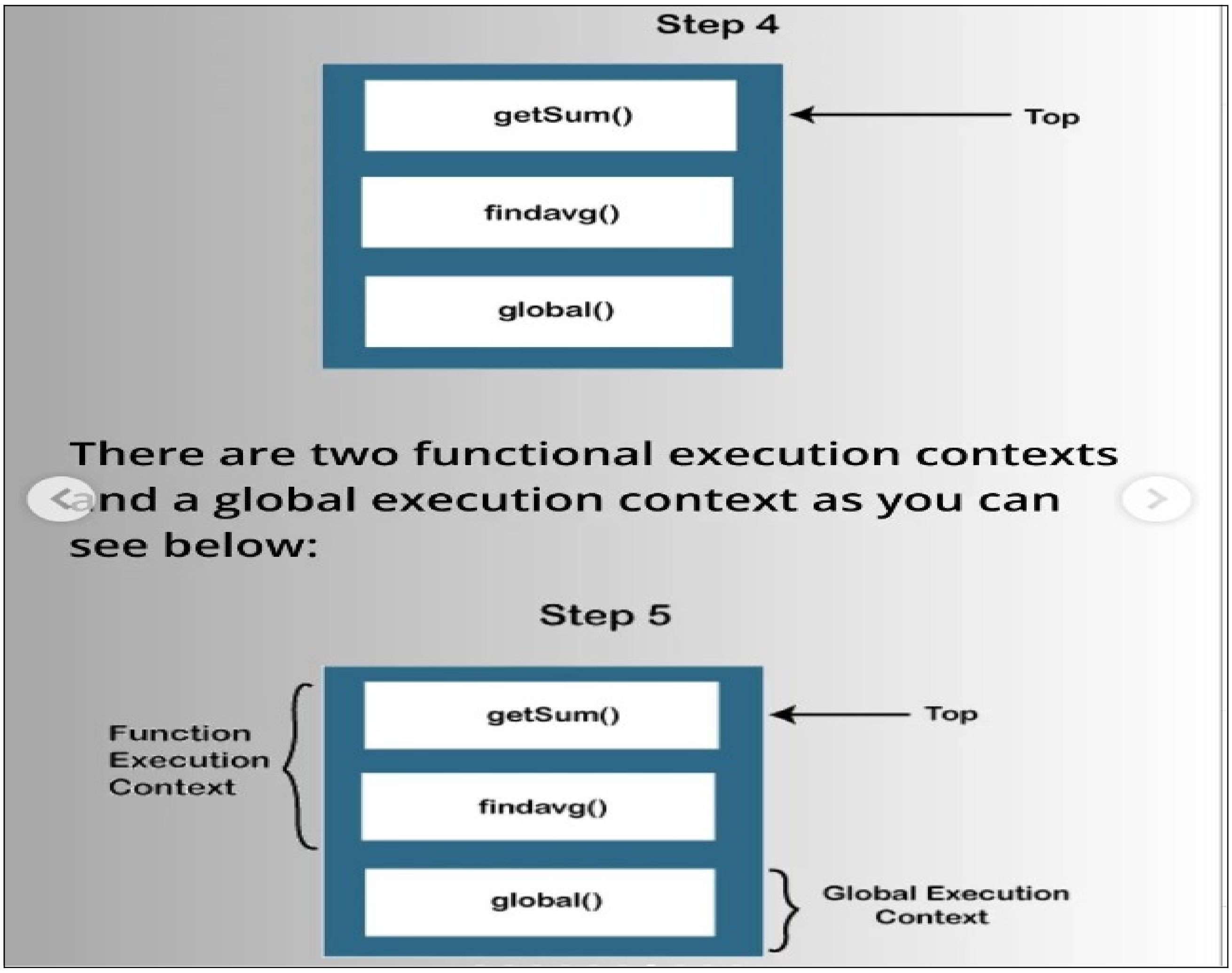


- The JS engine begins the execution of the findavg () function because it exists on the top of the stack, as you can see in the image:



- As in the code, the getSum () function is invoked inside the findavg () function definition, so the JS engine creates a function execution context for the getSum () function and pushes it on the top of the stack.

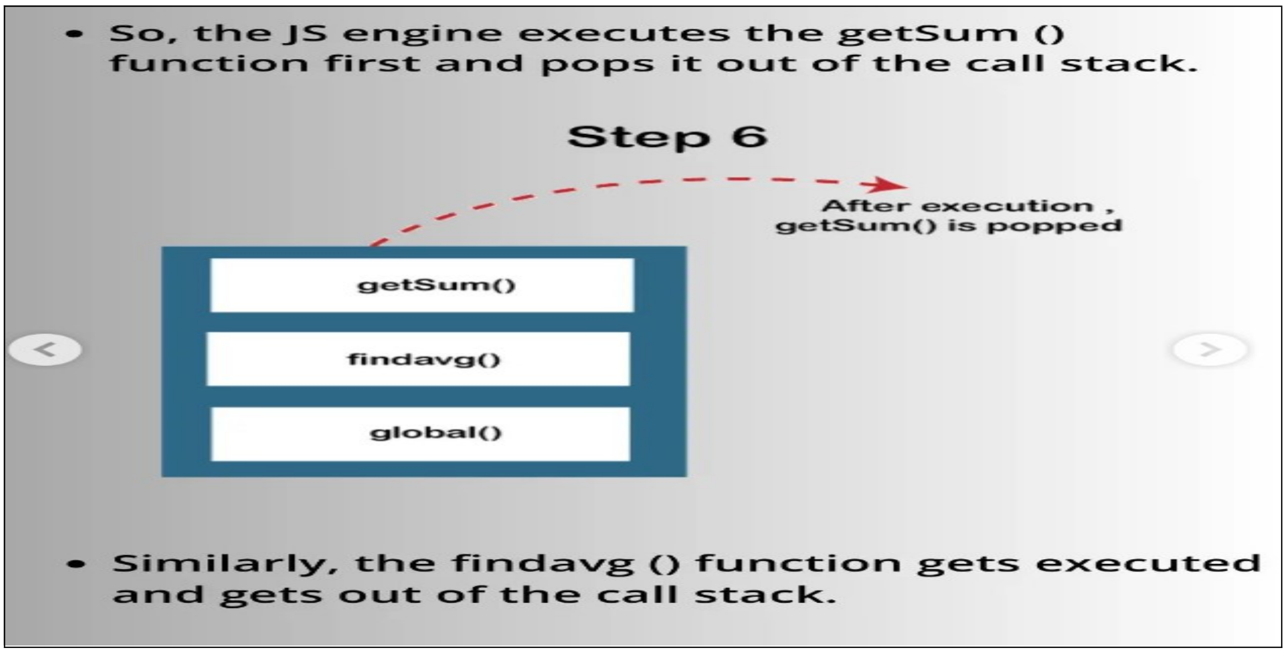
- Now, in the stack, there are three functions present, which are global (), findavg (), and getSum () functions, as you can see in the below image:



Step 4

There are two functional execution contexts and a global execution context as you can see below:

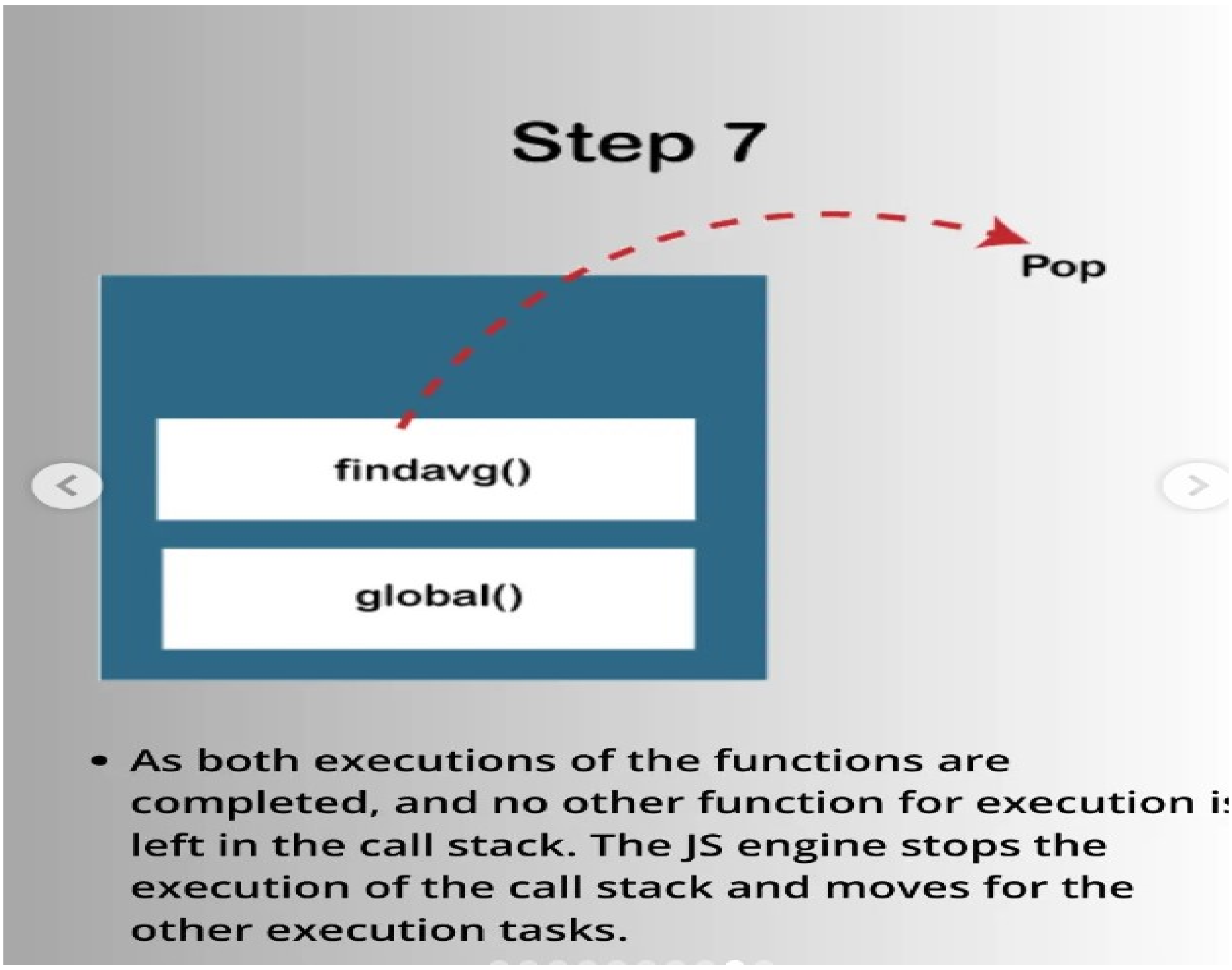
Step 5



- So, the JS engine executes the getSum () function first and pops it out of the call stack.

Step 6

- Similarly, the findavg () function gets executed and gets out of the call stack.



Step 7

- As both executions of the functions are completed, and no other function for execution i left in the call stack. The JS engine stops the execution of the call stack and moves for the other execution tasks.