

# Namaste JavaScript

Date \_\_\_\_\_

Page \_\_\_\_\_

\* How JavaScript works. \*

Q.

1) is JavaScript Synchronous or asynchronous.

2) is JavaScript Single threaded or multi-threaded.

\*)

IMPORTANT THING to KNOW

Note :-

Everything in JavaScript happens inside an execution context.

→ Assume that "execution context" is a big Box and the whole JavaScript code executed.

Execution Context of execution

Also called Thread

Memory

Here all the variables and functions stored in the form of key value pairs.

a : 10  
fn : { ... }

Code.

Here the code is executing one by one.

0 \_\_\_\_\_  
0 \_\_\_\_\_  
0 \_\_\_\_\_  
0 \_\_\_\_\_

Also called variable environment



## IMPORTANT THING TO KNOW

Note

JavaScript is a Synchronous  
Single-threaded Language

\* Let's understand with one example

```
var n = 2;
function square(num) {
  var ans = num * num;
  return ans;
}
```

Parameters

```
var s1 = square(n);
```

Argument

Start  
from  
here

First create  
Execution context

Memory

Code

① n : undefined → ②

② square : { store whole  
function code }

③ s1 : undefined → ④

~~④ s1 : undefined~~

first phase

Second phase

① store "2" value to "n"

② Here we have function  
and nothing to do.

③ function invocation

Memory

Code

num : undefined → ②

ans : undefined

① num : 2

② num \* num

③ return

↓

h

return

with value

④



Date \_\_\_\_\_  
Page \_\_\_\_\_

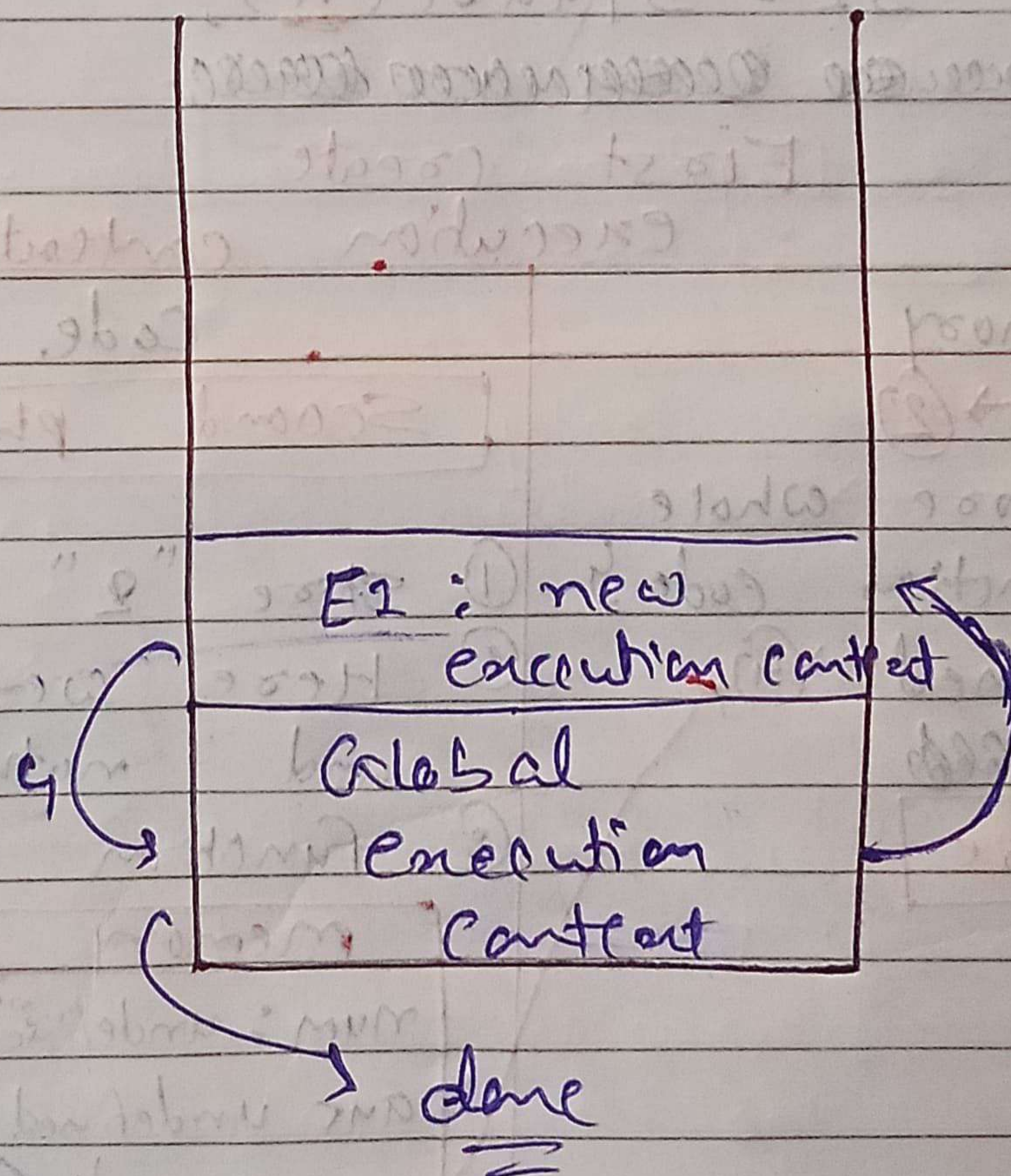
In this there are two  
phase

- ① memory creation phase
  - ② code execution phase

Note

When ~~there~~ a function  
is encountered to execute  
then the new execution  
context are created


JavaScript manages the  
Call stack





"Call stack maintains the order of execution of execution context."

Call stack also known as :

- 
- 1) execution context stack
  - 2) program stack
  - 3) control stack
  - 4) Runtime stack
  - 5) maintain stack.