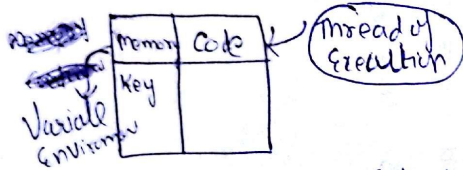


## How Javascript works

① Everything happens in JS done in Execution context.

Execution Context →



Memory component (Variable Environment) :- contains Variable & Functions as Key/Value pair  
 Code component (Thread of Execution) :- Where whole JS code is executed.

• Javascript is synchronous, single threaded lang

Phases → ① Memory creation → ② Code Execution Phase

```

1. var n = 2
2. function square(num) {
3.   var ans = num * num;
4.   return ans;
5. }
6. var square2 = square(n);
7. var square4 = square(4);
    
```

### ① Memory Creation Phase

Control moves to line 1 then ahead, takes memory & creates Execution Context first.

Variable Environment	Code
n: undefined	
square: {square code}	
square2: undefined	
square4: undefined	

Global Execution Context

A new Execution Context from E2

### ② Code Execution Phase

~~Control~~ In this phase at 1

V.E	
n: 2	
square: { }	
square2: { }	

nothing to with 2 to 4 on line 5 → function invocation

V.E	
n: 2	
square: { }	
square2: { }	
square4: { }	

V.E	T.Execution				
n: 2 Answer: 8 square2: 0 undefined square4: undefined	<table> <tr> <th>V.Env</th><th>T.Env</th></tr> <tr> <td>num: undefined ans: undefined</td><td></td></tr> </table>	V.Env	T.Env	num: undefined ans: undefined	
V.Env	T.Env				
num: undefined ans: undefined					

→

n: 2	<table> <tr> <th>V.Env</th><th>T.Env</th></tr> <tr> <td>num: 2 ans: 4 ...</td><td></td></tr> </table>	V.Env	T.Env	num: 2 ans: 4 ...	
V.Env	T.Env				
num: 2 ans: 4 ...					

E1

→ on return

V.E	
n: 2 Answer: 8 square2: 4	

```

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

```

During ~~Thread~~ execution phase  
n: 2  
~~new (2)~~ is passed  
over function &  
calculate value  
& put answer in  
ans: 4

Returns ans from V.Env  
to 0:0 & deletes execution  
context of function invocation

↓

Same for square(4)  
function invocation

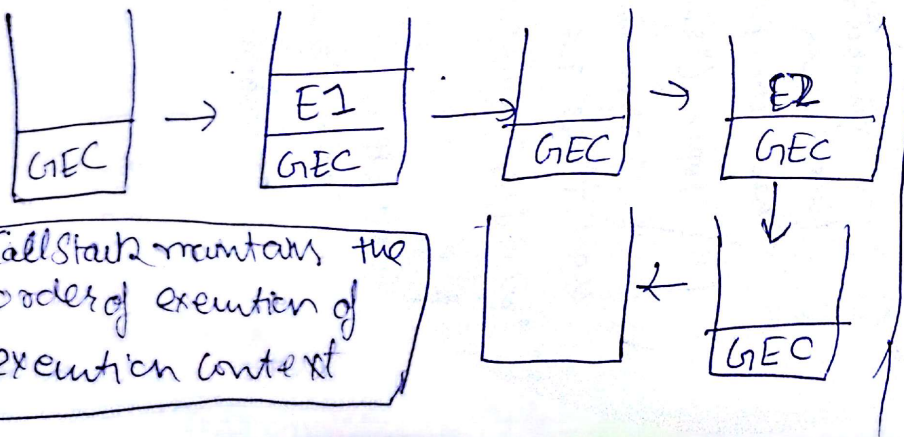
V.E							
square4: undefined	<table> <tr> <th colspan="2">E2</th></tr> <tr> <td></td><td></td></tr> <tr> <td></td><td></td></tr> </table>	E2					
E2							

GEC

V.E	
square4: 4	

GEC

→ JavaScript manages this via CallStack



① What is Lexical env?

Lexical env is create with Execution context.

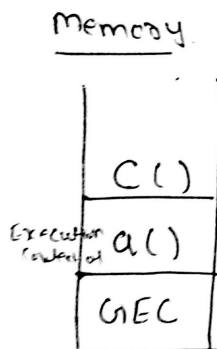
Lexical env = local memory + lexical environment of it's parent.

② What is Scope chain?

```

Ex:
var x = 20;
function a() {
  var b = 10;
  c();
  function c() {
    log(b);
    log(c);
  }
  a();
}

```



c will search var b in it's lexical environment, i.e first search for b in it's local & if not found then in it's parent scope.

For c: Search x in local scope then it's ~~parent scope~~ <sup>parent</sup> search on lexical env of it's parent. ~~if not~~

so ~~c()~~ <sup>access to</sup> c() in callstack has local scope & scope of it's parent.

Output →

10  
20