3 Video

HOW JS code is executed & call stack.

```
1   var n = 2;
2   function square(num)
5   {
3       var ans = num * num;
4       return ans;
5   }
6   var square2 = square(n);
7   var square4 = square(4);
10
```

parameter

argument

An Execution context is created

→ A global execution context will be created for the code is run

Execution context is created in 2 phases

Memory creation

> Memory is allocated to variables & functions in memory component of EC

Variable: A specific keyword which known as undefined (short a placeholder)

function: Exact code of function is copied to memory component.

code execution

> Whole code, run line by line. Every calculation & logical operation happens in phase only.

> Actual value to variable is assigned.

> In 6th line of code square function is invoked.

> function are hurt of JS They act as mini program.

Hence a execution context is also created. It will again go through 2 phase of EC creation

After phase 1 of EC

memory creation	Code Execution
n: undefined square: {...} square2: undefined square4: undefined	

After phase 2 of EC

memory creation	Code Execution
n = 2 square = {...}	local memory → Phase 1 memory: num: undefine, ans: undefine Code Execution: return:

return

Return will return value & control to execution context where the function was invoked

After phase 2 of execution context of square() function

memory	code
num: 2 ans: 4	num * num

⇒ For line 7 a brand new execution context will be created and similar process will happen.

After phase 2 of Global EC.

Memory	Code
n = 2 square : {...} square2 : 4 square4 : 16	memory: num: 4, ans: 16 code: return ans

Call stack maintains the order of execution of execution contexts

→ Call stack is also knows as following name:-

1. Execution Context Stack
2. Program Stack
3. Control stack
4. Runtime stack
5. Machine stack