JavaScript Execution Context & Call Stack

Deep Notes for Understanding JavaScript Internals

What is Execution Context?

An Execution Context is the environment in which JavaScript code is evaluated and executed. Every execution context contains memory creation, execution thread, and the this keyword.

Types of Execution Context

1. Global Execution Context (GEC)
2. Function Execution Context (FEC)
3. Eval Execution Context

Execution Phases

Memory Creation Phase: Variables are allocated memory and initialized as undefined, functions are stored with their full definition.

Execution Phase: Code is executed line by line and values are assigned.

Function Execution Context Example

When a function is invoked, a new execution context is created. It has its own memory phase and execution phase. After returning a value, the function execution context is removed from the call stack.

Call Stack (LIFO)

JavaScript uses a Call Stack to manage execution contexts. It follows the Last In First Out (LIFO) principle. The last function pushed into the stack is executed first and removed first.

Javascript Execution Context

{ } → Global EC
this

↳ Global Execution Context

↳ Function Execution Context

↳ Eval Execution Context

→ Execution Phase

① → Global Execution
↓
this

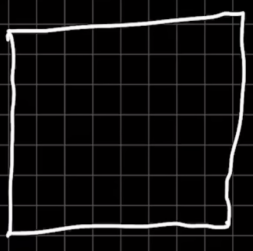
② Memory Phase

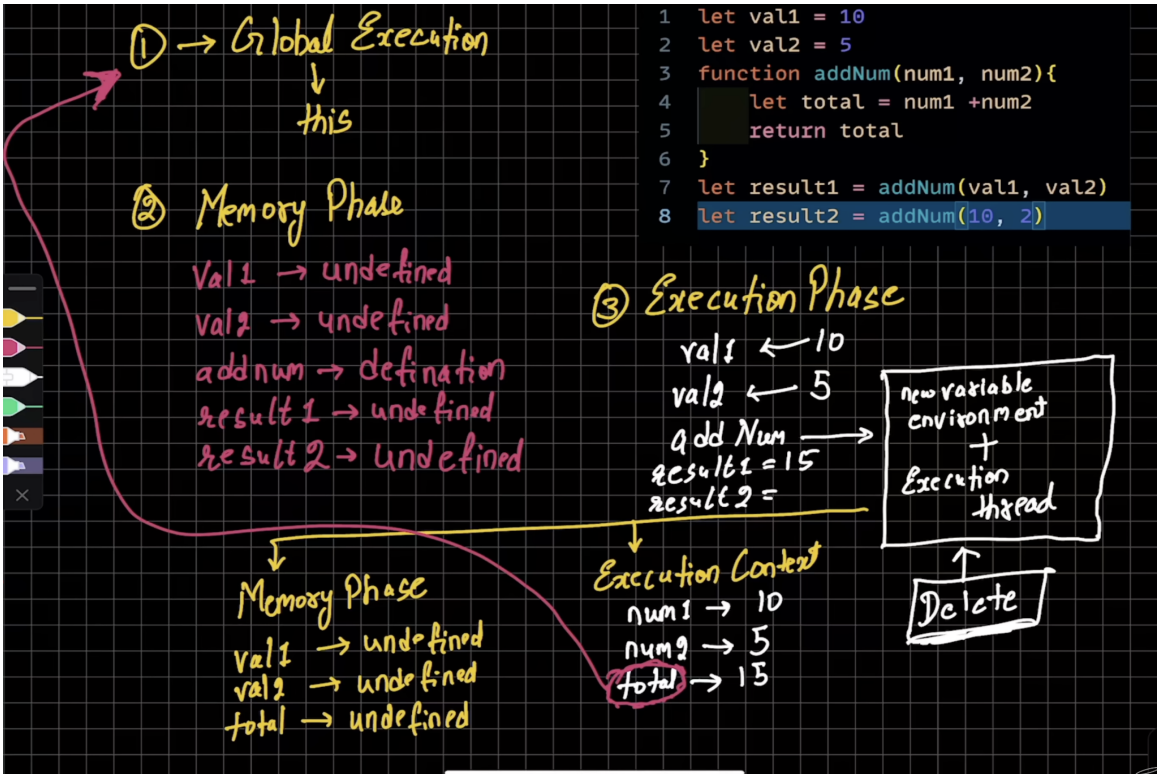
val1 → undefined
val2 → undefined
addnum → defination
result 1 → undefined
result 2 → undefined

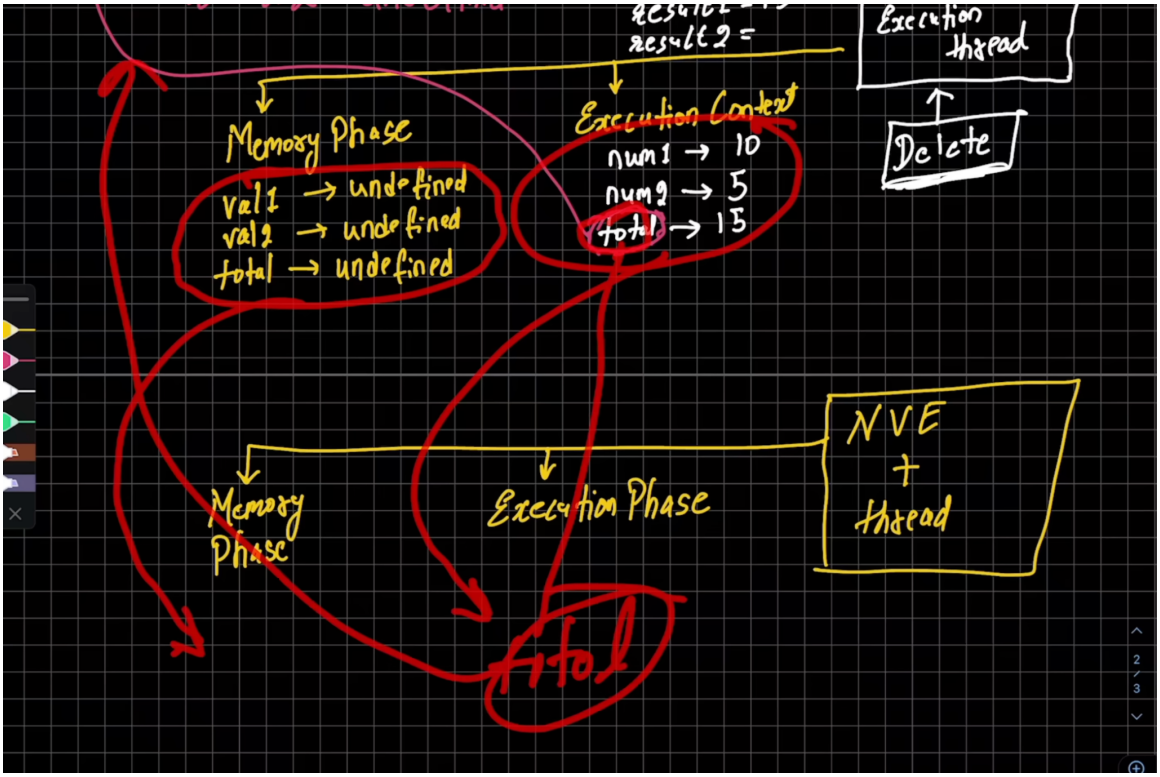
```
1  let val1 = 10
2  let val2 = 5
3  function addNum(num1, num2){
4      let total = num1 +num2
5      return total
6  }
7  let result1 = addNum(val1, val2)
8  let result2 = addNum(10, 2)
```

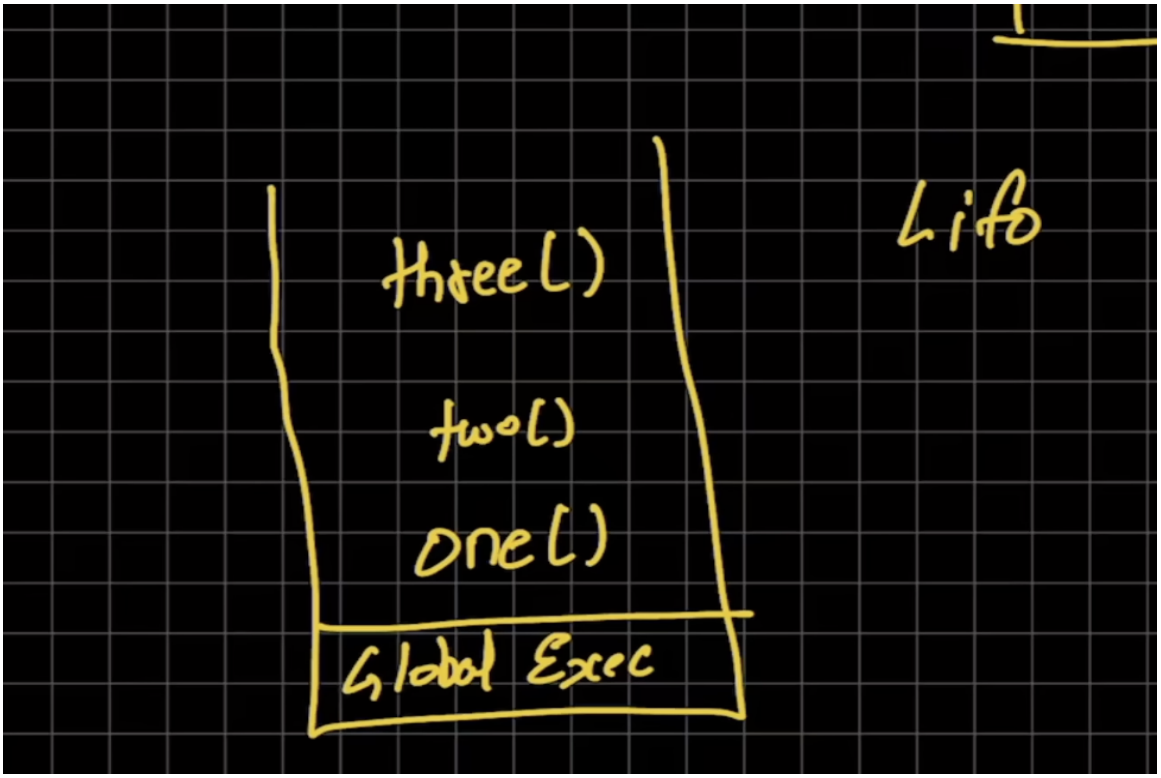
③ Execution Phase

val1 ← 10
val2 ← 5
add Num →




① → Global Execution
↓
this
② Memory Phase
val1 → undefined
val2 → undefined
addnum → defination
result1 → undefined
result2 → undefined
let val1 = 10
let val2 = 5
function addNum(num1, num2){
let total = num1 +num2
return total
}
let result1 = addNum(val1, val2)
let result2 = addNum(10, 2)
③ Execution Phase
val1 ← 10
val2 ← 5
add Num →
result1 = 15
result2 =
new variable environment + Execution thread
Delete
Memory Phase
val1 → undefined
val2 → undefined
total → undefined
Execution Context
num1 → 10
num2 → 5
total → 15


result2 =
Execution thread
Delete
Memory Phase
val1 → undefined
val2 → undefined
total → undefined
Execution Context
num1 → 10
num2 → 5
total → 15
NVE + thread
Memory Phase
Execution Phase
total


three()
two()
one()
Global Exec
Lifo

Key Takeaways

• JavaScript executes code using execution contexts
• Code runs in two phases: Memory Creation and Execution
• Each function call creates a new execution context
• Call Stack manages execution using LIFO principle
• Understanding execution context is crucial for mastering JavaScript

Prepared by: Akher Uddin Ahmed
Topic: JavaScript Internals & Execution Model