JavScript

20 May 2024 13:45

/* JavaScirpt Execution Context(How Code runs in JS) --

In Browser Execution Context is Window Object.

In JS Three Execution context are present--

1) Global Execution Context
2) Function Execution Context
3) Eval Execution Context

Every JavaScript code runs in two phases

1) Memory Creation Phase or Creation Phase --

    Here memory is allocated

2) Execution Phase

Let's Decode how this code will work --

Sample Code --

```
let val1 =10
let val2 =2
function addNum(num1,num2){
    let total = num1+num2
    return total
}
let result1 = addNum(val1,val2)
let result2 = addNum(10,2)
```

Decoding --

Step 1: Global Execution(It is allocated in this keyword)

Step 2: Memory phase(all variables are collected and given undefined value and functions names get the definition of function) --
        addNum=definition(function)
        val2=undefined
        val1=undefined
        result1=undefined
        result2=undefined
Step 3: Execution phase(In this every variable will get their value and)

        val1 <-- 10
        val2 <-- 5
        addNum --> Here the function will create a new

        Every time a function occur this box will be created
        And again new memory phase and execution phase will be created
        Only for that particular function.

New variable environment + Execution Thread

After all Work done it will deleted

        Memory Phase   Execution Context

        Num1 --> undefined   num1 --> 10
        Num2 --> undefined   num2 --> 5
        Total --> undefined   total --> 15

This total will return to Global Executional context

Step 4: Now for line no 8 of code that is

        let result2 = addNum(10,2)

        This box will be created and everything
        Happens in the same way.

New variable environment + Execution Thread

After all Work done it will deleted

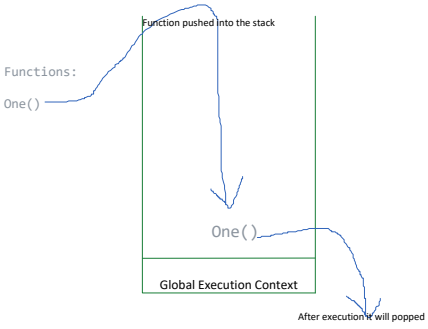
Memory Phase   Execution Context

Num1 --> undefined   num1 --> 10
Num2 --> undefined   num2 --> 2
Total --> undefined   total --> 12

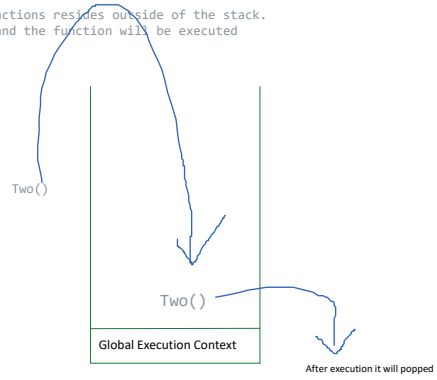
This total will return to Global Executional context

Call Stack -

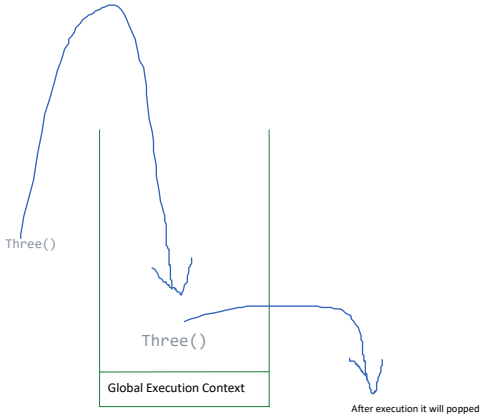
In Call stack a stack is created having Global Execution Context and all functions resides outside of the stack.
If they are called individually then one function will push into the stack and the function will be executed
And after execution it will pop out.



When Function one() will called

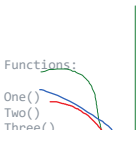


When Function two() will called



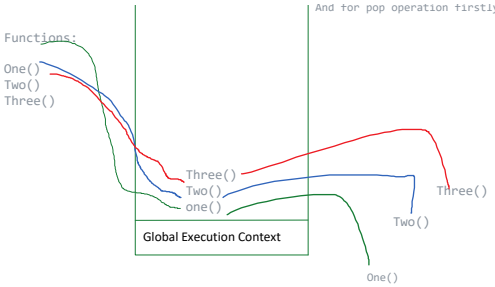
When Function three() will called

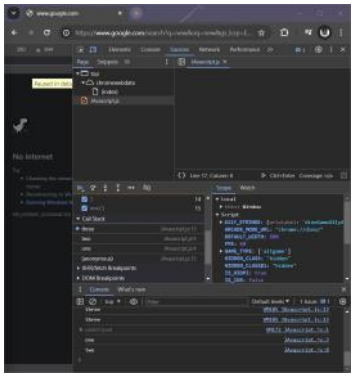
If we call a function into an another function then this process will be same and
Push operation and pop operation will work according to LIFO principle.

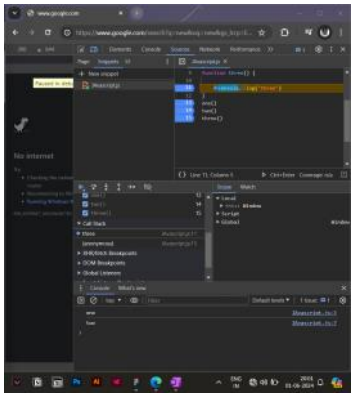


Here firstly one will called and it will push into the stack and if we call two into'
One then two will be pushed and again we call three into the two then three will be pushed
And for pop operation firstly three() will be popped then two() and In last one() will popped out.

And for pop operation firstly three() will be popped then two() and In last one() will popped out.







Here are two snippets that shows how Call Stacks works in browser