Javascript Execution Context

11 July 2024 22:46

The execution context in JavaScript is the environment in which the JavaScript code is executed. It is a fundamental concept that defines how the code is executed and what variables and functions are available to it. It executes one line of code at a time.

In a single thread, each operation waits for the last one to complete before executing.

Types of Execution Context:

1. **Global Execution Context:** This is the default execution context that is created when the JavaScript code first starts running. It represents the global scope. It's refers to the 'this' variable.

2. **Function Execution Context:** A new execution context is created whenever a function is called. This represents the local scope of the function.

3. **Eval Function Execution Context:** Any code executed within the eval() function also creates its own execution context.

Phases of Execution Context:

1. **Creation Phase:** In this phase, the JavaScript engine sets up the execution context. It:
   - Creates a global object (e.g. the window object in a browser)
   - Sets up memory space for variables and functions (known as hoisting)
   - Determines the value of the this keyword

2. **Execution Phase:** In this phase, JS engine executes the code line by line. Variables are assigned their values, functions are invoked, and the code is interpreted. The JS Engine reads the code and executes it one line at a time. This phase involves the following steps:
   - Assigning Values to Variables
   - Executing function and Code Blocks
   - Maintain the Call Stack

Now, let's discuss this program how it will run --->

1. **Global Execution Context is made and code will run through it ---> 'this' keyword**

2. **Memory Creation Phase --->**
   -> val1 = undefined
   -> val2 = undefined
   -> addNum = function definetion
   -> 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
   -> result1
   -> addNum ->

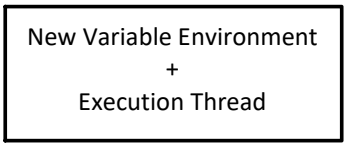
   New Variable Environment
   +
   Execution Thread

   a. **Memory Phase**
      --> num1 = undefined
      --> num2 = undefined
      --> total = undefined
   b. **Execution Phase**

--> num1 = val1(10)
--> num2 = val2(5)
--> total = 15

Therefore, **result1 = 15**

---> result2

--> addNum -> 

a. **Memory Phase**
   --> num1 = undefined
   --> num2 = undefined
   --> total = undefined
b. **Execution Phase**
   --> num1 = 10
   --> num2 = 2
   --> total = 12

Therefore, **result2 = 15**