Execution Context

03 May 2024 06:54 AM

JavaScript Execution Context

1. Global Execution Context
   --> Whenever the JS Code is run always Global Execution Context is made.
   --> It's refers to the 'this' variable
2. Functional Execution Context
3. Eval Execution Context

Note: JavaScript is a Single Threaded Language

~~JavaScript Code run into Two Phases~~

1. ~~Memory Creation Phase~~
   ~~--> In this phase, the JS Engine sets up the environment for the code to be executed. The "this" keyword is set to the value of the "this" object.~~
2. Execution Phase
   --> In this phase, the JS Engine executes the code line by line. The JS Engine reads the code and executes it one line at a time. This phase involves the following steps:
   a. Assigning Values to Variables
   b. Executing Function and Code Blocks.
   c. Maintain the Call stack

Now let's discussed this Program how it will run--->

1. Global Execution Context is made and code will run Through it.
2. Memory Creation Phase
   --> val1 = undefined
   --> val2 = undefined
   --> addNum = function definition
   --> result1 = undefined
   --> result2 = undefined
3. ~~Execution Phase~~
   --> val1 = 10
   --> val2 = 5
   --> result1
   -> addNum -> New variable environment + Execution Thread

```
95  let val1 = 10;
96  let val2 = 5;
97  function addNum(num1, num2) {
98      let total = num1 + num2;
99      return total;
100 }
101
102 let result1 = addNum(val1, val2);
103 let result2 = addNum(10, 2);
```

+

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 -> New Variable Environment

+

Execution Thread

a. Memory Phase

--> num1 = undefined

--> num2 = undefined

--> total = undefined

b. Execution Phase

--> num1 = 10

--> num2 = 2

--> total = 12

Therefore, result2 = 12