**JavaScript  Execution context**

**When** we run JavaScript code . first compile all the lines of code and then execute .

1. **Complication phase :**

Complication Phase complete in three phase

1. Tokenizing / Lexing : in this phase codes are breaks into small chunks.
2. Parsing : in this phase create a AST(Abstract syntax tree) with the help of small chunks of code.
3. Code generation : In this phase create a executable code with the help of AST.

Purpose of complication .

* Early error checking : in this process check any type of syntax error are exist or not in js code.
* Determining appropriates scope for variable (before the execution of first line of code compiler determine the variable scope)

**JavaScript  Execution context**

There are three type of execution Context

* Global Execution Context.
* Function Execution Context
* Eval Execution Context

JavaScript runs your js file in two phases:

When we try to execute the JavaScript code first it create a GEC .In GEC our code is execute in two phase

1. Memory Allocation Phase :- In Memory Allocation Phase three things happen

* First create a window object
* Second it takes all the variable and function and store inside this window object
* Third step initialized all the variable and function with undefined value

1. Code Execution Phase :- In Execution Phase execute the code.

**Example-1**

**let Val1=20;**

**let Val2=30;**

**function sumNum(num1, num2){**

**let total=num1+num2;**

**return total;**

**}**

**let Result1=sumNum(Val1, Val2);**

**let Result2=sumNum(10,5);**

Phase 1

1. Memory Allocation Phase :- In Memory Allocation Phase three things happen

* First create a window object
* Second it takes all the variable and function and store inside this window object
* Third step initialized all the variable and function with undefined value

**Global Execution Context.**

|  |  |
| --- | --- |
| **Memory Allocation Phase** | **Execution Phase** |
|  |  |

Val1=20 Execution Phase

Val2=30

sumNum(); Whenever call the function in that time create a separate execution context in stack for that function which is called. Once execution completed automatically delete the space from stack.

**Window object**

**Val1=>undefined**

**Val2=>undefined**

**sumNum=> function Definition**

**Result1=>undefined**

**Rsult2 =>undefined**

**Execution Context**

**New Variable Environment**

**+**

**Execution Thread**

se 3 : Execution Phase

**Execution Phase**

**Num1=20**

**Num2 = 30**

**Total=40(Total value return in global execution ) When finish the execution of function delete the execution context phase**

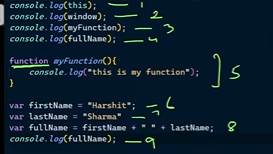
**Memory Allocation phase**

**Val1=>Undefined**

**Val2=>Undefined**

**total=>Undefined**

**Example-2**



**Global Execution Context.**

|  |  |
| --- | --- |
| Memory Allocation Phase | Code Execution Phase |
| 1. **Print window object:{}** 2. **Print window object:{}** 3. **Print function def** 4. **Assign value** 5. **Assign value** | window object : {}(window object is provide by the browser before the creation of GEC)  This : window object (When create GEC)  Firstname :Undefine =Harsit  Lastname : Undefine= Sharma  Lname : Undefine = Harsit Sharma  Myfunction : fun{} = |

**Javascript store these variable in object during the code execution phase that is called environment records**

Note

Lexical scope in JavaScript refers to the scope of variables and their accessibility during the compile time. It is based on where variables and blocks of scope are defined within the source code.