**Execution Context (EC):-**

JavaScript me **execution context** wo environment hota hai jisme code execute hota hai. Ye decide karta hai ke kis waqt kaunse variables, functions, aur objects accessible hain.

**Execution Context ke Types:**

**Mainly 2 type hai.**

1. **Global Execution Context (GEC)**:

JavaScript code global execution context me enter karta hai.

Is context me sab global variables aur functions define hote hain.

JS ka code Browser me window object, aur Node.js me global object global execution context hota hai.

Execution context bhi 2 phase me execute hota ha.

**I)Creation Phase**:

 Is phase me JavaScript engine variables aur functions ko memory me place karta hai, lekin unko abhi initialize nahi karta.

 Is process ko **hoisting** kehte hain. Iska matlab hai variables aur functions top pe chale jate hain.

**II)** **Execution Phase**:

 Is phase me variable initialize hote hai or JavaScript code line-by-line execute hota hai.

 functions call hote hain.

Jab function call hote hai to oos keliye naya function execution context tayar hota hai , jis me

a)New variable Enviroment

b)execution thread create hota hai.

1. **Function Execution Context (FEC)**:

Jitni bar bhi fnc ko call kiya gayenga utni bar uska ek alag se fnc execution context create honga jo ke same process follow karenge un sabhi variable keliye jo oos function me declair kiye gaye hai.

**I)Creation Phase**:

**II)** **Execution Phase**:

Jo function keliye execution context create kiya tha o oos ka kam khatam honeke bad Automatically delete bhi ho jata hai.

### **Execution Stack:**

* Jab multiple functions call hote hain, to har function ka execution context ek stack (pile) ke tarah maintain hota hai, jise **Call Stack** kehte hain.
* Pehle Global Execution Context stack me aata hai, aur phir jitne bhi function calls hote hain, unka context stack ke top pe push hota hai. Jab function complete hota hai, to wo context stack se pop ho jata hai.

Niche diye huve example se samz te hai execution kaise hota hai.

