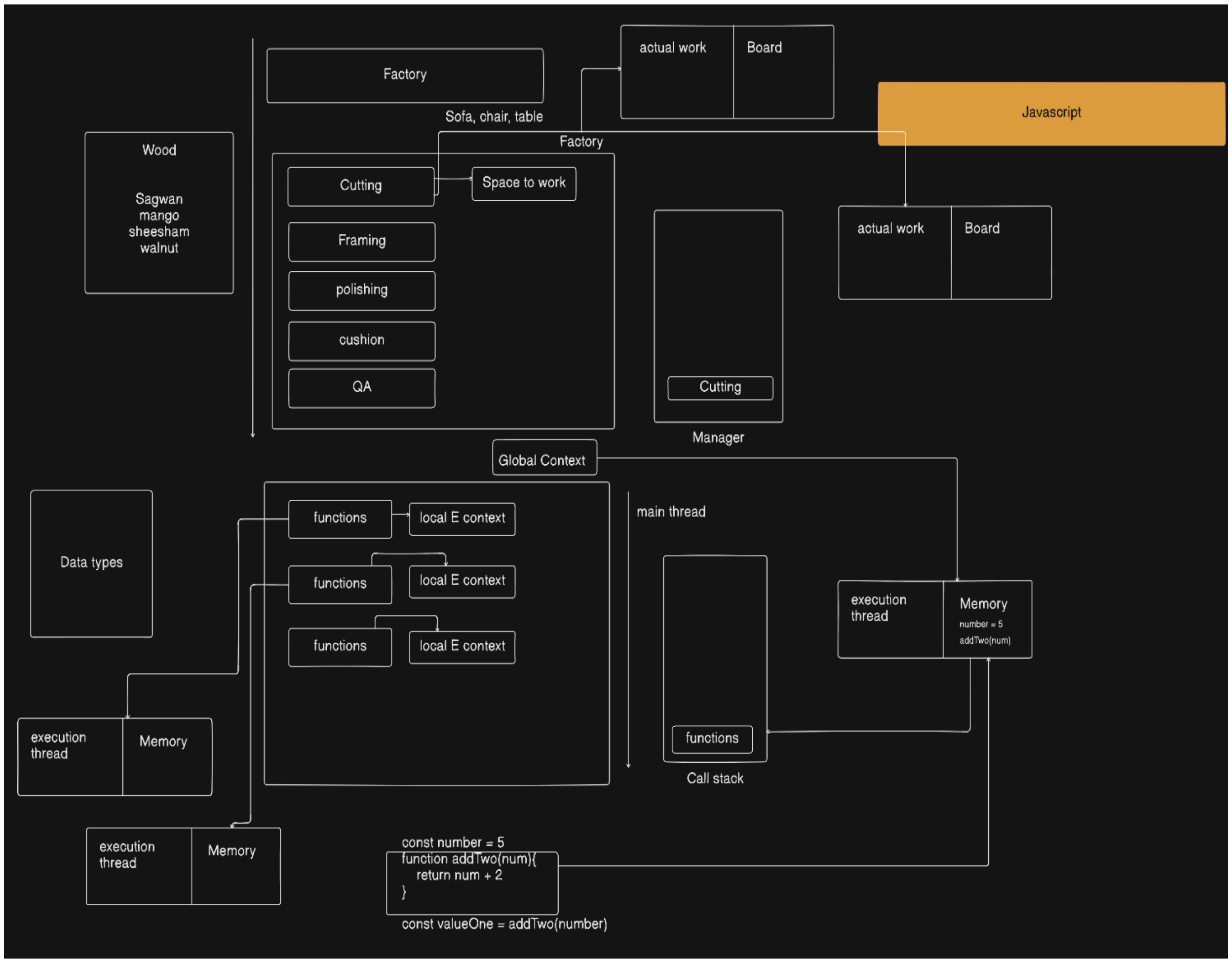
7 FEB 2026(ROSE-DAY)

CHAI AUR JAVACRIPT(DAY-1)



```
const number = 5
function addTwo(num){
    return num + 2
}
const valueOne = addTwo(number)
```

🏭 1. Factory Example → JavaScript Mapping

Factory Concept	JavaScript Concept
Wood types	Data Types

Factory	Global Execution Context
Departments (Cutting, Polishing etc)	Functions
Manager	Call Stack
Worker Notes	Local Execution Context
Notice Board	Global Memory
Work Process	Execution Thread

🌳 2. Wood = Data Types

Factory mein alag wood types:

- Sagwan
- Mango
- Sheesham
- Walnut

👉 JS mein ye represent karte hain:

- Number
- String
- Boolean
- Object
- Array

➡️ Raw material = Data Types

🌍 3. Global Execution Context (Main Factory)

👉 Jab JS program start hota hai
👉 Sabse pehle **Global Execution Context (GEC)** banta hai

Isme 2 cheeze hoti hain:

🧠 Memory Phase

Sab variables & functions store hote hain

⚙️ Execution Phase

Code line by line run hota hai

🧵 4. Main Thread (Single Worker System)

JavaScript = **Single Threaded**

👉 Ek time pe ek hi kaam karega
👉 Top → Bottom code scan karega

📦 5. Memory + Execution Thread

Har Execution Context ke paas hota hai:

🧠 Memory

Variables store

⚙️ Execution Thread

Code run karta hai

🏭 6. Functions = Factory Departments

Example:

- Cutting
- Framing
- Polishing
- Cushion
- QA

👉 JS mein:

- Har function = Separate Department
- Har function = Apna Local Execution Context

📒 7. Local Execution Context (Worker Notes)

Jab function call hota hai:
✔ Apni memory banata hai
✔ Apna execution thread hota hai
✔ Local variables store karta hai

👨‍💼 8. Call Stack = Manager

👉 Kaun kaam karega decide karta hai
👉 LIFO Rule (Last In First Out)

Example:

```
Global
   ↓
```

```
addTwo()
  ↓
Return → remove from stack
```

🔁 9. Flow (Factory Style)

1️⃣Raw Material (Data Types) aata hai
2️⃣Global Factory ready hoti hai
3️⃣Manager (Call Stack) task assign karta hai
4️⃣Department (Function) kaam karta hai
5️⃣Notes (Local Memory) store hoti hai
6️⃣Final result Global pe jata hai

💻 10. Real JS Example

```
const number = 5

function addTwo(num){
  return num + 2
}

const valueOne = addTwo(number)
```

🧠 Memory Creation Phase

Variable	Value
number	5
addTwo	function definition
valueOne	undefined

⚙️ Execution Phase

👉 number = 5 assign
👉 function ready
👉 addTwo(number) call

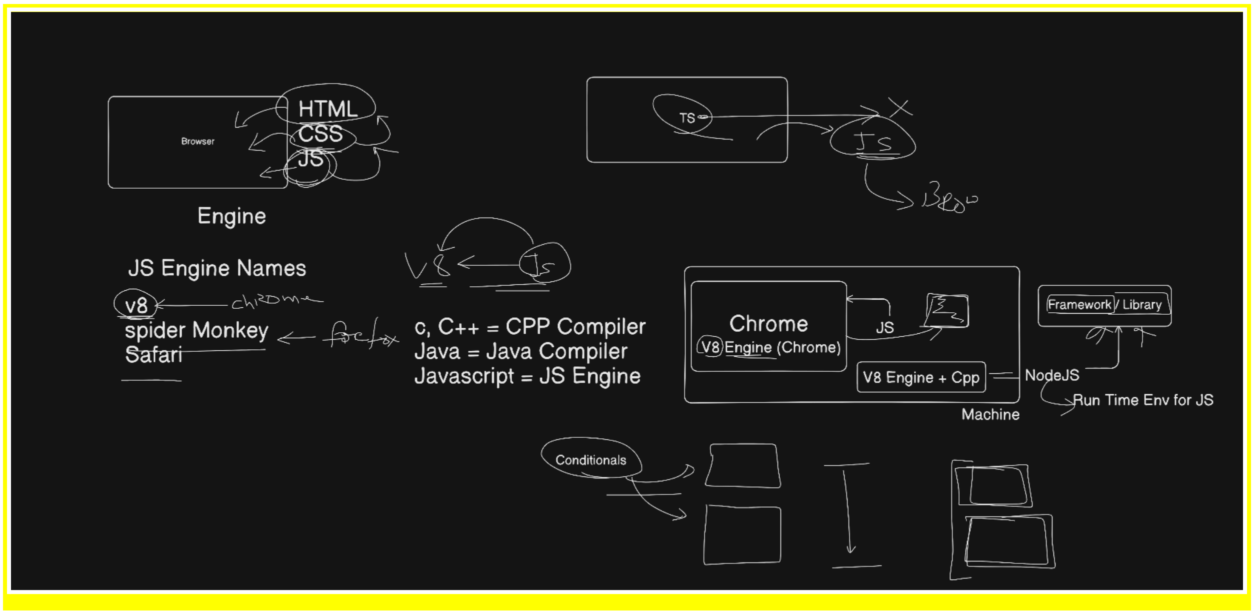
📦 Function Local Memory

Variable	Value
num	5

Return → 7

🌍 Back to Global

```
valueOne = 7
```



🌐 1. Browser Ka Kaam

Browser ko milta hai:

- HTML → Structure banata hai
- CSS → Design deta hai
- JavaScript → Logic / Functionality deta hai

👉 Browser ke andar ek **Engine** hota hai jo JS run karta hai.

⚙️ 2. JavaScript Engine

👉 JS Engine ka kaam = JavaScript ko Machine Code me convert karna

Famous JS Engines:

- V8 → Chrome
- SpiderMonkey → Firefox
- JavaScriptCore → Safari

💻 3. Languages Kaise Run Hoti Hai

- C / C++ ⟶ C++ Compiler
- Java ⟶ Java Compiler + JVM
- JavaScript ⟶ JS Engine

🌍 4. Browser Me JS Kaise Chalta Hai

👉 Chrome me V8 Engine hota hai
👉 Isliye Chrome JavaScript run kar sakta hai

🖥️ 5. System Me Direct JS Kyu Nahi Chalta

👉 System me JS run karne ke liye sirf engine enough nahi hota
👉 System access chahiye (file, network, OS work)

🚀 6. NodeJS Kya Hai

👉 NodeJS = V8 Engine + System bindings (C/C++)

✔ Browser ke bahar JS run karta hai
✔ Backend development me use hota hai
✔ Local machine pe JS run kara sakte hai

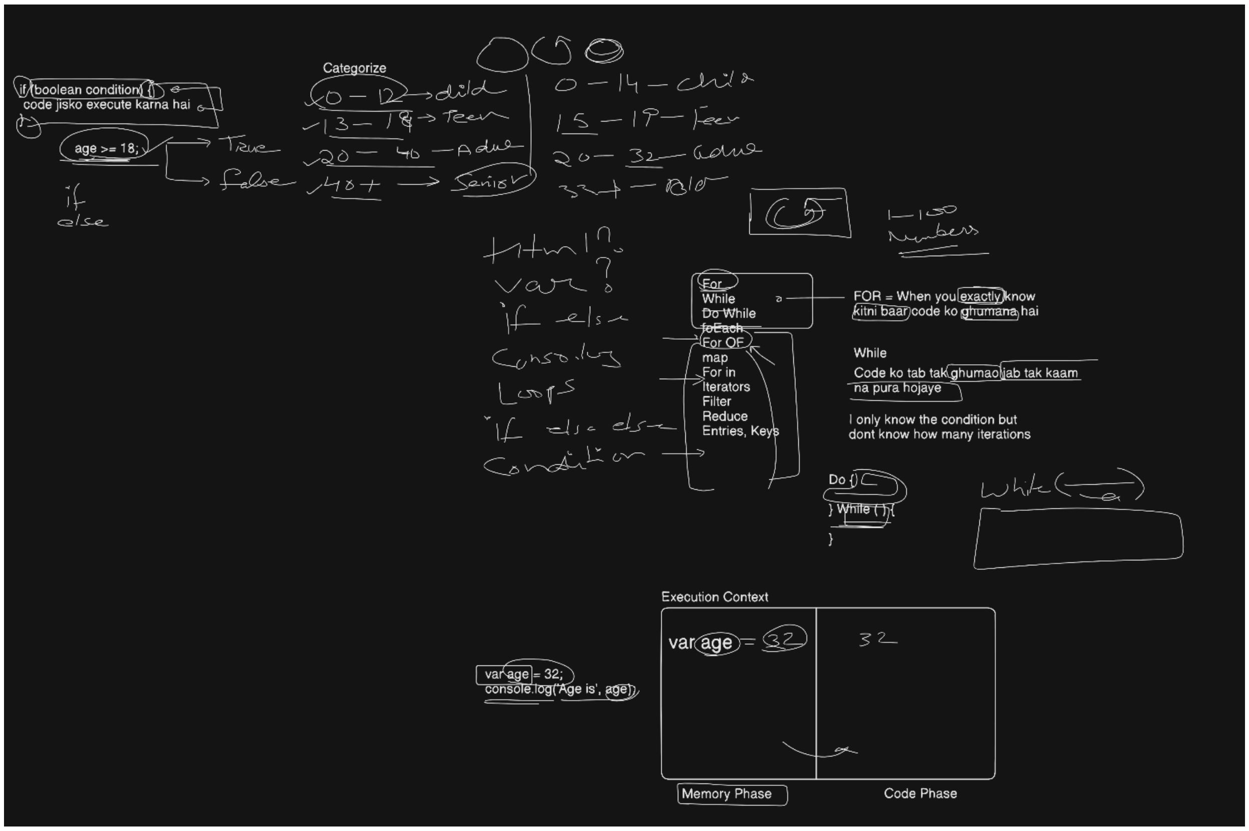
📦 7. NodeJS Kya Nahi Hai

❌ Framework nahi
❌ Library nahi

✔ Runtime Environment hai

⭐ Final Samajh

👉 Browser → JS run via JS Engine
👉 NodeJS → JS run via Runtime Environment (System pe)



if (boolean condition) {
code jisko execute karna hai
}

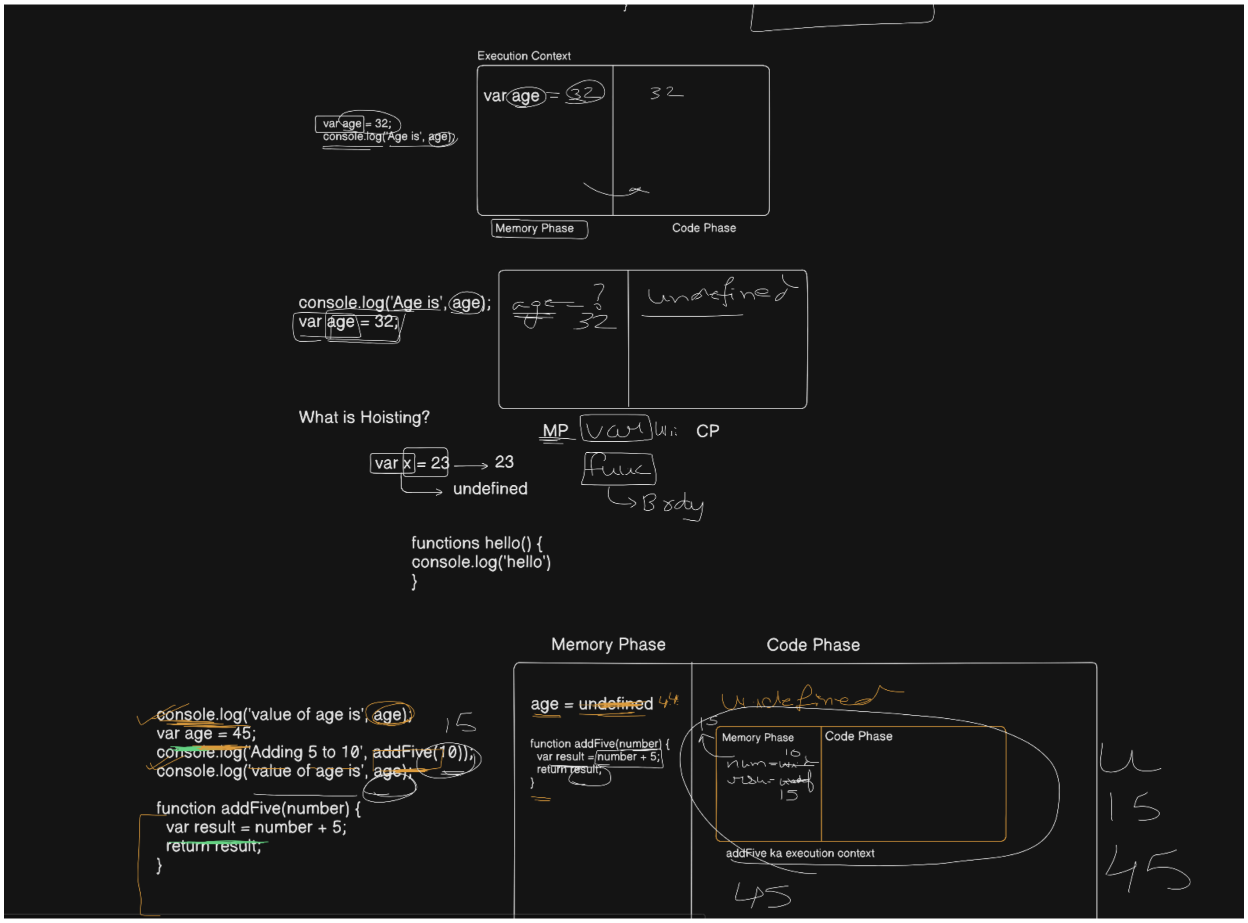
FOR = When you exactly know kitni baar code ko ghumana hai

While
Code ko tab tak ghumao jab tak kaam na pura hojaye

I only know the condition but dont know how many iterations

```
Do {}
} While ( ){
}
```

```
var age = 32;
console.log('Age is', age);
```



```
var age = 32;
console.log('Age is', age);
```

```
console.log('Age is', age);
var age = 32;
```

```
functions hello() {
console.log('hello')
}
```

```
console.log('value of age is', age);
var age = 45;
console.log('Adding 5 to 10', addFive(10));
console.log('value of age is', age);

function addFive(number) {
  var result = number + 5;
  return result;
}
```

```
function addFive(number) {
  var result = number + 5;
  return result;
}
```

🔹 1️⃣ **Variables (Starting with `var`)**

👉 Variable = Data store karne ke liye use hota hai
👉 Starting me mostly `var` se samjhaya jata hai

Example

```
var age = 25
```

✔ Value store hoti hai
✔ Change ho sakti hai

🔹 2️⃣ **JavaScript is Loosely Typed Language**

👉 Same variable me different type value aa sakti hai

Example

```
var x = 10
x = "Hello"
x = true
```

👉 Type define karna mandatory nahi hota

🔹 3️⃣Functions

👉 Function = Reusable code block
👉 Set of instructions ka wrapper

Example

```
function add(a,b){
 return a + b
}
```

✔ Code reuse hota hai
✔ Code clean hota hai

🔹 4️⃣Conditional Statements

👉 Boolean (True / False) pe kaam karta hai

✅ if → else if → else

👉 Multiple conditions check kar sakte hai

Example

```
if(age >= 18){
 console.log("Adult")
}
else if(age >= 13){
 console.log("Teen")
}
else{
 console.log("Child")
}
```

✔ Pehle if check hota hai
✔ Agar false → else if check hota hai
✔ Sab false → else run hota hai

🔹 5️⃣ **Loops**

👉 Same code multiple times run karne ke liye

🔁 **For Loop**

Order:
1️⃣ Initializer
2️⃣ Condition
3️⃣ Increment

```
for(let i=0; i<5; i++){
 console.log(i)
}
```

🔁 **While Loop**

👉 Pehle condition check
👉 Fir code run

```
while(i < 5){
 console.log(i)
 i++
}
```

🔁 Do While Loop

👉 Pehle code run
👉 Fir condition check

```
do{
 console.log(i)
 i++
}while(i < 5)
```

🔹 6️⃣Hoisting

👉 Variable aur function declarations memory phase me store ho jate hai
👉 Isliye declare hone se pehle access ho sakte hai

Variable Hoisting Example

```
console.log(a)
var a = 10
```

Output → `undefined`

✔ Memory Phase → a = undefined
✔ Code Phase → a = 10

🔹 7️⃣Memory Phase vs Code Phase

🧠 Memory Phase

✔ Variables → undefined
✔ Functions → Full function store

⚙️ Code Phase

✔ Value assign hoti hai
✔ Code execute hota hai

🔹 8️⃣ Debugger (VS Code)

👉 Debugger se check kar sakte hai:
✔ Code step by step execution
✔ Variable values
✔ Memory → Code flow

👉 Breakpoints laga ke execution dekh sakte hai

⭐ Final Flow (Sequence)

Variables → Loosely Typed → Functions →
Conditionals (if → else if → else) →
Loops → Hoisting →
Memory Phase vs Code Phase → Debugger