✨ LECTURE 18 – THIS KEYWORD IN JAVASCRIPT

🔷 GLOBAL OBJECT (ROOT OF JS)

👉 Global Object Kya Hota Hai?

JavaScript ka code chalne ke liye ek **execution environment** chahiye hota hai.
Is environment ka **sabse bada object** hota hai → **Global Object.**

📌 Environment ke hisaab se naam:

- 🌐 **Browser** → `window`
- 🟢 **Node.js** → `global`
- 🌍 **Universal (ES2020+)** → `globalThis`

❓ console.log(), Math.random() kaha se aate hain?

🧠 Comparison:

- **C++** → `#include <iostream>`
- **JavaScript** → Sab kuch **by default Global Object ke andar hota hai**

```
console.log("Hello World");

console.log(Math.random());
```

👉 `console`, `Math`, `setTimeout`, `setInterval`
➡ sab **Global Object ki properties** hain.

⚠️ Global Variables Rule

- `var` → **global object ka part** ban jata hai
- `let / const` → **global object ka part nahi bante**

✅ Universal Access

```
window.Math.random();      // Browser

global.Math.random();      // Node.js

globalThis.Math.random(); // Har jagah
```

💡 **Interview Tip**
❓ *console.log kaise kaam karta hai?*
✅ `console` global object ka part hai, aur `log()` uska method.

🔷 THIS KEYWORD (CONTEXT BASED)

👉 **this ka matlab hota hai:**

> “Current execution context ka owner kaun hai?”

⚠️ `this` ki value **call hone ke tareeke pe depend karti hai**, na ki likhne pe.

🔶 a) Global Context

```
console.log(this);
```

📤 Output:

- Browser → `window`
- Node.js → `{}`

```
console.log(globalThis);
```

👉 Universal solution.

🔶 b) Inside a Function

🔷 Normal Function

```
function greet() {
  console.log(this);
}
greet();
```

📤 Output:

- Browser → `window`
- Node.js → `global`

🔷 Strict Mode

```
"use strict";
function greetStrict() {
  console.log(this);
}
greetStrict();
```

📤 Output:

```
undefined
```

🧠 Reason:

> Strict mode me JS **default binding allow nahi karta**

🔶 c) Inside an Object Method

```
const obj = {
  name: "Harshal",
  sayName() {
```

```
    console.log(this.name);
  }
};

obj.sayName();
```

📤 Output:

```
Harshal
```

👉 **Rule:**
`this` → **jis object ne method call kiya**

🔶 d) Arrow Functions & this

🚫 Arrow Function ka apna this nahi hota

Arrow function **lexical this** use karta hai (parent se inherit).

```
const obj = {
  name: "Harshal",
  arrow: () => {
    console.log(this);
  }
};

obj.arrow();
```

📤 Output:

- Browser → `window`
- Node.js → `{}` / `global`

❌ `obj` nahi aayega

✅ Lexical Binding Example

```
const obj = {
  name: "Rohit",
  greet() {
    const arrow = () => console.log(this.name);
    arrow();
  }
};

obj.greet();
```

📤 Output:

```
Rohit
```

🧠 Reason:

- Arrow → parent (`greet`) ka `this` use karta hai

🔶 e) this Inside Class / Constructor

```
class Person {
  constructor(name) {
    this.name = name;
  }
}
const p1 = new Person("Saurav");
console.log(p1.name);
```

📤 Output:

```
Saurav
```

👉 Constructor me `this` → **naya object**

🔶 f) this in setTimeout

```
setTimeout(function () {
  console.log(this);
}, 1000);
```

📤 Output:

- Browser → `window`

```
setTimeout(() => {
  console.log(this);
}, 1000);
```

📤 Output:

- Lexical parent ka `this`

🔶 g) this in Event Listeners

```
btn.addEventListener("click", function () {
  console.log(this);
});
```

📤 Output:

```
button element
```

```
btn.addEventListener("click", () => {
  console.log(this);
});
```

📤 Output:

```
lexical parent
```

📌 IMPORTANT TAKEAWAYS

✓ Arrow function ka **apna this nahi hota**
✓ Regular function ka this → **call pe depend**
✓ `var` → global object ka part
✓ `let / const` → global object ka part nahi
✓ Browser & Node.js me global object alag hota hai
✓ Universal solution → `globalThis`

🔁 SUMMARY – EK NAZAR ME

- 🌍 Global Object:
  - Browser → `window`
  - Node → `global`
  - Universal → `globalThis`
- 🧠 `this` behaviour:
  - Global → window / global
  - Function → global (strict me undefined)
  - Object method → calling object
  - Arrow → parent ka this
  - Class/constructor → new object

INTERVIEW MASTER LINE

> **“this is not where the function is written, it’s where the function is called.”**