✨ LECTURE 12 — FUNCTIONS IN JAVASCRIPT

📘 **JavaScript Core Concept**
🧠 *Definition → Why → How → Example → Rules*

⬛ 1 WHAT IS A FUNCTION? (DEFINITION)

🔷 Definition

Function ek **reusable block of code** hota hai jo
- 🔶 ek specific task perform karta hai
- 🔶 inputs (parameters) le sakta hai
- 🔶 output (return value) de sakta hai

📌 **Simple words me**
Function = *machine*
Input do → processing hoti hai → output milta hai

🔷 Real-Life Example

Calculator:

- Input → 2, 3
- Process → addition
- Output → 5

👉 Ye kaam code me **function** karta hai.

⬛ 2 FUNCTION DECLARATION

🔷 Definition

Function Declaration me hum `function` keyword ka use karke function ko **direct define** karte hain.

🔷 Syntax

```
function greet() {

  console.log("Hello Codes Army");

}
```

🔷 Function Call

```
greet();
```

🔷 Characteristics

✅ Readable
✅ Beginner-friendly
✅ Hoisting support karta hai

⬛ FUNCTION WITH PARAMETERS & ARGUMENTS

🔷 Definition

Parameters → function ke input variables
Arguments → function call ke time di gayi actual values

```
function sum(a, b) {   // parameters
  return a + b;
}

let result = sum(3, 5); // arguments
console.log(result);   // 8
```

🧠 Yaad Rakho

- Parameter = box
- Argument = value jo box me dali

⬛ 3 FUNCTION EXPRESSION

🔷 Definition

Jab function ko **variable me store** kar dete hain
use **Function Expression** kehte hain.

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

🔷 Characteristics

✅ Function ko value ki tarah treat kar sakte ho
❌ Hoisting support nahi karta

📌 Mostly callbacks me use hota hai.

⬛ ⚠️ IMPORTANT RULE — `return`

🔷 Definition

`return` function ka **final statement** hota hai
Ye value bhi bhejta hai aur function ko stop bhi karta hai.

```
const fun = function() {
  console.log("Hello Codes Army");
  return "money";
  console.log("Ye kabhi execute nahi hoga");
};
```

🧠 Rule

`return` ke baad ka code **kabhi execute nahi hota**

⬛ 4 ARROW FUNCTION (ES6+) — COMPLETE DEPTH

🔷 4.1 Arrow Function — DEFINITION

Arrow Function ek **short & modern syntax** hai
jo function likhne ka cleaner tareeka deta hai.

📌 Ye **function expression ka hi advanced form** hai.

🔷 4.2 Why Arrow Functions Were Introduced?

Problems with normal functions:

- Zyada code
- `this` confusing hota tha

👉 Arrow functions:
✅ Short syntax
✅ Readable
✅ Callbacks me perfect

🔷 4.3 Basic Arrow Function

```
const sum = (a, b) => {

  return a + b;

};
```

🧠 Breakdown:

- `(a, b)` → parameters
- `=>` → arrow (function indicator)
- `{}` → function body

🔷 4.4 One-Line Arrow Function

🔷 Definition

> Agar function body me **sirf ek statement** ho
> to `{}` aur `return` dono hata sakte ho.

```
const sum = (a, b) => a + b;
```

📌 JS automatically return kar deta hai.

🔷 4.5 Single Parameter Arrow Function

🔷 Definition

Agar sirf **ek parameter** ho
to `()` bhi optional hote hain.

```
const cube = a => a * a * a;
```

🧠 Rules Summary:

- 1 parameter → `()` optional
- 1 statement → `return` optional

🔷 4.6 Arrow Function vs Normal Function (Conceptual)

Feature	Normal Function	Arrow Function
Syntax	Long	Short
this	Own `this`	Parent `this`
Hoisting	Yes (declaration)	No
Callbacks	Okay	Best

📌 **Interview Tip**
Arrow function ka **apna this nahi hota —** ye parent se leta hai.

⬛ 5 REST PARAMETERS (`...`)

🔷 Definition

Rest Parameter multiple arguments ko
ek **single array** me collect karta hai.

```
const sum = function(...numbers) {
  let total = 0;
  for (let i = 0; i < numbers.length; i++) {
    total += numbers[i];
  }
  return total;
};

sum(2, 3);        // 5
sum(2, 3, 4);     // 9
```

🧠 ...numbers = [2, 3, 4]

⬛ 6 FUNCTIONS WITH OBJECTS

🔷 Definition

JavaScript me objects ko directly function me pass kar sakte ho.

```
let obj = { name: "Sourav", age: 20 };

function show(user) {
  console.log(user.name, user.age);
}
```

🔷 Destructuring in Function (BEST PRACTICE)

```
function show({ name, age }) {
  console.log(name, age);
}
```

📌 Clean + readable + industry standard.

⬛ 7 FUNCTIONS ARE OBJECTS (VERY IMPORTANT)

🔷 Definition

JavaScript me **functions bhi objects hote hain**
isliye unke paas prototype hota hai.

```
const add = (a, b) => a + b;


add.__proto__ === Function.prototype; // true

Function.prototype.__proto__ === Object.prototype; // true
```

⬛ 8 PROTOTYPE CHAIN (FLOW)

```
Function

  ↑

Function.prototype

  ↑

Object.prototype

  ↑

null
```


📌 Isi wajah se functions ke paas:

- call()
- bind()
- apply()
  jaise methods hote hain.

⬛ FINAL SUMMARY — EK READ ME ⬛

- ✅ Function = reusable logic block
- ✅ Declaration, Expression, Arrow — 3 main types
- ✅ Arrow functions = short + modern
- ✅ `return` ke baad code stop
- ✅ Rest parameter = multiple args handle
- ✅ Functions internally objects hote hain
- ✅ Prototype chain follow hoti hai

⬛ FINAL THOUGHT

🧠 **Agar functions clear hain → JavaScript 70% clear hai**