+91 96622 16697
jay@frameboxxers.com
F-201, Shilp Square-B, Nr. Shreeji Tower,
Drive-in Road, Vastrapur, Ahmedabad - 380015

📘 Lecture 5: Functions in JavaScript

🧠 What is a Function?

A **function** is a reusable block of code designed to perform a particular task.
Think of it as a **recipe**: you define the steps once and call it whenever needed — without rewriting the instructions.

✅ Why Use Functions?

- **Reusability**: Write once, use many times
- **Organization**: Break large problems into small chunks
- **Avoid Repetition**: DRY (Don’t Repeat Yourself) principle
- **Better Debugging**: Errors are easier to locate and fix

🧩 Function Syntax

javascript
CopyEdit

```javascript
function functionName(parameters) {
  // code block
}
```

🔸 Example:

javascript
CopyEdit

```javascript
function greet(name) {
  console.log("Hello, " + name + "!");
}
greet("Alice"); // Output: Hello, Alice!
greet("Bob");   // Output: Hello, Bob!
```

📌 Function Components

Part	Description
function	Keyword to declare a function
functionName	Name to identify the function
parameters	Inputs that the function accepts
return	Sends back a result from the function

🔄 Function With Return Value

You can return values using return.

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

let sum = add(5, 3);
console.log(sum); // Output: 8
```

📝 Function Without Parameters

javascript
CopyEdit
```
function sayHello() {
  console.log("Hello, world!");
}
sayHello(); // Output: Hello, world!
```

💡 Function Expression (Storing Function in Variable)

javascript
CopyEdit
```
const multiply = function(x, y) {
  return x * y;
};

console.log(multiply(4, 5)); // Output: 20
```

⚡ Arrow Functions (ES6 Feature)

A shorter way to write functions.

javascript
CopyEdit

```
const square = (n) => {
  return n * n;
};

console.log(square(6)); // Output: 36
```

👉 If it has only one line and one parameter, you can write:

javascript
CopyEdit

```
const double = n => n * 2;
```

🔁 Function Calling Another Function

Functions can call each other.

javascript
CopyEdit

```
function greetUser(name) {
  let message = buildGreeting(name);
  console.log(message);
}

function buildGreeting(name) {
  return "Hi " + name + ", welcome!";
}
greetUser("Alex"); // Output: Hi Alex, welcome!
```

⚠️ Parameters vs Arguments

- **Parameters**: placeholders in function definition
- **Arguments**: actual values passed when calling the function

javascript
CopyEdit

```
function sayHi(name) {   // name is a parameter
  console.log("Hi, " + name);
}
```

```
sayHi("Sarah");        // "Sarah" is the argument
```

🧠 Scope in Functions

Variables defined **inside** a function are not accessible outside it.

javascript
CopyEdit

```
function testScope() {
  let x = 10;
  console.log(x);
}
testScope();      // 10
// console.log(x); // ❌ Error: x is not defined
```

🔁 Real-World Analogy

Imagine a coffee machine (function). You give it input (coffee type), press a button (call the function), and it gives you output (coffee) — every time without changing the internals.

🧩 Practice Tasks

1. Create a function that returns the square of a number.
2. Write a function that checks if a number is even or odd.
3. Build a function to calculate the factorial of a number.
4. Create a greeting function that returns "Good morning, <name>".
5. Convert a normal function into an arrow function.

📌 Summary

Concept	**Description**
Function	Block of reusable code
Parameters	Inputs in function definition
Arguments	Actual values passed to the function

return	Sends a value back to the caller
Arrow Function	Concise syntax for writing functions

Functions are the **foundation of structured programming**. They help make your code cleaner, reusable, and easier to debug.