Return statements

Let's explore `return` **statements** in JavaScript — one of the most fundamental concepts for controlling function output and flow.

🔷 What Is a `return` Statement?

The `return` statement is used to:

1. **Return a value** from a function.
2. **Stop the execution** of the function immediately.

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

✅ Basic Example

```
function greet() {
  return "Hello!";
}

let message = greet();
console.log(message); // Output: Hello!
```

- `greet()` returns `"Hello!"`
- That value is stored in `message`

⛔ `return` Ends Function Execution

```
function example() {
  console.log("Before return");
  return "Done";
  console.log("After return"); // ❌ This will never run
}

example();
// Output:
// Before return
```

> Anything after a return is ignored.

🧠 Returning Expressions

You can return any **valid expression**, not just plain values:

```
function square(x) {
  return x * x;
}

console.log(square(5)); // 25
```

🧾 Implicit vs Explicit Return

❌ Regular functions require `return` explicitly:

```
function multiply(x, y) {
  x * y;       // No return — returns undefined
}
```

✅ Arrow functions can return implicitly (if one-liner):

```
const multiply = (x, y) => x * y;
console.log(multiply(3, 4)); // 12
```

But for multiple lines, you **must** use `return`:

```
const multiply = (x, y) => {
  const result = x * y;
  return result;
};
```

🚫 What if You Don't Use `return`?

If no `return` is written, the function returns `undefined`:

```
function noReturn() {
  let x = 10;
}

console.log(noReturn()); // undefined
```

✅ Returning Multiple Values

JavaScript functions can't return multiple values directly, but you can **return them in arrays or objects**:

```
function getUser() {
  return ["Abhi", 22];
}
// OR
function getUserObj() {
  return { name: "Abhi", age: 22 };
}
```

🔄 Return from Nested Functions

You can use `return` inside a function within another function:

```
function outer() {
  function inner() {
    return "Hello from inner";
  }
  return inner();
}

console.log(outer()); // Hello from inner
```

🧾 Summary

Feature	Description
`return` keyword	Exits a function and optionally returns a value
Stops execution	Yes — any code after `return` won't run
Returns default	`undefined` if no value is returned
Arrow function support	Supports both implicit and explicit return
Useful for	Calculations, conditionals, early exits, etc.

Would you like to go over **Function Scope and Local Variables** next?