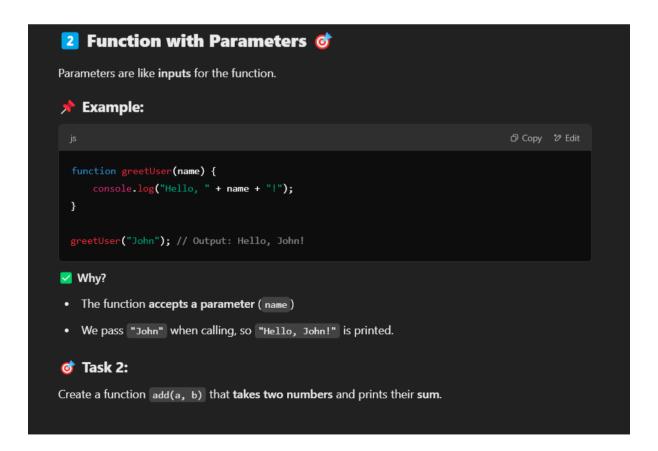
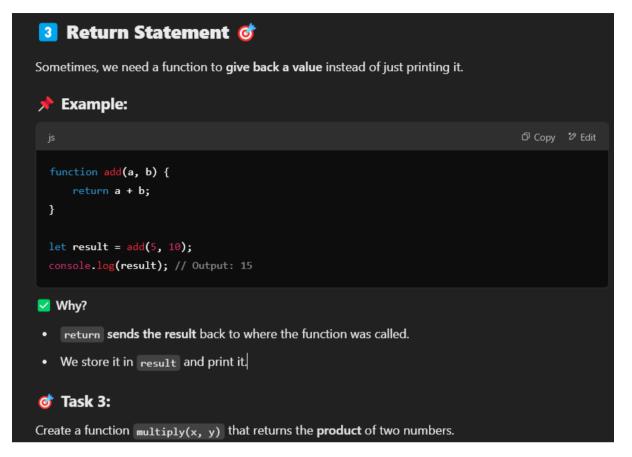
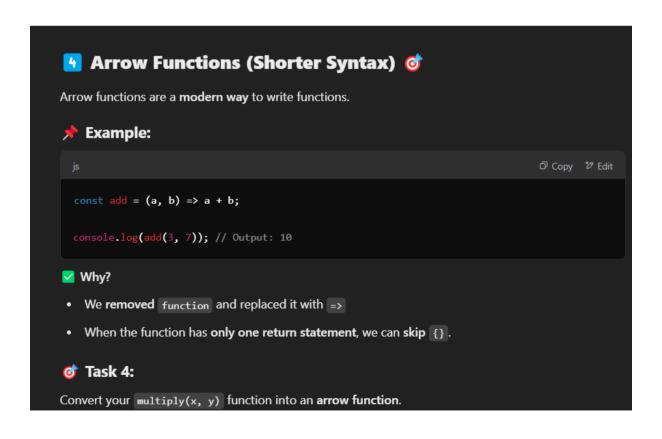


Basic function





We don't need to call the function if we use return



A callback function is a function passed inside another function.

Higher-Order Functions &

A higher-order function is a function that takes another function as an argument or returns a function.

📌 Exam<u>ple:</u>

```
function operation(action, a, b) {
   return action(a, b);
}

console.log(operation((x, y) => x + y, 4, 5)); // Output: 9
console.log(operation((x, y) => x * y, 4, 5)); // Output: 20
```

✓ Why?

- operation() takes a function as an argument (action).
- We pass x + y or x * y to decide the operation.

o Task 6:

Create a higher-order function applyFunction(func, a, b) that applies a given function to a and b.

5 IIFE (Immediately Invoked Function Expressions)

? A function that runs immediately after being defined.

Why use IIFE?

- Avoids polluting global scope.
- Useful for one-time execution of code.

Example

```
(function() {
    console.log("This function runs immediately!");
})();
// Output: This function runs immediately!
```

Since the function **executes instantly**, it's useful for things like **module patterns** and **initialization scripts**.

Closures

• A function that "remembers" variables from its outer scope, even after the outer function has executed.

Why use closures?

- To preserve data across function calls.
- Used in private variables, counters, and module patterns.

Example: Counter

```
function createCounter() {
    let count = 0; // Private variable

    return function() {
        count++;
        console.log(count);
    };
}

const counter = createCounter();
counter(); // Output: 1
counter(); // Output: 3
Here, the count variable is "remembered" by the returned function even after createCounter() is
```