Function

a function is a reusable block of code designed to perform a specific task.

Function Parameters:

Parameters are additional information passed to a function.

function multiply(a, b) {

b = typeof b !== "undefined" ? b : 1;

return a \* b;

}

console.log(multiply(69)); // 69

Calling Functions:

functionName( Value1, Value2, ..);

Return Statement:

There are some situations when we want to return some values from a function after performing some operations.

This is an optional statement and most of the time the last statement in a JavaScript function.

return value;

1. Function Declarations

A function declaration is the most basic form of a function. Function declarations are hoisted, meaning they are available in their entire scope, regardless of where they are defined.

function sayHello() {

console.log("Hello!");

}

sayHello(); // Outputs: Hello!

2. Function Expressions

Function expressions are not hoisted, and they allow you to create anonymous functions (functions without names) which can be useful for callbacks and immediate execution.

const greet = function(name) {

console.log("Hello " + name + "!");

};

greet("Alice"); // Outputs: Hello Alice!

3. Arrow Functions

Introduced in ES6, arrow functions offer a concise syntax and are great for short function expressions. They do not have their own this, arguments, super, or new.target bindings.

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

console.log(add(5, 3)); // Outputs: 8

const numbers = [1, 2, 3];

const squares = numbers.map(number => number \* number);

console.log(squares); // Outputs: [1, 4, 9]

6. IIFE (Immediately Invoked Function Expressions)

IIFEs are used to execute functions immediately as they are defined. This is useful for creating private scopes.

(function() {

const temp = "I am private";

console.log(temp);

})(); // Outputs: I am private