JAVASCRIPT FUNCTIONS

A function in JavaScript (and in programming in general) is a block of reusable code designed to perform a specific task. Functions help organize code, promote reusability, and make programs more modular and maintainable.

In JavaScript, functions can take input values (called parameters) and return an output (a return value). Functions are essential building blocks in JavaScript and can be used for various tasks, from simple calculations to handling events and performing asynchronous operations.

# Syntax of a Function

function functionName(parameters) {

return value;

}

## **Function Types in JavaScript**

1. **Named Functions**
2. **Anonymous Functions**
3. **Arrow Functions**
4. **Immediately Invoked Function Expressions (IIFE)**
5. **Higher-Order Functions**
6. **Async Functions**
7. **Constructor Function**
8. **Default Parameter Function**

### **1. Named Functions**

Functions declared with a specific name. These can be called by referencing the function name.

function greet(name) {

return `Hi, ${name}`;

}

console.log(greet("Venkat"));

### **2. Anonymous Functions**

Functions without a name, typically used as callbacks or assigned to variables.

const add = function (a, b) {

return a + b;

};

console.log(add(20, 30));

### **3. Arrow Functions**

A concise way to write functions using the => syntax. They have a lexically bound this.

const multiply = (a, b) => a \* b;

console.log(multiply(4, 5));

### **4. Immediately Invoked Function Expressions (IIFE)**

Functions that are defined and executed immediately.

(function () {

console.log("This runs immediately");

})();

Or using an arrow function:

(() => console.log("IIFE using arrow function"))();

### **5. Higher-Order Functions**

Functions that take other functions as arguments or return functions.

function operate(operation, a, b) {

return operation(a, b);

}

### **6. Async Functions**

Functions that return a Promise and use async/await for asynchronous operations.

async function fetchData() {

return await fetch("https://jsonplaceholder.typicode.com/posts/1").then(res => res.json());

}

### **8. Constructor Functions**

Functions used to create objects with the new keyword.

function Person(name, age) {

this.name = name;

this.age = age;

}

const person1 = new Person("Bob", 25);

console.log(person1.name); // Bob

### **9. Default Parameters in Functions**

Functions with default values for parameters.

function greet(name = "Guest") {

return `Hello, ${name}`;

}

console.log(greet());