**Documentation On JS Functions**

A **function** in JavaScript is a block of code designed to perform a particular task. A function can take input (called parameters), process it, and return a result. Functions help in reusing code, improving readability, and managing tasks in an organized way.

The basic syntax for a function in JavaScript is as follows:

**function** functionName(parameter1, parameter2) {

// code to execute

**return** result;

}

Example:

For example:

**function** add(a, b) {

return a + b;

}

**console.log**(add(2, 3)); // Output: 5

### ****Types of Functions in JavaScript****

JavaScript has several types of functions, depending on their syntax, how they are invoked, and their scope.

#### 1. ****Regular Functions****

These are functions declared using the function keyword.

Syntax:

**function** greet() {

console.log("Hello, world!");

}

**greet();**

#### 2. ****Function Expressions****

These are functions assigned to a variable. A function expression can be anonymous (without a name) or named.

**Anonymous Function Expression:**

const sum = function(a, b) {

return a + b;

};console.log(sum(2, 3)); // Output: 5

**Named Function Expression:**

const multiply = function multiply(a, b) {

return a \* b;

};console.log(multiply(3, 4)); // Output: 12

#### 3. ****Arrow Functions****

Arrow functions provide a more concise way to write functions, often used for short or anonymous functions.

Syntax:

const add = (a, b) => a + b;console.log(add(2, 3)); // Output: 5

**Arrow Function Characteristics:**

* Shorter syntax.
* Do not have their own this context (inherit this from the surrounding code).

#### 4. ****Anonymous Functions****

Functions that do not have a name. These are often used in function expressions, callbacks, or event handlers.

Example:

setTimeout(function() {

console.log("This is an anonymous function.");

}, 1000);

#### 5. ****Immediately Invoked Function Expression (IIFE)****

An IIFE is a function that is defined and executed immediately.

(function() {

console.log("I am an IIFE!");

})();

#### 6. ****Generator Functions****

Generator functions use the function\* syntax and allow for pausing and resuming function execution using the yield keyword.

Example:

function\* myGenerator() {

yield 1;

yield 2;

yield 3;

}

const gen = myGenerator();console.log(gen.next().value); // Output: 1console.log(gen.next().value); // Output: 2console.log(gen.next().value); Output: 3

#### 7. ****Recursive Functions****

A recursive function is a function that calls itself to solve a problem.

Example:

function factorial(n) {

if (n <= 1) return 1;

return n \* factorial(n - 1);

}

console.log(factorial(5)); // Output: 120