Functions

Question 1: What are functions in JavaScript? Explain the syntax for declaring and calling a function.

Answer 1: Functions are **reusable blocks of code** designed to perform a specific task. They help make code modular, organized, and easy to maintain.

**Declaring a Function:**

* To declare a function, use the **function** keyword, followed by a name, parentheses, and a code block.

**Syntax:**

function functionName(parameters) {

// Code to run

}

**Example:**

function great(name) {

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

}

**Calling a Function**

* To call (or execute) a function, use its name followed by parentheses. Pass any required arguments inside the parentheses.

**Syntax:**

functionName(arguments);

**Example:**

great("Alice"); // Output: Hello, Alice!

**Explanation:**

1. **Function Declaration:** function greet(name) {...} creates the function.
2. **Calling:** greet("Alice") runs the function, passing "Alice" as the argument for name.
3. **Result:** The message "Hello, Alice!" is printed.

Question 2: What is the difference between a function declaration and a function expression?

Answer 2: Difference Between Function Declaration and Function Expression

| **Aspect** | **Function Declaration** | **Function Expression** |
| --- | --- | --- |
| **Definition** | A named function defined using the function keyword. | A function assigned to a variable, can be anonymous. |
| **Syntax** | function functionName() { ... } | const functionName = function() { ... }; |
| **Hoisting** | Can be called **before its declaration** due to hoisting. | Cannot be called before its definition. |
| **Name** | Always has a name. | Can be named or anonymous. |

**Examples:**

**Function Declaration:**

greet(); // Works due to hoisting

function greet() {

console.log("Hello!");

}

**Function Expression:**

greet(); // Error: greet is not defined yet

const greet = function() {

console.log("Hello!");

};

**Explanation:**

Use **function declarations** for globally available functions and **function expressions** for more flexible or conditional function assignments.

Question 3: Discuss the concept of parameters and return values in functions.

Answer 3: Parameters and Return Values in Functions

1. **Parameters:**

* Parameters are **placeholders** for values you pass into a function.
* They allow the function to work with different inputs.
* You define parameters in the parentheses when declaring a function.

**Example:**

function add(a, b) { // 'a' and 'b' are parameters

return a + b;

}

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

1. **Return Values:**

* A **return value** is the result a function sends back after it finishes running.
* You use the return keyword to specify the value to be returned.
* If there’s no return, the function returns undefined by default.

**Example:**

function multiply(x, y) {

return x \* y; // Returns the product of x and y

}

let result = multiply(4, 6); // result = 24

console.log(result); // Output: 24

**Explanation:**

* **Parameters**: Input values for the function (e.g., a, b in add(a, b)).
* **Return Values**: Output of the function sent back to the caller using return.