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

**Functions in JavaScript**

A **function** in JavaScript is a reusable block of code designed to perform a specific task. Functions help in making the code modular, reusable, and easier to maintain.

**Syntax for Declaring a Function**

In JavaScript, functions can be declared using different methods, but the most common is the function keyword.

1. **Function Declaration**

function greet() {

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

}

* **function**: Keyword to define a function.
* **greet**: Function name (identifier).
* **() { }**: Parentheses for parameters (empty in this case) and curly braces for the function body.

1. **Function Expression**

const greet = function() {

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

};

Here, the function is assigned to a variable greet.

1. **Arrow Function (ES6)**

const greet = () => {

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

};

Shorter syntax, especially useful for anonymous functions.

**Calling a Function**

To execute a function, we use its name followed by parentheses:

greet(); // Output: Hello, World!

**Function with Parameters**

Functions can accept inputs (parameters):

function greetUser(name) {

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

}

greetUser("Dharti"); // Output: Hello, Dharti!

**Function with Return Value**

Functions can return a value using the return statement:

function add(a, b) {

return a + b;

}

let sum = add(5, 3);

console.log(sum);

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

Both function declarations and function expressions define functions in JavaScript, but they have key differences in syntax, behaviour, and hoisting.

**1. Function Declaration**

A function declaration defines a function using the function keyword with a name. It is hoisted, meaning it can be used before its definition.

**Example:**

console.log(greet());

function greet() {

return "Hello, World!";

}

console.log(greet()); // Output: "Hello, World!"

**Hoisting:** The function is available throughout the scope, even before the declaration.

**2. Function Expression**

A function expression defines a function and assigns it to a variable. It is **not hoisted**, so it cannot be called before its definition.

**Example:**

console.log(greet()); // ❌ Error: Cannot access 'greet' before initialization

const greet = function() {

return "Hello, World!";

};

console.log(greet()); // ✅ Output: "Hello, World!"

🚫 **No Hoisting:** The function cannot be called before its definition because it is stored in a variable.

**Key Differences**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Function Declaration** | **Function Expression** |
| **Syntax** | function funcName() {} | const funcName = function() {}; |
| **Hoisting** | ✅ Hoisted (can be used before declaration) | ❌ Not hoisted (must be defined before use) |
| **Usage Before Definition** | ✅ Allowed | ❌ Not allowed (Reference Error) |
| **Naming** | Must have a function name | Can be anonymous or named |
| **Best Use Case** | When defining reusable functions globally | When assigning functions to variables, passing as arguments |

**Example Use Cases**

✅ **Use Function Declarations** when defining general functions used throughout the script.  
✅ **Use Function Expressions** when assigning functions to variables, callbacks, or event handlers.

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

Functions in JavaScript can take parameters (inputs) and return values (outputs). This makes functions more dynamic and reusable.

**1. Function Parameters (Inputs)**

* Parameters allow functions to accept values when they are called.
* These values are used inside the function body.
* If no argument is provided, the parameter is undefined unless a default value is assigned.

**Example of Function with Parameters:**

function greet(name) {

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

}

greet("Rahul"); // Output: Hello, Rahul!

greet("Raj"); // Output: Hello, Raj!

Function with Multiple Parameters:

function add(a, b) {

console.log(a + b);

}

add(5, 3); // Output: 8

**2. Default Parameters**

If no argument is passed, we can set default values to avoid undefined.

function greet(name = "Guest") {

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

}

greet(); // Output: Hello, Guest!

**3. Return Values (Outputs)**

* The return statement sends a value back to where the function was called.
* If no return is specified, the function returns undefined.

**Example of Function with Return Value:**

function multiply(a, b) {

return a \* b; // Returns the product of a and b

}

let result = multiply(4, 5);

console.log(result);

**Function Without Return (Implicit undefined):**

function sayHello() {

console.log("Hello!");

}

let output = sayHello(); // Output: "Hello!"

console.log(output); // Output: undefined

**4. Returning Multiple Values**

JavaScript functions can return multiple values using **arrays** or **objects**.

**Returning an Array:**

function getCoordinates() {

return [40.7128, -74.0060]; // Latitude, Longitude

}

let coords = getCoordinates();

console.log(coords[0]);

console.log(coords[1]);

**Returning an Object:**

function getUserInfo() {

return { name: "Modi", age: 25 };

}

let user = getUserInfo();

console.log(user.name);

console.log(user.age);