# Functions

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

**Ans.**

JavaScript, a function is a block of organized, reusable code designed to perform a specific task. Functions allow for code modularity, reusability, and maintainability

Syntax for Declaring a Function (Function Declaration):

The most common way to declare a function in JavaScript is using the function keyword.

```
function functionName(parameter1, parameter2, ...) {
// Code to be executed
// Optional: return value;
}
```

- function keyword: This keyword signifies the declaration of a function.
- functionName: This is the name given to the function. It should follow standard JavaScript variable naming conventions.
- parameter1, parameter2, ... (optional): These are variables that act as placeholders for values that will be passed into the function when it is called. They are enclosed in parentheses and separated by commas.
- { ... }: The curly braces enclose the function body, which contains the JavaScript statements that will be executed when the function is called.
- return value; (optional): The return statement can be used to send a value back to the part of the code that called the function. If no return

statement is present or if return; is used without a value, the function implicitly returns undefined.

Example of Function Declaration:

```
function greet(name) {
console.log("Hello, " + name + "!");
}
```

Syntax for Calling a Function (Function Invocation):

To execute the code within a function, you need to call or invoke it.

```
functionName(argument1, argument2, ...);
```

- functionName: This is the name of the function you want to execute.
- argument1, argument2, ... (optional): These are the actual values passed to the function, which correspond to the parameters defined in the function declaration. They are enclosed in parentheses and separated by commas.

Example of Calling a Function:

```
greet("Alice"); // This will log "Hello, Alice!" to the console
 let myName = "Bob";
 greet(myName); // This will log "Hello, Bob!" to the console
```

---

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

**Ans.**

| function declaration | function expression |
|---|---|
| A function declaration must have a function name. | A function expression is similar to a function declaration without the function name. |
| Function declaration does not require a variable assignment. | Function expressions can be stored in a variable assignment. |
| These are executed before any other code. | Function expressions load and execute only when the program interpreter reaches the line of code. |
| The function in function declaration can be accessed before and after the function definition. | The function in function expression can be accessed only after the function definition. |
| Function declarations are hoisted | Function expressions are not hoisted |
| Syntax: function geeksforGeeks(paramA, paramB) { // Set of statements } | Syntax: var geeksforGeeks= function(paramA, paramB) { // Set of statements } |

---

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

**Ans.**

Functions in programming often involve mechanisms for receiving input and providing output, which are handled by parameters and return values, respectively.

- Parameters :

  Parameters are variables defined in a function's declaration or definition that serve as placeholders for values that will be passed into the function when it is called. These values, known as arguments, provide the necessary data for the function to perform its operations.

- ✓ Purpose :
  Parameters allow functions to be flexible and reusable. Instead of hardcoding specific values, a function can accept different inputs each time it's invoked, enabling it to operate on varying data without requiring modifications to its internal logic.

- ✓ Declaration :
  Parameters are typically specified within parentheses in the function's signature, often including a data type (in statically typed languages) and a name.

- ✓ Passing Arguments:
  When a function is called, arguments are provided in the same order as the parameters are declared, and their values are copied into the corresponding parameters within the function's scope.

- Return Values:
  A return value is a piece of data that a function sends back to the part of the code that called it after the function has completed its execution.

  - ✓ Purpose :
    Return values allow functions to produce results or convey information back to the calling context. This enables functions to perform calculations, retrieve data, or indicate the success or failure of an operation.

- ✓ Mechanism :

  The return statement is used within a function to specify the value that will be sent back. Once a return statement is executed, the function terminates, and the specified value is passed back to the caller.

- ✓ Types:

  Functions can return various data types, including numbers, strings, objects, or even void (or None in some languages) if no explicit value is returned. In such cases, the function is primarily executed for its side effects rather than for a specific output.

- Relationship:

  Parameters facilitate the input of data into a function, while return values facilitate the output of data from a function. Together, they create a clear interface for interaction with a function, promoting modularity and reusability in code.

---