1.2 Javascript Functions

A function is a block of code that performs a specific task. Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

▼ Declaring a Function

The syntax to declare a function is:

```
function nameOfFunction () {
    // function body
}
```

A function is declared using the `function` keyword.

▼ Calling a Function

```
function nameOfFunction () {
    // function body
}

nameOfFunction(); //calling the function
```

▼ Function Parameters

```
// program to print the text
// declaring a function
function greet(name) {
    console.log("Hello " + name + ":)");
}

function add(a, b) {
    console.log(a + b);
}
```

▼ Function Return

The `return` statement can be used to return the value to a function call.

The `return` statement denotes that the function has ended. Any code after `return` is not executed.

If nothing is returned, the function returns an `undefined` value.

▼ Arrow Functions

The arrow function is one of the features introduced in the ES6 version of JavaScript. It allows you to create functions in a cleaner way compared to regular functions. For example,This function

```
// function expression
let multiply = function(x, y) {
   return x * y;
}
```

can be written as

```
// using arrow functions
let multiply = (x, y) => x * y;
```

using an arrow function.

```
let sum = (a, b) => {
    let result = a + b;
    return result;
}

let result1 = sum(5,7);
```

▼ Default Parameter Values

In the ES6 version, you can pass default values in the function parameters. For example,

```
function sum(x, y = 5) {

    // take sum
    // the value of y is 5 if not passed
    console.log(x + y);
```

```
}

sum(5); // 10
sum(5, 15); // 20
```

In the above example, if you don't pass the parameter for `y`, it will take **5** by default.

▼ First-class citizens

Functions are treated as first-class citizens, meaning they can:

- Be assigned to variables.
- Be passed as arguments.
- Be returned from other functions.

Functions that take other functions as arguments or return functions as results are called Higher-order functions

```
const greet = name => `Hello, ${name}`;
const logGreeting = (greetingFunction, name) => {
    console.log(greetingFunction(name));
};
logGreeting(greet, "Alice"); // Hello, Alice
```

```
const applyOperation = (a, b, operation) => operation(a, b

const add = (x, y) => x + y;
const multiply = (x, y) => x * y;

console.log(applyOperation(3, 5, add));      // 8
console.log(applyOperation(3, 5, multiply)); // 15
```

Aspect	Java	JavaScript
Declaration	Functions are part of a class (methods).	Functions can exist independently or as part of objects.
Syntax	Defined using `returnType methodName(parameters) {}` within a class.	Defined using `function` keyword or as **arrow functions** (`=>`).
Overloading	Allows method overloading (same name, different parameters).	Does not support function overloading natively (last definition overwrites the previous ones).
First-Class Citizen	Functions are not first-class citizens; you cannot pass methods directly as parameters.	Functions are **first-class citizens**: they can be passed as arguments, returned, or assigned.
This Context	`this` refers to the current instance of the class.	`this` depends on the calling context (can vary inside objects, arrow functions, etc.).

Assignments

1. Create a function that takes multiple functions as argument to build a calculator
2. Create a method to take the average of three numbers, and then convert it to an arrow function.