Table of Contents

## 0.1 FunctionDef add(a, b)

# 1 Function: add(a, b)

## 1.1 Overview

The add function computes the sum of two given numbers.

## 1.2 parameters

| Parameter | Type | Description |
| --- | --- | --- |
| a | Number | The first number to be added. |
| b | Number | The second number to be added. |

## 1.3 Description

This function takes two parameters, a and b, and returns their sum. It uses the standard addition operator (+) to perform the calculation. The core logic is contained in a single statement return a + b;, which evaluates the sum of the two inputs and immediately returns the result.

// The function returns the result of a + b  
return a + b;

## 1.4 Usage Notes

* For standard arithmetic addition, ensure both a and b are of the Number type.
* If either a or b is a string, the + operator will perform string concatenation instead of numerical addition. For example, add(5, "3") would result in the string "53".

**Output Example**: A number representing the sum of the inputs.

## 1.5 Example

// Example 1: Adding two numbers  
const result = add(15, 7);  
console.log(result);

**Output:**

22

## 1.6 FunctionDef factorial(n)

# 2 Function: factorial(n)

## 2.1 Overview

The factorial function recursively calculates the factorial of a given non-negative integer.

## 2.2 Parameters

* **n** (Number): The non-negative integer for which the factorial will be calculated.

## 2.3 Description

The factorial function is a recursive implementation designed to compute the factorial of a number n, which is the product of all positive integers up to n.

The function operates based on two main cases:

1. **Base Case**: The recursion terminates when the input n is less than or equal to 1. In this scenario, the function returns 1. This is because the factorial of 1 (1!) is 1, and the factorial of 0 (0!) is also defined as 1.
2. **Recursive Step**: If n is greater than 1, the function multiplies n by the result of calling itself with the argument n - 1. This process continues, decrementing n by 1 in each subsequent call, until the base case is reached.

For example, calling factorial(4) unfolds as follows: 4 \* factorial(3) 4 \* (3 \* factorial(2)) 4 \* (3 \* (2 \* factorial(1))) 4 \* (3 \* (2 \* 1)) = 24

## 2.4 Usage Notes

* This function is recursive. Providing a very large number for n can lead to a “Maximum call stack size exceeded” error (stack overflow).
* The function is intended for non-negative integers. Passing a negative number will result in infinite recursion and a stack overflow.
* While it may accept floating-point numbers, the mathematical concept of a factorial is typically defined only for non-negative integers, and results for non-integers may not be meaningful.

**Output Example**: The function returns a single number representing the calculated factorial.

## 2.5 Example

// Example usage  
const result = factorial(5);  
console.log(result);

**Output:**

120

## 2.6 FunctionDef greet(name)

# 3 Function: greet

## 3.1 Overview

The greet function prints a personalized greeting message to the console.

## 3.2 parameters

* name (String): The name to be included in the greeting message.

## 3.3 Description

This function provides a simple way to display a standardized greeting. It accepts a single argument, name, which is expected to be a string.

The core logic uses the console.log method to output a message to the standard output, such as a web browser’s developer console or a Node.js terminal. The message is constructed using a template literal: `Hello, ${name}!`. The ${name} placeholder is dynamically replaced with the value passed to the name parameter when the function is invoked.

## 3.4 Usage Notes

* This function does not return a value (it implicitly returns undefined). Its sole purpose is to produce a side effect by logging output to the console.
* While the name parameter is intended to be a string, JavaScript’s type coercion will attempt to convert any passed value into its string representation for the output. For example, passing the number 123 will result in the output "Hello, 123!".

## 3.5 Example

// Example usage  
greet("World");  
greet("Alice");

**Output:**

Hello, World!  
Hello, Alice!