Table of Contents

## 0.1 FunctionDef add(a, b)

# 1 Function: add(a, b)

## 1.1 Overview

The add function calculates the sum of two given arguments.

## 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 fundamental JavaScript addition operator (+) to perform the calculation. The expression a + b is evaluated, and the resulting value is returned by the function.

It’s important to understand the behavior of the + operator in JavaScript. While this function is intended for numeric addition, if one or both of the arguments are strings, the operator will perform string concatenation instead.

// Numeric addition  
add(10, 20); // Returns 30  
  
// String concatenation due to type coercion  
add(10, "20"); // Returns "1020"

## 1.4 Usage Notes

* This function is designed for adding two numbers. For predictable results, ensure both a and b are of the Number type.
* The function does not include any type checking. If non-numeric arguments are provided, JavaScript’s type coercion rules will apply, which may lead to unexpected results like string concatenation instead of arithmetic addition.

**Output Example**: A numeric value representing the sum of the inputs, or a string if concatenation occurs.

## 1.5 Example

// Example usage with two numbers  
let 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 to calculate the factorial.

## 2.3 Description

This function computes the factorial of a number n using a recursive approach. The factorial of a non-negative integer n, denoted by n!, is the product of all positive integers less than or equal to n.

The function’s logic is based on a ternary operator which serves as a compact if-else statement:

1. **Base Case:** The condition n <= 1 is checked first. If n is 0 or 1, the function returns 1. This is the termination condition for the recursion, as the factorial of both 0 and 1 is defined as 1.
2. **Recursive Step:** If n is greater than 1, the function returns the product of n and the result of calling itself with the argument n - 1. This process repeats, decrementing n by 1 in each subsequent call, until the base case (n <= 1) is reached.

For example, factorial(4) would be calculated as follows:

factorial(4) = 4 \* factorial(3)  
 = 4 \* (3 \* factorial(2))  
 = 4 \* (3 \* (2 \* factorial(1)))  
 = 4 \* (3 \* (2 \* 1))  
 = 24

## 2.4 Usage Notes

* This function is intended for non-negative integers. Providing a negative number will result in infinite recursion and cause a stack overflow error.
* Due to the nature of recursion, calculating the factorial of very large numbers can lead to a “Maximum call stack size exceeded” error. For large-scale calculations, an iterative approach might be more memory-efficient.

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

24

## 2.5 Example

// Example usage  
const number = 5;  
const result = factorial(number);  
console.log(`The factorial of ${number} is ${result}`);

**Output:**

The factorial of 5 is 120

## 2.6 FunctionDef greet(name)

# 3 Function: greet(name)

## 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 standard greeting. It accepts a single argument, name, which is expected to be a string.

The core logic uses a template literal to construct the output string Hello, ${name}!. The ${name} placeholder is dynamically replaced with the value of the name parameter provided during the function call.

Finally, the fully constructed string is passed to the console.log() method, which outputs the message to the standard console. The function does not return any value.

// The function uses a template literal for string formatting  
`Hello, ${name}!`

## 3.4 Usage Notes

* This function does not return a value (undefined); its sole purpose is to log a message to the console.
* If a non-string value (e.g., a number or an object) is passed as the name parameter, JavaScript will attempt to convert it to its string representation for the output.

## 3.5 Example

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

**Output:**

Hello, World!  
Hello, Alice!