# Practical 1

Task-1

Aim: Declare a variable using var, let, and const. Assign different data types to each variable and print their values.

Theoretical Background:

Source Code:

// Task 1: Variables and Data Types

var Name = "John";

let Number = 10;

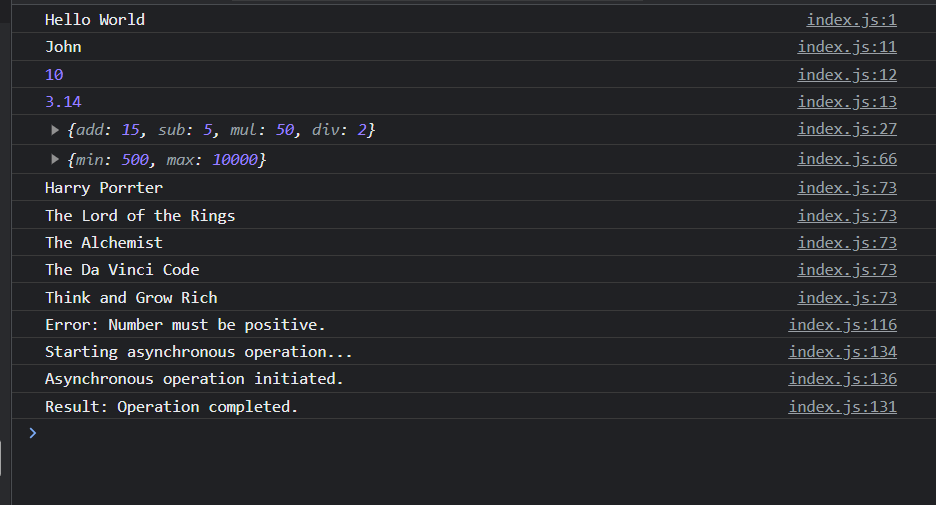
const PI = 3.14;

console.log(Name);

console.log(Number);

console.log(PI);

Output:



Task-2

Aim: Write a function that takes two numbers as arguments and returns their sum, difference, product, and quotient using arithmetic operators.

Theoretical Background:

Source Code:

let calc = function (num1, num2) {

    return {

        add: num1 + num2,

        sub: num1 - num2,

        mul: num1 \* num2,

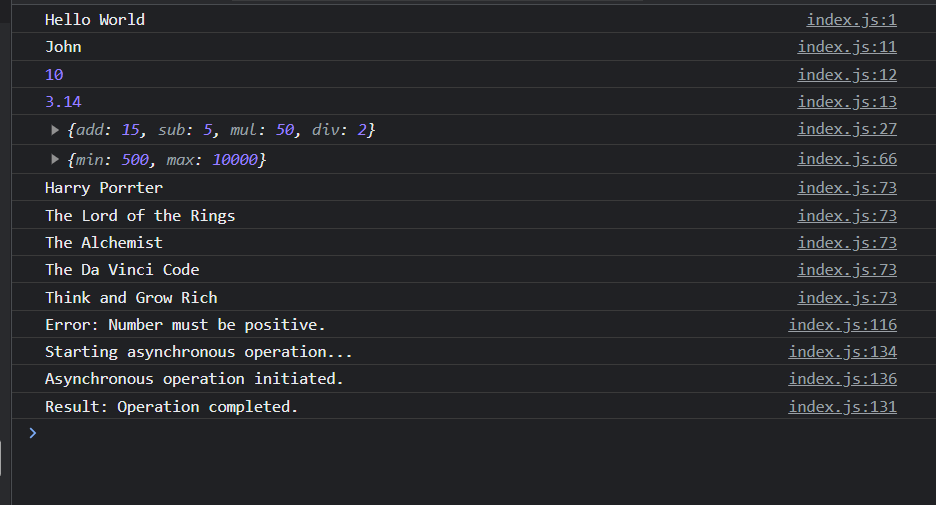
        div: num1 / num2,

    }

}

console.log(calc(10, 5));

Output:



Task-3

Aim: Write a program that prompts the user to enter their age. Based on their age, display different messages:

○ If the age is less than 18, display "You are a minor."

○ If the age is between 18 and 65, display "You are an adult."

○ If the age is 65 or older, display "You are a senior citizen."

Theoretical Background:

Source Code:

let getage = function () {

    var age = document.getElementById("id\_Age").value;

    // age = parseInt(age);

    if (age < 18) {

        alert("You are a minor");

    } else if (age >= 18 && age < 65) {

        alert("You are an adult");

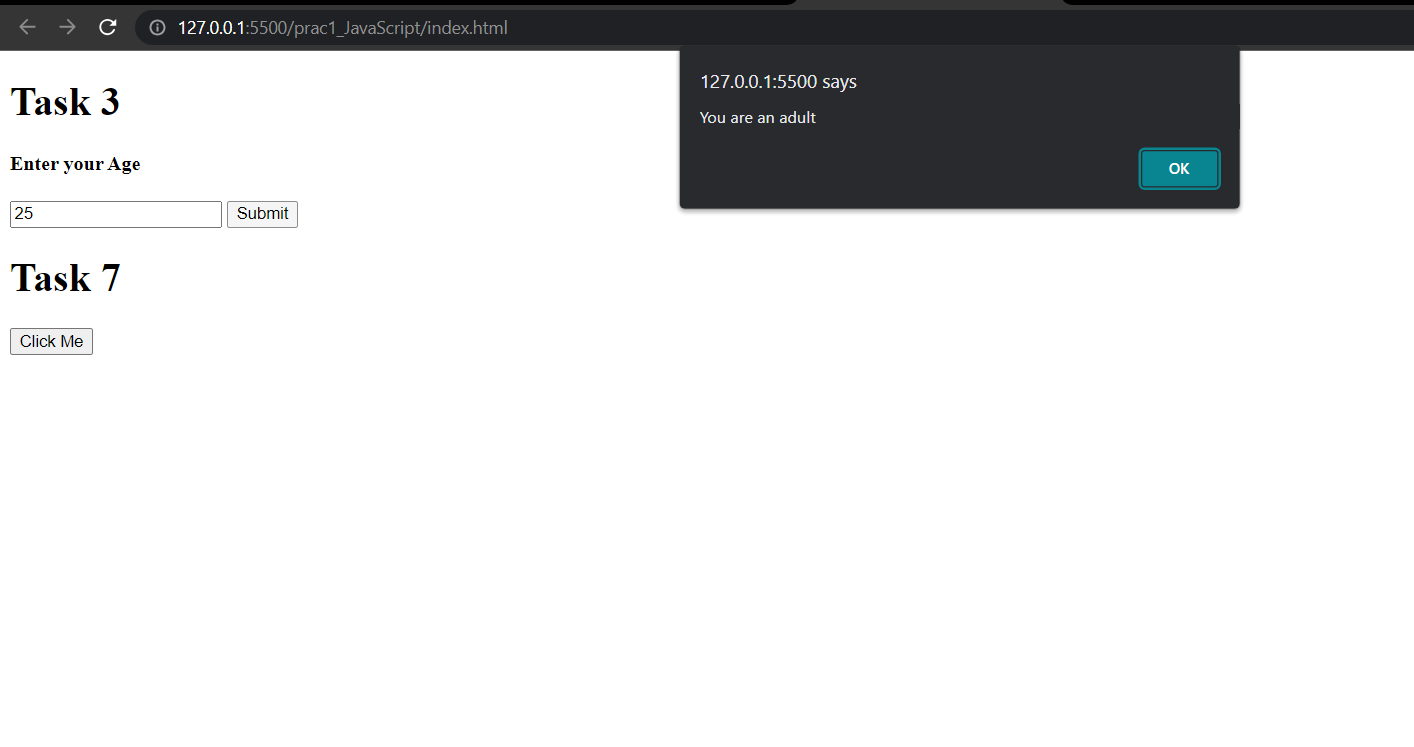
    } else {

        alert("You are a senior citizen");

    }

}

Output:



Task-4

Aim: Write a function that takes an array of salary as an argument and returns the min/max salary in the array.

Theoretical Background:

Source Code:

// Task 4: Functions

let getMinMaxSalary = function (salaries) {

    let min = salaries[0];

    let max = salaries[0];

    for (let i = 0; i < salaries.length; i++) {

        if (salaries[i] < min) {

            min = salaries[i];

        }

        if (salaries[i] > max) {

            max = salaries[i];

        }

    }

    return {

        min: min,

        max: max

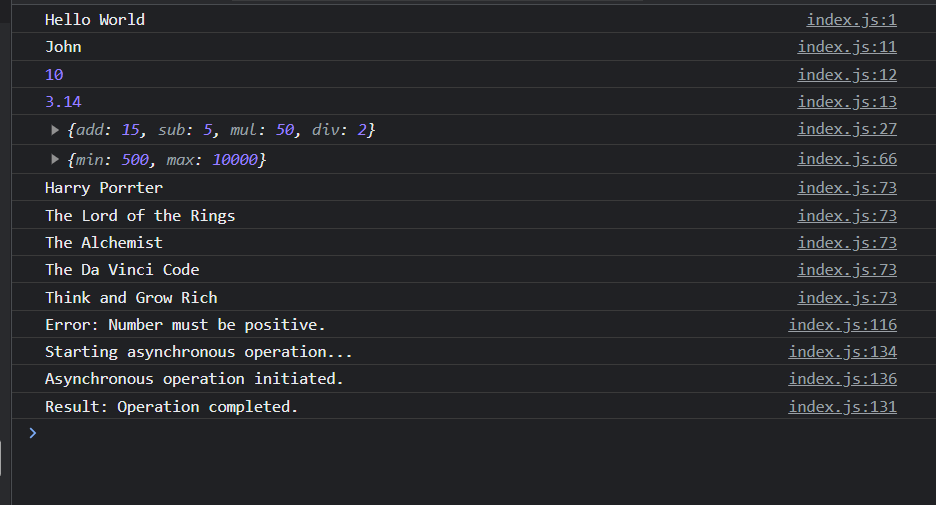
    };

}

let salaries = [1000, 500, 2000, 3000, 4000, 6000, 7000, 8000, 9000, 10000];

console.log(getMinMaxSalary(salaries));

Output:



Task-5

Aim: Create an array of your favorite books. Write a function that takes the array as an argument and displays each book title on a separate line.

Theoretical Background:

Source Code:

// Task 5: Arrays and Objects

let displayFavBooks = function (books) {

    for (let i = 0; i < books.length; i++) {

        console.log(books[i]);

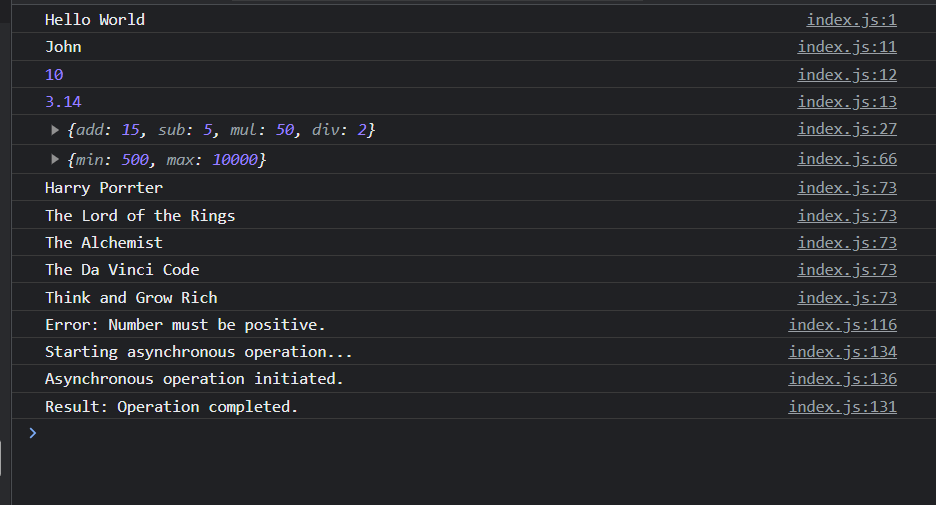
    }

}

let books = ["Harry Porrter", "The Lord of the Rings", "The Alchemist", "The Da Vinci Code", "Think and Grow Rich"];

displayFavBooks(books);

Output:



Task-6

Aim: Declare a variable inside a function and try to access it outside the function. Observe the scope behavior and explain the results. [var vs let vs const]

Theoretical Background:

Source Code:

// Task 6: Scope and Hoisting

let ScopeandHoisting = function () {

    let vara = 100;

    var varb = 200;

    const varc = 300;

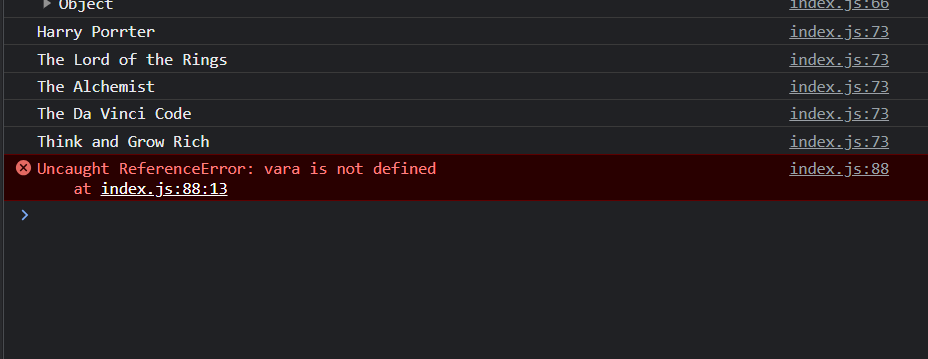
}

console.log(vara);

console.log(varb);

console.log(varc);

Output:



Task-7

Aim: Create an HTML page with a button. Write JavaScript code that adds an event listener to the button and changes its text when clicked.

Theoretical Background:

Source Code:

// Task 7: DOM Manipulation

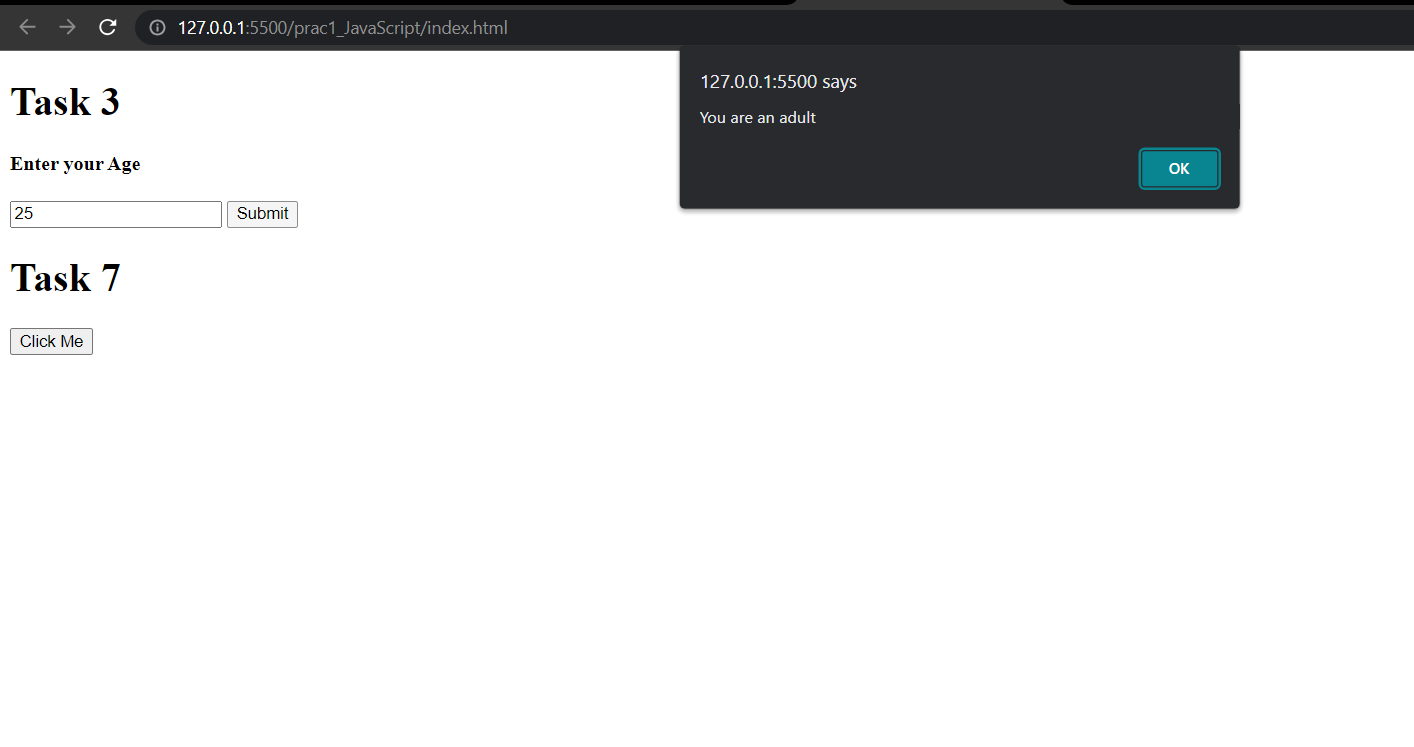
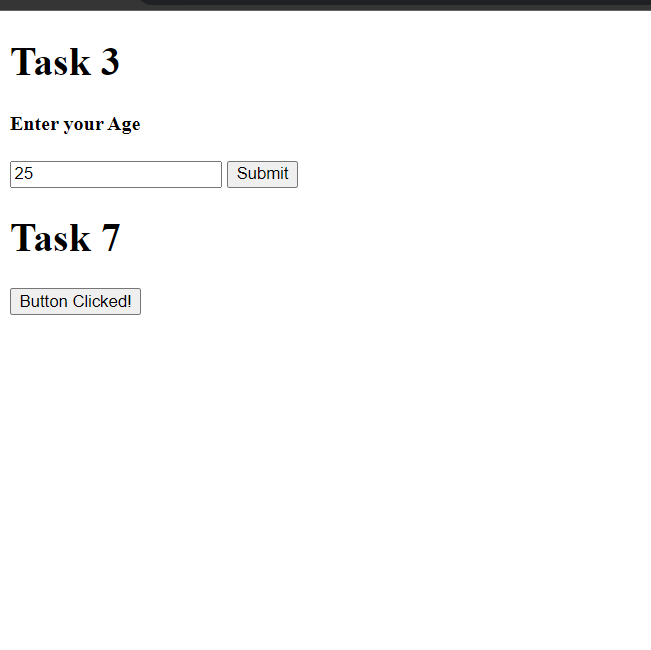
var button = document.getElementById("myButton");

button.addEventListener("click", function () {

    button.textContent = "Button Clicked!";

});

Output:

Task-8

Aim: Write a function that takes a number as an argument and throws an error if the number is negative. Handle the error and display a custom error message.

Theoretical Background:

Source Code:

// Task 8: Error Handling

function checkPositiveNumber(number) {

    if (number < 0) {

        throw new Error("Number must be positive.");

    }

    return number;

}

try {

    var result = checkPositiveNumber(-5);

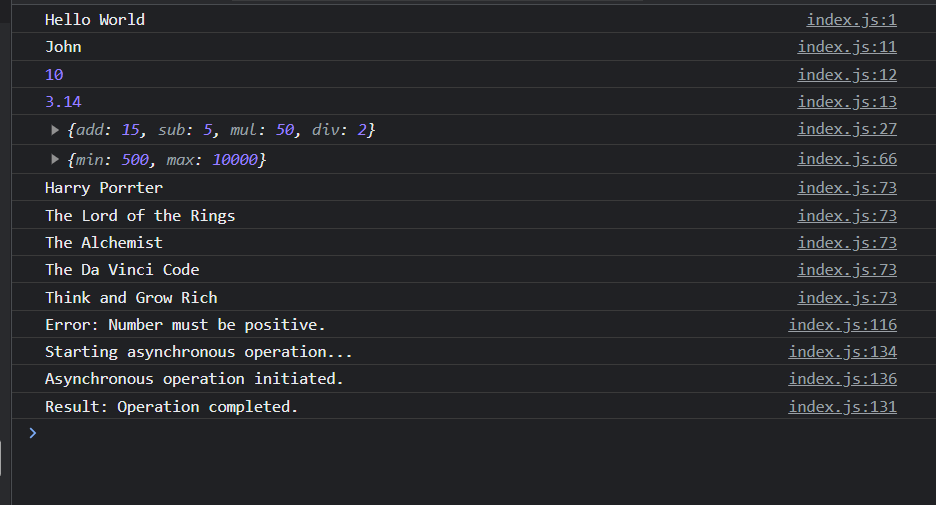
    console.log("Result:", result);

} catch (error) {

    console.log("Error:", error.message);

}

Output:



Task-9

Aim: Write a function that uses setTimeout to simulate an asynchronous operation. Use a callback function to handle the result.

Theoretical Background:

Source Code:

// Task 9: Asynchronous JavaScript

function asyncFunction(callback) {

    setTimeout(function () {

        var result = "Operation completed.";

        callback(result);

    }, 10000);

}

function callback(result) {

    console.log("Result:", result);

}

console.log("Starting asynchronous operation...");

asyncFunction(callback);

console.log("Asynchronous operation initiated.")

Output:

# 