Task-1

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

Description:

Variables:

A variable is a named location in a computer's memory that can hold a value. It acts as a container for storing and referencing data during program execution. Variables allow programmers to assign values, retrieve and modify them as needed. When a variable is created, it is typically assigned a data type, which determines the kind of values it can hold.

Data Types:

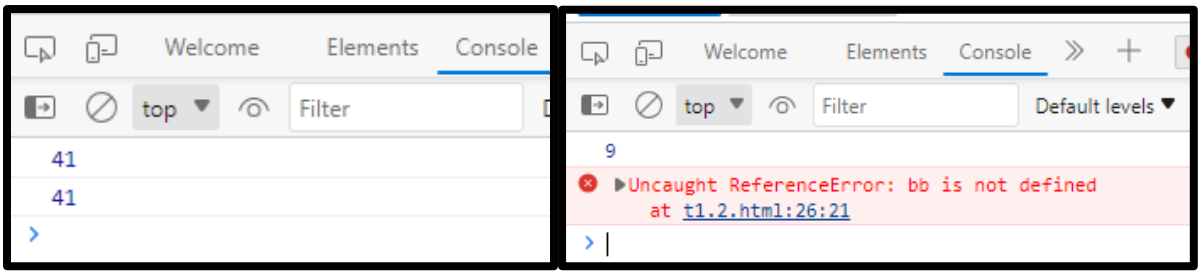
Data types define the nature of data that can be stored in a variable. They specify the range of values a variable can hold, the operations that can be performed on it, and the memory required to store it. Common data types include

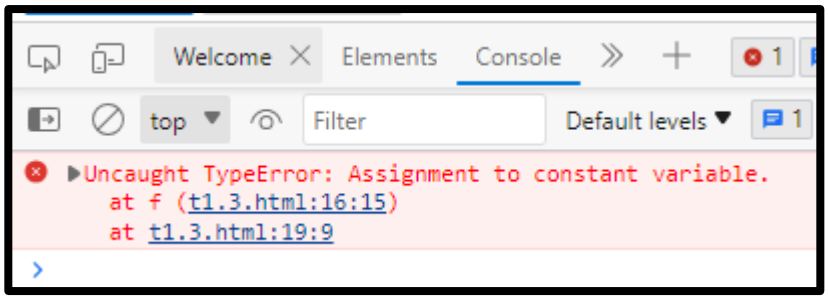
Source Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>
```

```
        //var keyword
        var a = 41
        function f() {
            console.log(a)
        }
        f();
        console.log(a)



       // let keyword

        let aa = 10;
        function f() {

        let bb = 9
        console.log(bb);

        }
        f()

        console.log(bb)

        //const keyword
        const a = 10;
        function f() {
            a = 9
            console.log(a)
        }
        f();



    </script>
</body>

</html>
```

Output:





Task-2

Aim:

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

Description:

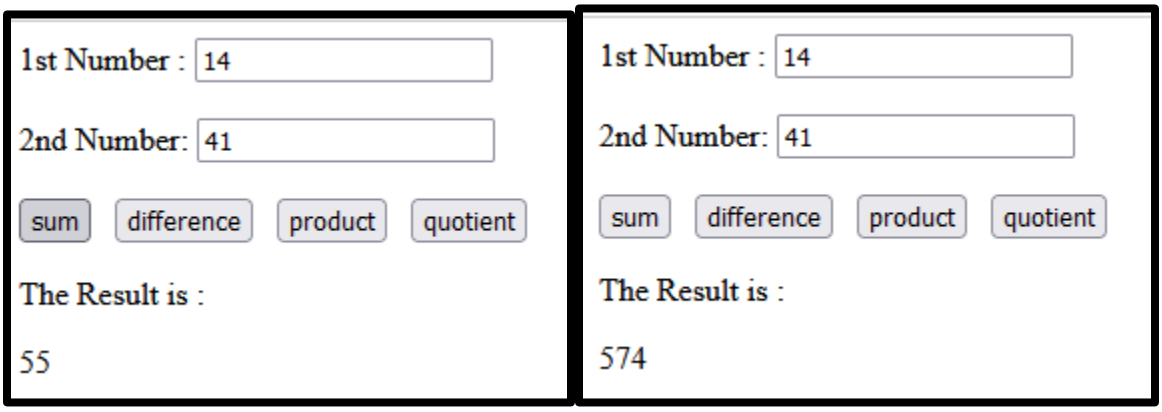
- Operators are symbols or keywords used to perform actions or computations on operands. They include arithmetic, assignment, comparison, logical, and bitwise operators. Expressions are combinations of operators, operands, and other expressions that result in a single value. They represent computations to be performed and can involve variables, constants, and function calls.

Source Code:

```
 <!DOCTYPE html>
<html>
<head>
<meta charset=utf-8 />
<title>JavaScript  oprasion two numbers </title>

</head>
<body>
    <script>
        function sum()
        {
                num1 = parseInt(document.getElementById("firstNumber").value);
                num2 = parseInt(document.getElementById("secondNumber").value);
                document.getElementById("ans").innerHTML = num1 + num2;
        }

        function difference()
        {
                num1 = document.getElementById("firstNumber").value;
                num2 = document.getElementById("secondNumber").value;
                document.getElementById("ans").innerHTML = num1 - num2;
        }

         function product()
        {
                num1 = document.getElementById("firstNumber").value;
                num2 = document.getElementById("secondNumber").value;
        document.getElementById("ans").innerHTML = num1 * num2;
        }
        function quotient()
        {
                num1 = document.getElementById("firstNumber").value;
                num2 = document.getElementById("secondNumber").value;
        document.getElementById("ans").innerHTML = num1 / num2;
        }
    </script>
<form>

1st Number : <input type="text" id="firstNumber" /><br><br>
2nd Number: <input type="text" id="secondNumber" /><br><br>
<input type="button" onClick="sum()" Value="sum" />  
<input type="button" onClick="difference()" Value="difference" />  
<input type="button" onClick="product()" Value="product" />  
<input type="button" onClick="quotient()" Value="quotient" />  
</form>
<p>The Result is : <div id = "ans"></div></p>
```

```
</body>
</html>
```

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."

Description:

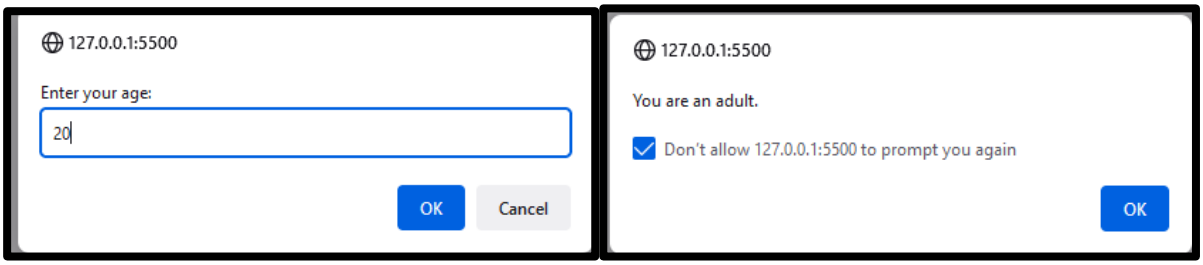
use html structure and get input from user with prompt.

Source Code:

```
<!DOCTYPE html>
<html>
<head>
```

```
  <title></title>
  <script>
    function check() {
      var ageInput = prompt("Enter your age:");
      var age = parseInt(ageInput);

      var message;

      if (age < 18) {
        message = "You are a minor.";
      } else if (age >= 18 && age <= 65) {
        message = "You are an adult.";
      } else {
        message = "You are a senior citizen.";
      }

      alert(message);
    }
  </script>
</head>

<body>
  <h1>Age Checker</h1>

  <button onclick="check()">Check</button>
</body>
</html>
```

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.

Description:

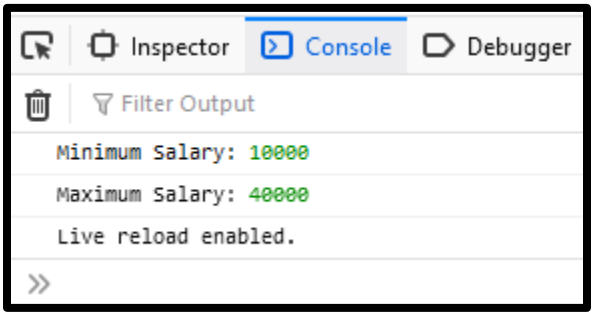
- CA function consists of the function keyword, followed by: The name of the function. ● A list of parameters to the function, enclosed in parentheses and separated by commas. ● The JavaScript statements that define the function, enclosed in curly brackets, {/* … */}.

Source Code:

```
 <!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>

        var salaryArray = [10000, 20000, 30000, 40000];
        var result = findMinMaxSalary(salaryArray);

        function findMinMaxSalary(salaries) {
            if (salaries.length === 0) {
                return null; // Return null if the array is empty
            }
            var minSalary = salaries[0];
            var maxSalary = salaries[0];

            for (var i = 1; i < salaries.length; i++) {
                if (salaries[i] < minSalary) {
                    minSalary = salaries[i];
                }

                if (salaries[i] > maxSalary) {
                    maxSalary = salaries[i];
                }
            }
            return {
                minSalary: minSalary,
                maxSalary: maxSalary
            };
```

```
        }
        console.log("Minimum Salary:", result.minSalary);
        console.log("Maximum Salary:", result.maxSalary);
    </script>
</body>

</html>
```

Ouput:



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.

Description:

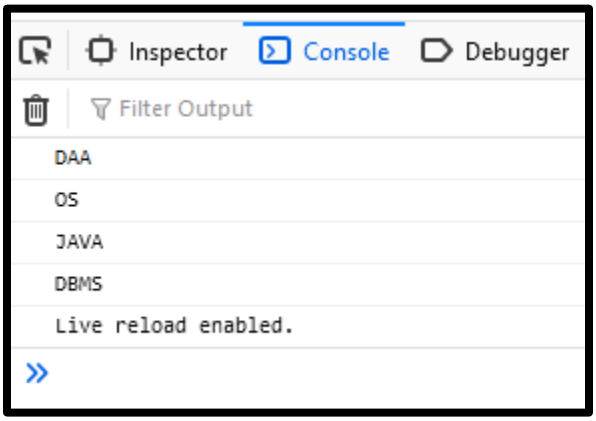
use array of books and display it with new line

Source Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
```

```
        function Books(books) {
            for (var book of books) {
                console.log(book);
            }
        }

        // Example usage:
        var favoriteBooks = ["DAA", "OS", "JAVA","DBMS"];
      Books(favoriteBooks);
    </script>
</body>

</html>
```

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]

Description:

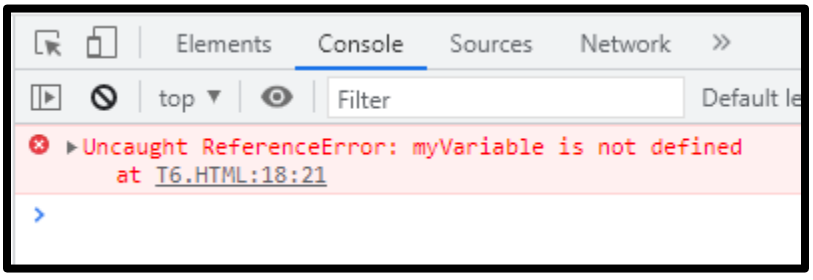
- Scope refers to where variables and functions are available in your code. It determines which
  parts of your program can access and use them. Variables and functions can have global scope, meaning they can be accessed from anywhere in your code, or they can have function
  or block scope, which limits their accessibility to specific parts of your code.

- Hoisting is a behaviour in JavaScript where variable and function declarations are moved to
  the top of their respective scopes during the code execution. This means that you can use variables and call functions before they are actually declared in your code, but their values

Source Code:

```
function scopeWithVar() {
            var myVariable = "Hey";
        }

        scopeWithVar();
        console.log(myVariable);
        function scopeWithLet() {
            let myVariable = "Hey";
        }

        scopeWithLet();
        console.log(myVariable);
        function scopeWithConst() {
            const myVariable = "Hey";
        }

        scopeWithConst();
        console.log(myVariable);
```

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.

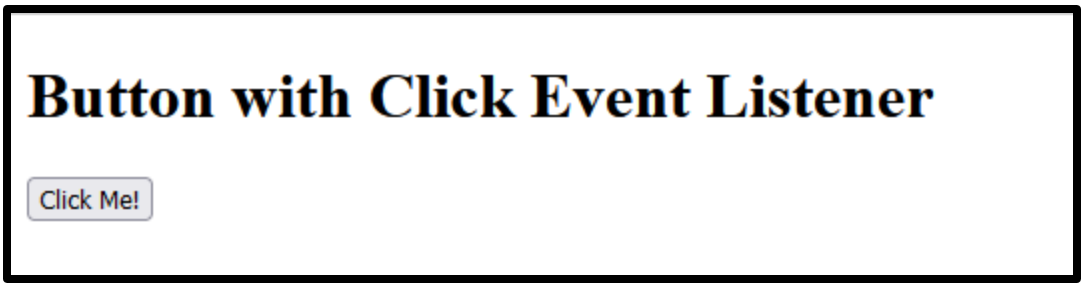
Description:

try to understand event listener.

Source Code:

```
<!DOCTYPE html>
<html>

<head>
    <title>Button with Click Event Listener</title>
</head>

<body>
    <h1>Button with Click Event Listener</h1>
    <button id="myButton">Click Me!</button>
    <script>
        const myButton = document.getElementById("Button");
        myButton.addEventListener("click", function () {
            myButton.innerHTML = "click";
        });
    </script>
</body>
</html>
```

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.

Description:

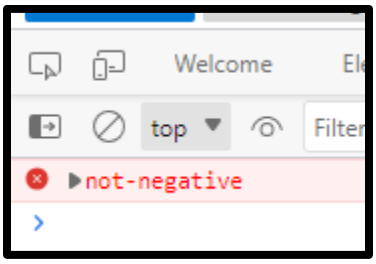
- Errors and Exceptions:
- Errors and exceptions are abnormal conditions or events that disrupt the normal flow of a program's execution. They can occur due to various reasons, such as invalidinputs, logical errors, external dependencies, or system failures.
- Syntax Errors: These occur during the parsing stage of the program when the code violates the language's syntax rules. Syntax errors typically prevent the program from executing at all. Runtime Errors: Also known as exceptions, these occur during the execution of the program when unexpected situations arise. Runtime errors can be caused by various factors, such as invalid operations, resource unavailability, or unexpected data.

Source Code:

```
 <!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>

<body>
    <script>
        function validateNumber(number) {
            "use strict";

            if (number < 0) {
                throw new Error("not-negative");
            }
        }

        try {
            validateNumber(-1);
        } catch (error) {
            console.error(error.message);
        }
```

```
    </script>
</body>
</html>
```

Output:



Task-9

Aim:

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

Description:

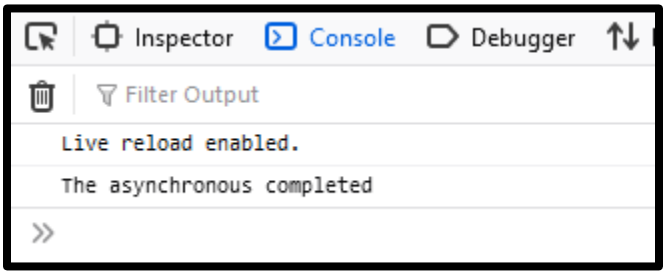
try to understand Set timeout function.

Source Code:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>

        function simulateAsyncOperation(callback) {
            "use strict";

            setTimeout(function () {
                callback("The asynchronous completed");
            }, 50000);
```

```
        }

        simulateAsyncOperation(function (result) {
            console.log(result);
        });

    </script>
</body>
</html>
```

Output:



Learning outcome(co4):

Demonstrate the use of JavaScript to fulfill the essentials of frontend development to back-end development.