Task-1

Aim: Variables and Data Types

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

Description:

- JavaScript is a "duck-typed" language, and therefore every variable is defined using the var, let and const keywords, and can contain all types of variables.
- Datatype in JavaScript:
  - In JavaScript, the Number type can be both a floating-point number and an integer.
  - Boolean variables can only be equal to either true or false.
  - There are also data type such as an array and object.
  - There are two special types called –
    - **Undefined:** When a variable is used without first defining a value for it
    - **Null**: The null value is used when a variable should be marked as empty.

Source Code:

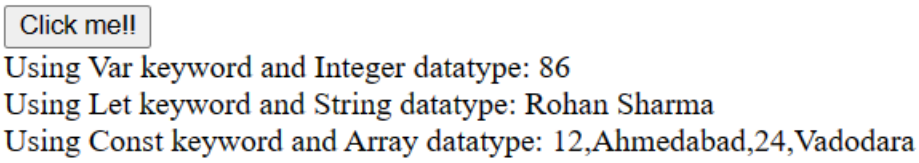
```
<!DOCTYPE html>
<head>
    <title>Variable and DataType</title>
</head>
<body>
    <button onclick="myFun()"> Click me!! </button><br>
    <div id="demo"></div><br>
    <script>
        function myFun(){
            //Use of var keyword
            var num1=15;
            var num2=71;
            var result=num1+num2;
            console.log(result);

            //Use of let keyword
            let firstName="Rohan";
            let lastName="Sharma";
            let name=firstName.concat(" ",lastName);
            console.log(name);

            //Use of const keyword
            const arr=[12,'Ahmedabad',24,'Vadodara']
            console.log(arr);

            //Display content
            var outputElement = document.getElementById("demo");
            outputElement.innerHTML = "Using Var keyword and Integer datatype: "+ result + "<br>";
            outputElement.innerHTML += "Using Let keyword and String datatype: "+ name + "<br>";
            outputElement.innerHTML += "Using Const keyword and Array datatype: "+ arr + "<br>";
        }
    </script>
</body>
</html>
```

Output:



Task-2

Aim: Operators and Expressions

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

Description:

There are following types of operators in JavaScript for performing operations such as addition, subtraction, multiplication, increment, decrement, comparison and logical and bitwise shift and other operations.

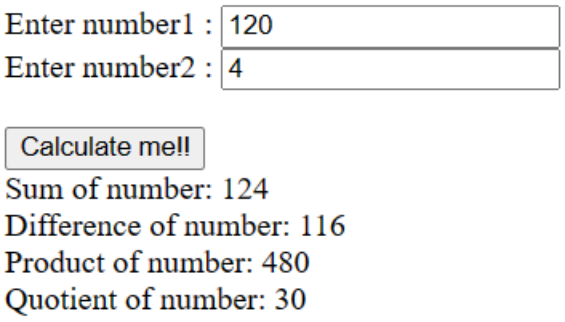
1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Bitwise Operators
4. Logical Operators
5. Assignment Operators
6. Special Operators

Source Code:

```
<head>
    <title>Operators and Expression</title>
</head>
<body>
    Enter number1 : <input type="text" id="numId1"><br>
    Enter number2 : <input type="text" id="numId2"><br><br>
    <button onclick="myMath()"> Calculate me!! </button><br>
    <div id="demo"></div><br>
    <script>
        function myMath(){
            let num1=parseInt(document.getElementById('numId1').value);
            let num2=parseInt(document.getElementById('numId2').value);
            let sum=num1+num2;
            let diff=num1-num2;
            let product=num1*num2;
            let quotient=num1/num2;

            //Display content
            var output = document.getElementById("demo");
            output.innerHTML += "Sum of number: "+ sum + "<br>";
            output.innerHTML += "Difference of number: "+ diff + "<br>";
            output.innerHTML += "Product of number: "+ product + "<br>";
            output.innerHTML += "Quotient of number: "+ quotient + "<br>";
        }
    </script>
</body>
</html>
```

Output:



Task-3

Aim: Control Flow

- 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:

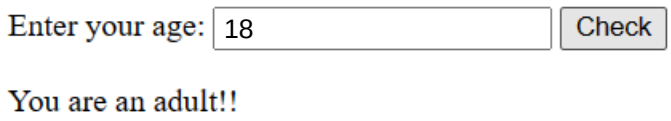
The JavaScript if-else statement is used for control flow of program and also used for execute the code whether condition is true or false. There are three forms of if statement in JavaScript:

1. If Statement
2. If else statement
3. if else if statement

Source Code:

```
<!DOCTYPE html>
<head>
    <title>Operators and Expression</title>
</head>
<body>
    Enter your age: <input type="text" id="ageId">
    <button onclick="displayAge()">Check</button>
    <p id="demo"></p>
    <script>
        function displayAge() {
            var age = parseInt(document.getElementById('ageId').value);
            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!!";
            }
            var outputElement = document.getElementById("demo");
            outputElement.innerHTML = message;
        }
    </script>
</body>
</html>
```

Output:



Task-4

Aim: Functions

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

Description:

- A JavaScript function is a block of code designed to perform a particular task.
- A JavaScript function syntax is as follows--

```
function name (parameter1, parameter2, parameter3) {
  // code to be executed
}
```

Source Code:

```
<!DOCTYPE html>
<head>
    <title>Functions</title>
</head>
<body>
    <div id="demo"></div>
    <div id="salaryId"></div>
    <script>
        function findSalary(salaries) {
          var minSalary = Math.min(...salaries);
          var maxSalary = Math.max(...salaries);
          return {
            min: minSalary,
            max: maxSalary
          };
        }
        var salaries = [15000, 30000, 22000, 14000, 43000];
        var result = findSalary(salaries);
        var array=document.getElementById("demo");
        array.innerHTML="Salary:"+ salaries +"<br>";
        var output=document.getElementById("salaryId");
        output.innerHTML="Maximum salary:"+ result.max+"<br>";
        output.innerHTML+="Minimum salary:"+ result.min+"<br>";
        console.log("Minimum salary:", result.min);
        console.log("Maximum salary:", result.max);
      </script>
  </body>
</html>
```

Output:

Salary:15000,30000,22000,14000,43000
Maximum salary:43000
Minimum salary:14000

Task-5

Aim: Arrays and Objects

- 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:

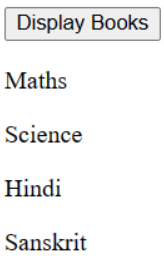
- Objects in JavaScript are same as variables but objects can contain many values.
- An array can hold many values under a single name, and you can access the values by referring to an index number.
- Syntax of array:
  - const *array_name* = [*item1*, *item2*, ...];

Source Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Arrays and Objects</title>
  </head>
  <body>
  <button onclick="displayBooks(['Maths', 'Science', 'Hindi', 'Sanskrit'])">Display Books</button>
  <div id="bookId"></div>
  <script>
    function displayBooks(books) {
      var booksContainer = document.getElementById("bookId");
      booksContainer.innerHTML = "";

      for (var i = 0; i < books.length; i++) {
        var book = books[i];
        var bookElement = document.createElement("p");
        bookElement.textContent = book;
        booksContainer.appendChild(bookElement);
      }
    }
  </script>
</body>
</html>
```

Output:



Task-6

Aim: Scope and Hoisting

- 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 determines the accessibility and visibility of variables.
- JavaScript has three types of scope:
  1. Block scope: let and const keyword provide Block scope in JavaScript. Variables declared inside a {} block cannot be accessed from outside the block.
  2. Function scope: JavaScript has function scope: Each function creates a new scope. Variables defined inside a function are not accessible (visible) from outside the function.
  3. Global scope : Variables declared Globally (outside any function) have Global Scope. Global variables can be accessed from anywhere in a JavaScript program.

Source Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>Scope and Hosting</title>
  </head>
  <body>
    <div id="demoId"></div>
    <script>
        let carName;
        function display() {
            carName="Swift Desire";
        }
        display();
        console.log(carName);
        document.getElementById('demoId').innerHTML+="The car name is "+carName+'<br>';
    </script>
</body>
</html>
```

Output:

The car name is Swift Desire

Task-7

Aim: DOM Manipulation

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

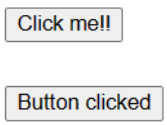
Description:

- When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.
- The HTML DOM can be accessed with JavaScript (and with other programming languages).
- In the DOM, all HTML elements are defined as **objects**.
- The programming interface is the properties and methods of each object:
  1. A **property** is a value that you can get or set (like changing the content of an element).
  2. A **method** is an action you can do (like add or deleting an HTML element).

Source Code:

```
<!DOCTYPE html>
<html>
<head>
  <title>DOM Manipulation</title>
  </head>
  <body>
    <button id="btn">Click me!!</button><br>
    <script>
        const button=document.getElementById('btn');
        button.addEventListener('click',function displayContent(){
            let initialText="Click me";
            if (button.textContent.toLowerCase().includes(initialText.toLowerCase())) {
                button.textContent = 'Button clicked';
            }
            else {
                button.textContent = initialText;
        };
        });
    </script>
</body>
</html>
```

Output:



Task-8

Aim: Error Handling

- 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.

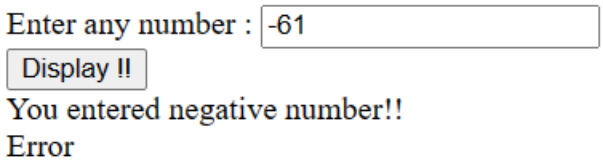
Description:

- When executing JavaScript code, different errors can occur. Errors can be coding errors made by the programmer, errors due to wrong input, and other unforeseeable things.
- We could deal with error by using given below keywords:
  1. The try statement defines a code block to run (to try).
  2. The catch statement defines a code block to handle any error.
  3. The finally statement defines a code block to run regardless of the result.
  4. The throw statement defines a custom error.

Source Code:

```
<!DOCTYPE html>
<head>
    <title>Error Handling</title>
</head>
<body>
    Enter any number : <input type="text" id="numId"><br>
    <button onclick="mydisplay()"> Display !! </button><br>
    <div id="demoId"></div><br>
    <script>
        function mydisplay(){
            let num=parseInt(document.getElementById('numId').value);
            let msg=document.getElementById('demoId');
            msg.innerHTML="";
            function throwAnError() {
                if(num<0)
                throw new Error('You entered negative number!!');
            }
            try {
                throwAnError();
            } catch (error) {
                msg.innerHTML=error.message+'<br>';
                msg.innerHTML+=error.name;
                console.log(error.message);
                console.log(error.name); // Error
            }
        }
    </script>
</body>
</html>
```

Output:



Task-9

Aim: Asynchronous JavaScript

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

Description:

- Callbacks in JavaScript are functions that are passed as arguments to other functions. This is a very important feature of asynchronous programming, and it enables the function that receives the callback to call our code when it finishes a long task, while allowing us to continue the execution of the code.
- In the real-world callbacks are most often used with asynchronous functions.
- When using the JavaScript function setTimeout(), you can specify a callback function to be executed on time-out function.

Source Code:

```
<!DOCTYPE html>
<head>
    <title>Asynchronous JavaScript</title>
</head>
<body>
    <div id="demo"></div>
    <script>
        function myDisplayer(some) {
            document.getElementById("demo").innerHTML = "The multiplication of two numbers: "+some+'<br>';
          }

          function myCalculator(num1, num2) {
            let sum = num1 * num2;
            myDisplayer(sum);
          }

          myCalculator(15, 5);
          setTimeout(myFunction, 2000);

        function myFunction() {
            document.getElementById("demo").innerHTML += "Welcome to the world!!";
        }
    </script>
</body>
</html>
```

Output:

The multiplication of two numbers: 75

The multiplication of two numbers: 75
Welcome to the world!!

Learning Outcome:

From this practical, I learnt about various concepts of JavaScript which leads help to add logic to web application. And I also understand and fulfill the Course Outcome.

CO4: Demonstrate the use of JavaScript to fulfill the essentials of front-end development to back-end development.