Practical-1

Task-1

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

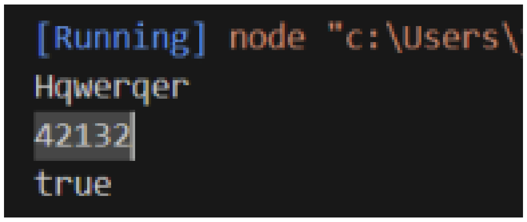
Theoretical Background:

- var and let create variables that can be reassigned another value. const creates "constant" variables that cannot be reassigned to another value.
- As a general rule, you should always declare variables with const, if you realize that the value of the variable needs to change, go back and change it to let. Use let when you know that the value of a variable will change.

Source Code:

```
// Using var
var example = 'Hqwerqer';
console.log(example);

// Using let
let exapmle1 = 42132;
console.log(exapmle1);

// Using const
const myConst = true;
console.log(myConst);
```

Output:



```
[Running] node "c:\Users\
Hqwerqer
42132
true
```

Task-2

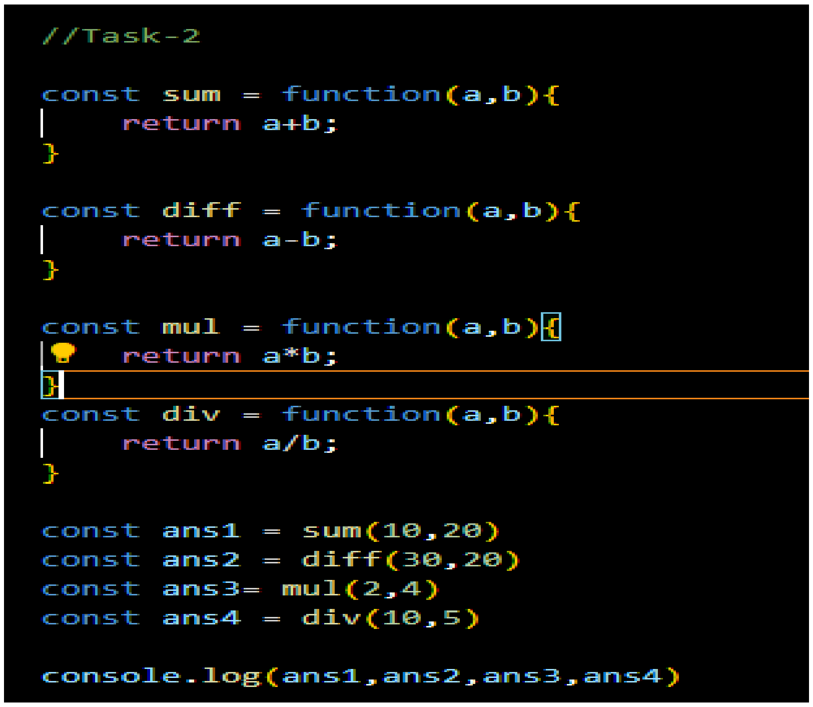
Aim:
Write a function that takes two numbers as arguments and returns their sum, difference, product, and quotient using arithmetic operators.

Theoretical Background:

➢ JavaScript Arithmetic Operators are the operators that operate upon the numerical values and return a numerical value.

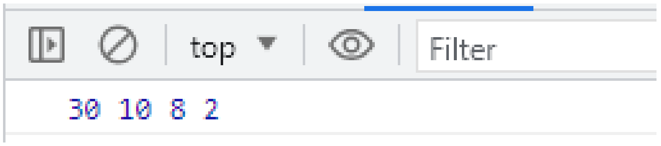
➢ Arithmetic operators like, Addition, Subtraction, Multiplication, Division, Modulus etc.

Source Code:



```
//Task-2

const sum = function(a,b){
    return a+b;
}

const diff = function(a,b){
    return a-b;
}

const mul = function(a,b){
    return a*b;
}
const div = function(a,b){
    return a/b;
}

const ans1 = sum(10,20)
const ans2 = diff(30,20)
const ans3= mul(2,4)
const ans4 = div(10,5)

console.log(ans1,ans2,ans3,ans4)
```

Output:



```
30 10 8 2
```

Task-3

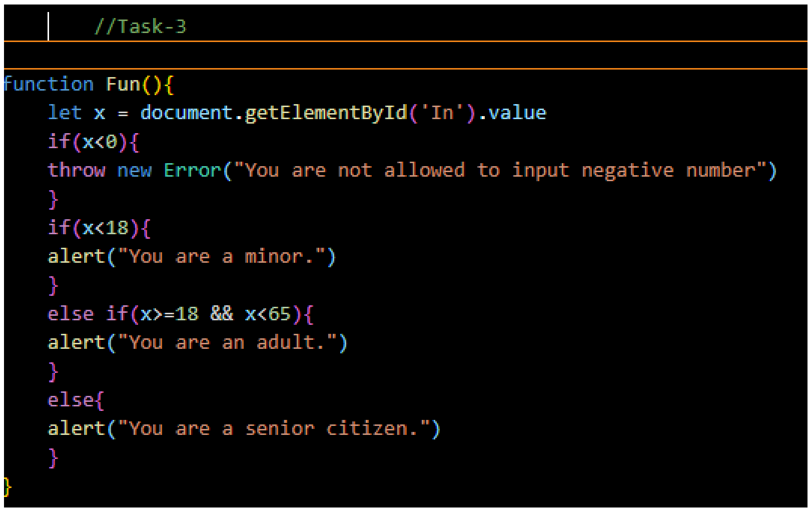
Aim:

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:

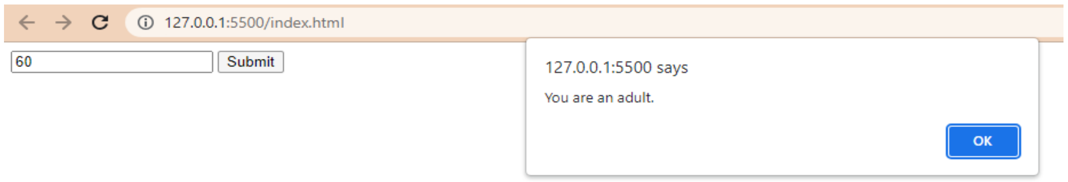
- ➢ The control flow is the order in which the computer executes statements in a script.
- ➢ Flow control is a technique used to regulate data transfer between computers or other nodes in a network.

Source Code:



```
    //Task-3

function Fun(){
    let x = document.getElementById('In').value
    if(x<0){
    throw new Error("You are not allowed to input negative number")
    }
    if(x<18){
    alert("You are a minor.")
    }
    else if(x>=18 && x<65){
    alert("You are an adult.")
    }
    else{
    alert("You are a senior citizen.")
    }
}
```

Output:


127.0.0.1:5500/index.html

60 Submit

127.0.0.1:5500 says

You are an adult.

OK

Task-4

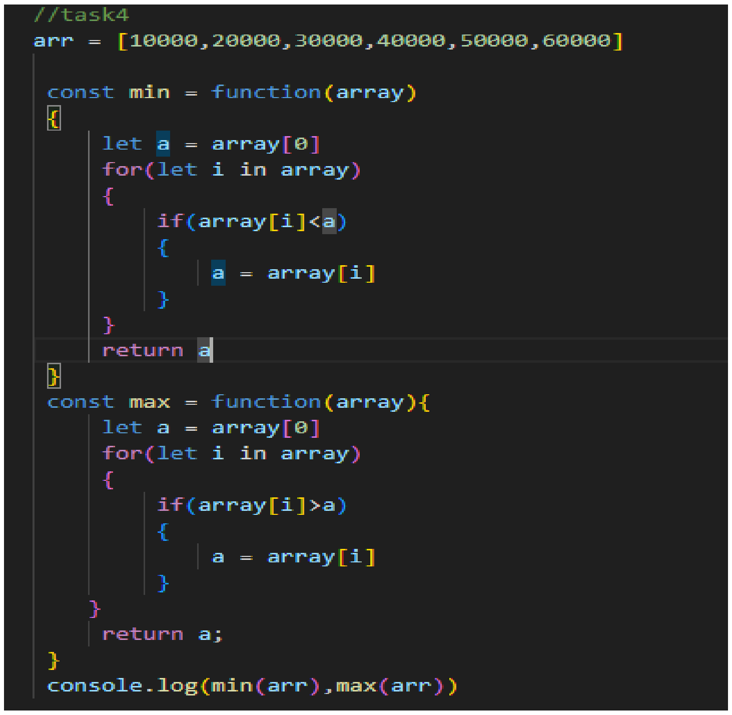
Aim:

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

Theoretical Background:

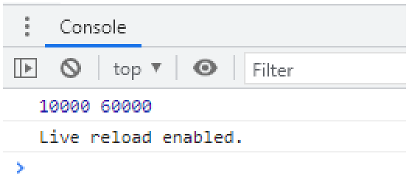
- ➢ A function in JavaScript is similar to a procedure—a set of statements that performs a task or calculates a value, but for a procedure to qualify as a function.
- ➢ A JavaScript function is defined with the function keyword, followed by a name, followed by parentheses ().
- ➢ Function names can contain letters, digits, underscores, and dollar signs (same rules as variables).

Source Code:



```
//task4
arr = [10000,20000,30000,40000,50000,60000]

 const min = function(array)
 {
     let a = array[0]
     for(let i in array)
     {
         if(array[i]<a)
         {
             a = array[i]
         }
     }
     return a
 }
 const max = function(array){
     let a = array[0]
     for(let i in array)
     {
         if(array[i]>a)
         {
             a = array[i]
         }
    }
     return a;
 }
 console.log(min(arr),max(arr))
```

Output:



```
10000 60000
Live reload enabled.
```

Task-5

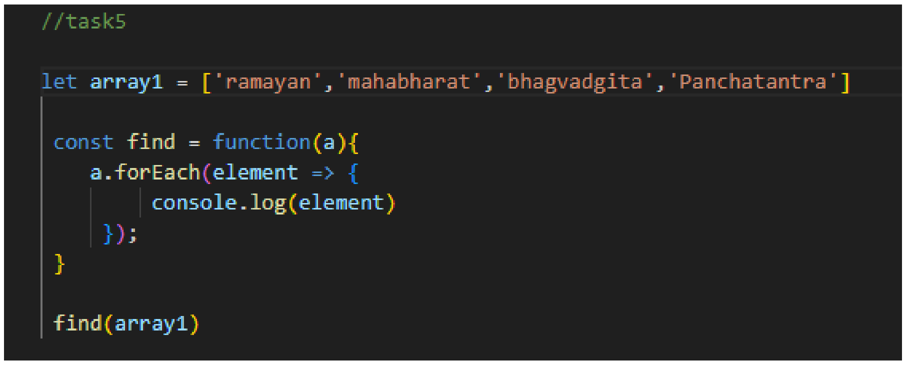
Aim:

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:

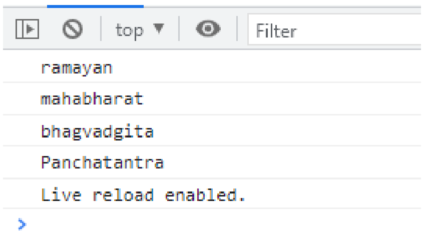
- ➢ Both objects and arrays are considered “special” in JavaScript. Objects represent a special data type that is mutable and can be used to store a collection of data (rather than just a single value).
- ➢ Arrays are a special type of variable that is also mutable and can also be used to store a list of values.

Source Code:



```
//task5

let array1 = ['ramayan','mahabharat','bhagvadgita','Panchatantra']

const find = function(a){
    a.forEach(element => {
        console.log(element)
    });
}

find(array1)
```

Output:



```
ramayan
mahabharat
bhagvadgita
Panchatantra
Live reload enabled.
>
```

Task-6

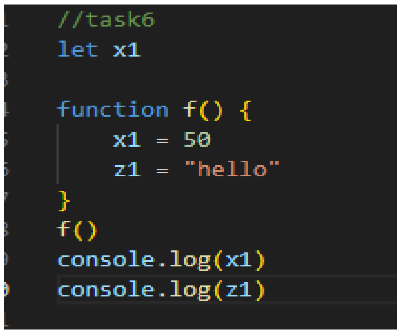
Aim:

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:

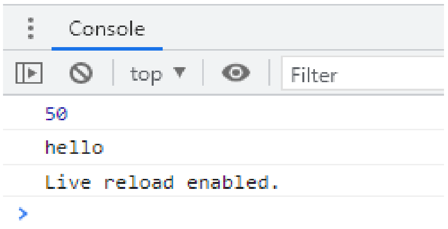
- ➢ Hoisting is a JavaScript mechanism where variables and function declarations are moved to the top of their scope before code execution.
- ➢ This means that no matter where functions and variables are declared, they are moved to the top of their scope regardless of whether their scope is global or local.

Source Code:



```
//task6
let x1

function f() {
    x1 = 50
    z1 = "hello"
}
f()
console.log(x1)
console.log(z1)
```

Output:



```
50
hello
Live reload enabled.
>
```

Task-7

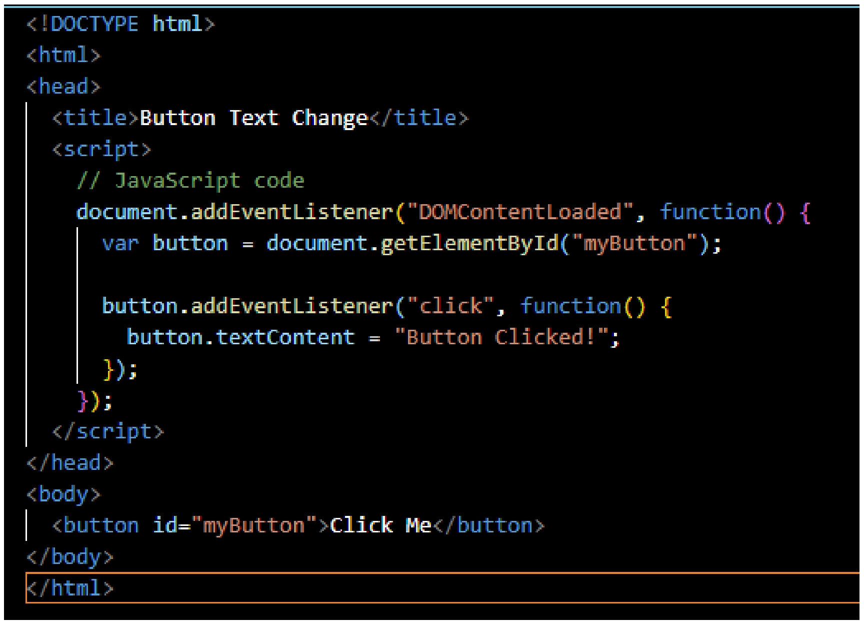
Aim:

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:

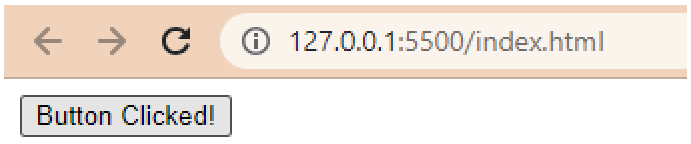
- ➢ DOM manipulation in JavaScript is the process of interacting with the DOM API to change or modify an HTML document that will be displayed in a web browser.
- ➢ By manipulating the DOM, we can create web applications that update the data in a web page without refreshing the page. The DOM stands for Document Object Model.

Source Code:



```html
<!DOCTYPE html>
<html>
<head>
  <title>Button Text Change</title>
  <script>
    // JavaScript code
    document.addEventListener("DOMContentLoaded", function() {
      var button = document.getElementById("myButton");

      button.addEventListener("click", function() {
        button.textContent = "Button Clicked!";
      });
    });
  </script>
</head>
<body>
  <button id="myButton">Click Me</button>
</body>
</html>
```

Output:



Task-8

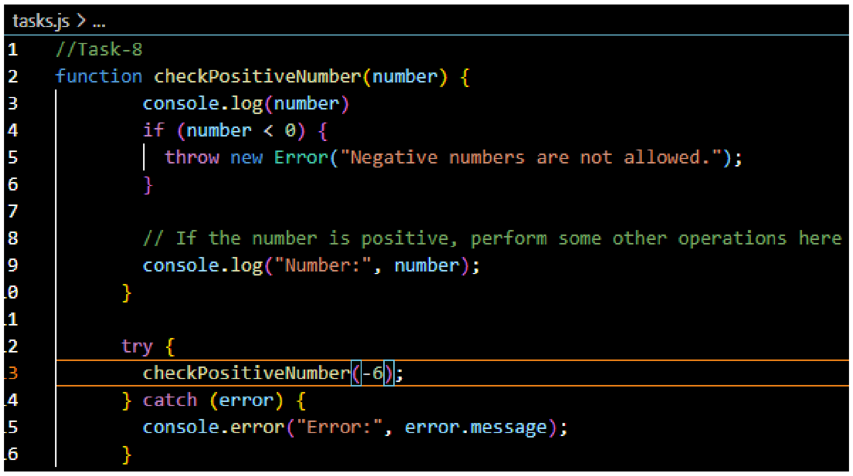
Aim:

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.

Theoretical Background:

- ➢ JavaScript provides an error-handling mechanism to catch runtime errors using try-catch-finally block.
- ➢ exception handling is a process or method used for handling the abnormal statements in the code and executing them. It also enables the flow control of the code/program.

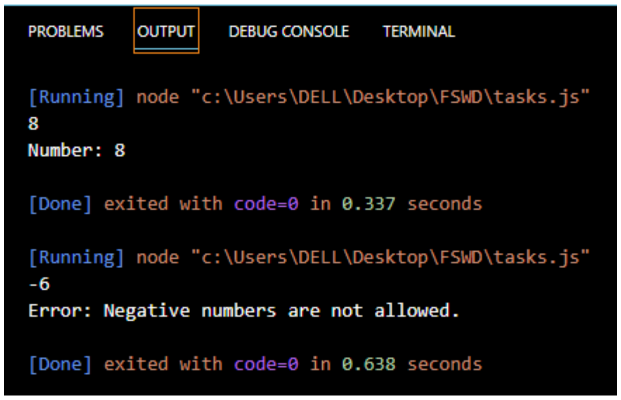
Source Code:



tasks.js > ...

```
//Task-8
function checkPositiveNumber(number) {
        console.log(number)
        if (number < 0) {
          throw new Error("Negative numbers are not allowed.");
        }

        // If the number is positive, perform some other operations here
        console.log("Number:", number);
      }

      try {
        checkPositiveNumber(-6);
      } catch (error) {
        console.error("Error:", error.message);
      }
```

Output:



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[Running] node "c:\Users\DELL\Desktop\FSWD\tasks.js"
8
Number: 8

[Done] exited with code=0 in 0.337 seconds

[Running] node "c:\Users\DELL\Desktop\FSWD\tasks.js"
-6
Error: Negative numbers are not allowed.

[Done] exited with code=0 in 0.638 seconds
```

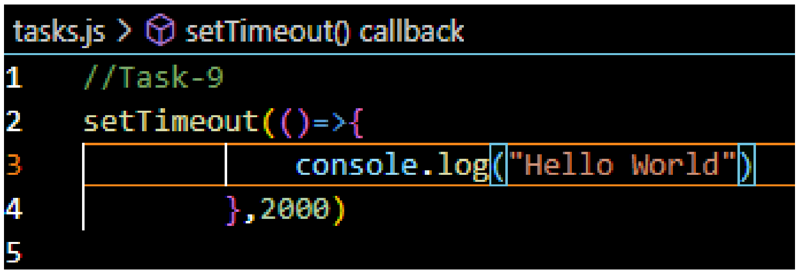
Task-9

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

Theoretical Background:

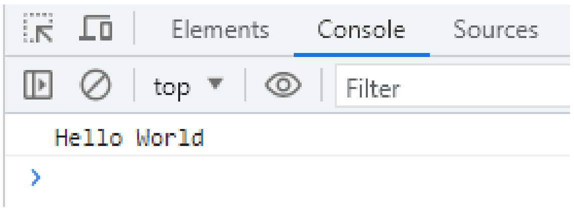
➢ Asynchronous programming is a technique that enables your program to start a potentially long-running task and still be able to be responsive to other events while that task runs, rather than having to wait until that task has finished.
➢ the server might take some time to process the request while blocking the main thread making the web page unresponsive. That's where we can use Asynchronous JavaScript.

Source Code:



```
tasks.js > setTimeout() callback
//Task-9
setTimeout(()=>{
            console.log("Hello World")
        },2000)
```

Output:



```
Hello World
>
```