

Task-1

Aim:

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

Theoretical background:

JavaScript Variables can be declared in 4 ways:

- Automatically
- Using var
- Using let
- Using const

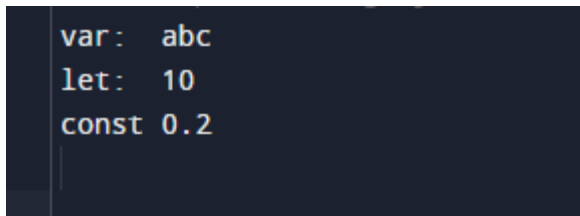
The var keyword was used in all JavaScript code from 1995 to 2015.

The let and const keywords were added to JavaScript in 2015.

The var keyword should only be used in code written for older browsers

Source Code:

```
var Bank_Name="abc";  
let current_balance=10;  
const intrest=0.2;  
console.log("var: ",Bank_Name);  
console.log("let: ",current_balance);  
console.log("const",intrest);
```

Output:

```
var:  abc  
let:  10  
const 0.2
```

Task-2

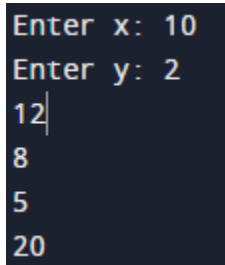
Aim:

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

Theoretical background:**Source Code:**

```
function arithOp(x,y)
{
    let a=parseInt(x)+parseInt(y),
        d=x/y,
        m=x*y,
        s=x-y;
    return [a,s,d,m];
}
```

```
let x=prompt("Enter x: ");
let y=prompt("Enter y: ");
let [a,d,m,s]=arithOp(x,y);
console.log(a)
console.log(d)
console.log(m)
console.log(s)
```

Output:

```
Enter x: 10
Enter y: 2
12
8
5
20
```

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:

A JavaScript function is a block of code designed to perform a particular task.

A JavaScript function is executed when "something" invokes it (calls it).

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).

The parentheses may include parameter names separated by commas:

(parameter1, parameter2, ...)

The code to be executed, by the function, is placed inside curly brackets: { }

Source Code:

```
let age=parseInt(prompt("Enter your Age: "));
if(age<18){
    console.log("You are a minor.");
}
else if(age>=18 && age<65)
{
    console.log("You are an adult.");
}
else{
    console.log("You are a senior citizen.");
}
```

Output:

You are an adult.

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:

It is a common practice to declare arrays with the const keyword.

An array is a special variable, which can hold more than one value.

JavaScript array is an object that represents a collection of similar type of elements.

There are 3 ways to construct array in JavaScript

- 1) By array literal
- 2) By creating instance of Array directly (using new keyword)
- 3) By using an Array constructor (using new keyword)

Source Code:

```
function find_max(arr)
{
    let i=0,j=arr.length-1;
    let max=arr[0];
    while(i!=j && i<arr.length && j>=0)
    {
        max=Math.max(arr[i],max);
        max=Math.max(arr[j],max);
        i++;
        j--;
    }
    return max;
}
console.log(find_max([1,2,30,4,-1,15,1,6]));
```

Output:

30

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:

The `console.log()` method outputs a message to the web console. The message may be a single string (with optional substitution values), or it may be any one or more JavaScript objects. Creates a new Function object. Calling the constructor directly can create functions dynamically but suffers from security and similar (but far less significant) performance issues to `eval()`. However, unlike `eval()`, the Function constructor creates functions that execute in the global scope only.

Source Code:

```
function disp_Title(books)
{
    for(let i in books)
    {
        console.log(books[i]);
    }
}
let books=['The Monk who sold his ferrari','Ikigai'];
disp_Title(books);
```

Output:

```
The Monk who sold his ferrari
Ikigai
```

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:

var variables have function scope, which means they can be accessed anywhere within the function they are declared in.

let variables have block scope, which means they can only be accessed within the block of code they are declared in.

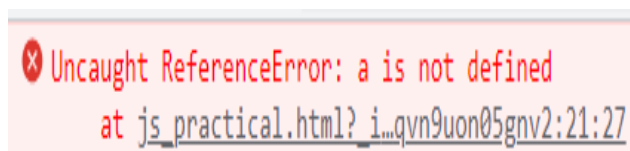
const variables also have block scope, but they cannot be reassigned once they are declared.

Source Code:

```
function var_test(){  
  var a="This is an var";  
}  
function let_test(){  
  let b="This is an let";  
}  
function const_test(){  
  const b="This is an const";  
}  
console.log("var",a)  
console.log("let",b);  
console.log("const",c);
```

Output:

```
function const_test(){  
  const b="This is an const"  
}  
console.log("var",a) ✖  
console.log("let",b);  
console.log("const",c);
```



✖ Uncaught ReferenceError: a is not defined
at js_practical.html?i...qvn9uon05gnv2:21:27

Task-7

Aim:

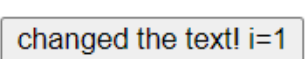
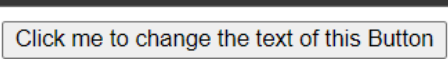
Create an HTML page with a button. Write JavaScript code that adds an event listener to the button and changes its text when clicked.

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:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport"
content="width=device-width, user-scalable=no, initial-scale=1.0, maximum-scale=1.0,
minimum-scale=1.0">
<meta http-equiv="X-UA-Compatible" content="ie=edge">
<title>Practical Output</title>
</head>
<body>
<button onclick="changeText()" id="btnChange">Click me to change the text of this
Button</button>
<script>
let i=0;
function changeText(){
i++;
document.getElementById('btnChange').innerText=`changed the text! i=${i}`;
}
</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.

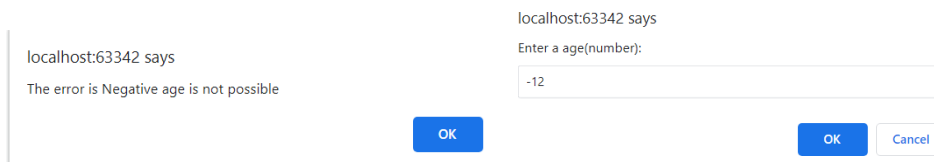
Theoretical background:

The throw statement is valid in all contexts where statements can be used. Its execution generates an exception that penetrates through the call stack. For more information on error bubbling and handling.

Source Code:

```
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<title>Practical Output</title>
</head>
<body>
<script>
try{
let n=prompt("Enter a age(number): ");
if(n>=0){
console.log("Your age is ",n);
}else throw new Error("Negative age is not possible");}
catch (err){
alert("The error is "+err.message);
}
</script></body></html>
```

Output:



Course Outcome:

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