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.

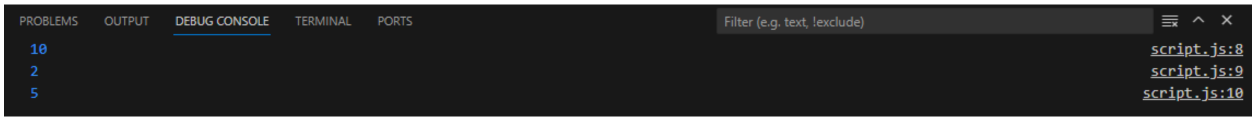
Description:

var variables can be redeclared and reassigned**, let** variables can be reassigned but not redeclared in the same scope, and **const** variables cannot be redeclared or reassigned.

Source Code:

```
let a = 10;
var n;
const m=5;
n=2;
console.log(a);
console.log(n);
console.log(m);
```

Output:



```
10                                                    script.js:8
2                                                     script.js:9
5                                                     script.js:10
```

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:

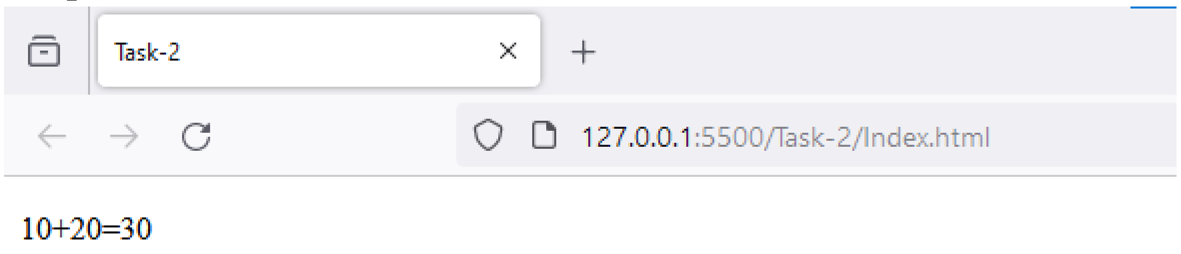
operators are symbols that are used to perform operations on operands.
An **expression** is a block of code that evaluates to a value. A **statement** is any block of code that is performing some action.

Source Code:

```
let a = parseInt(prompt("Enter first number: "))+prompt("Enter the operator (+,-,*,/)")+parseInt(prompt("Enter second number : "))

document.getElementById("demo").innerText=`${a}=${eval(a)}`
```

Output:



10+20=30

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:

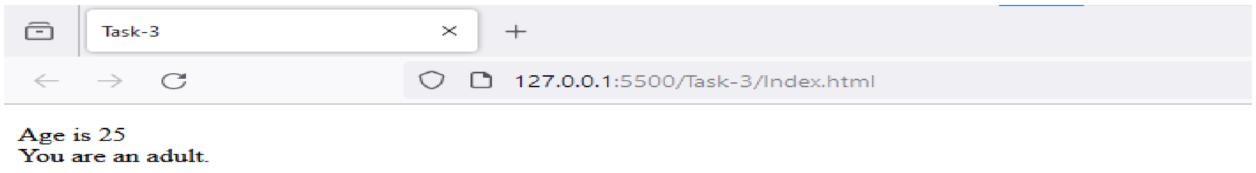
- Use `if` to specify a block of code to be executed, if a specified condition is true

- Use `else` to specify a block of code to be executed, if the same condition is false
- Use `else if` to specify a new condition to test, if the first condition is false

Source Code:

```
let a = parseInt(prompt("Enter your age: "))
document.getElementById("demo").innerText+="Age is "+a;
if(a<18)
    document.getElementById("demo").innerText+="\nYou are a minor."
else if(a>18 && a<65)
    document.getElementById("demo").innerText+="\nYou are an adult."
else
    document.getElementById("demo").innerText+="\nYou are a senior citizen."
```

Output:



```
Age is 25
You are an adult.
```

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.

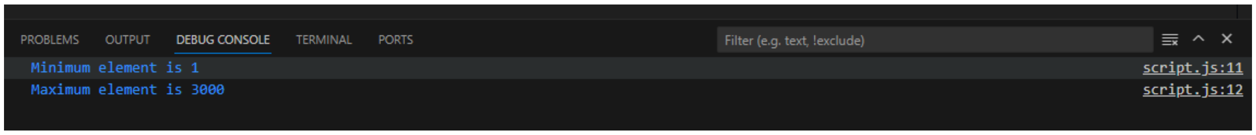
Source Code:

```
function getMinMax(arr) {
    const minmax = {};
    arr.sort((a, b) => a - b);
    minmax.min = arr[0];
    minmax.max = arr[arr.length - 1];
    return minmax;
}
```

```
function main() {
    const arr = [1000, 11, 445, 1, 330, 3000];
    const minmax = getMinMax(arr);
    console.log("Minimum element is " + minmax.min);
    console.log("Maximum element is " + minmax.max);
}

main();
```

Output:



```
Minimum element is 1                                  script.js:11
Maximum element is 3000                               script.js:12
```

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:

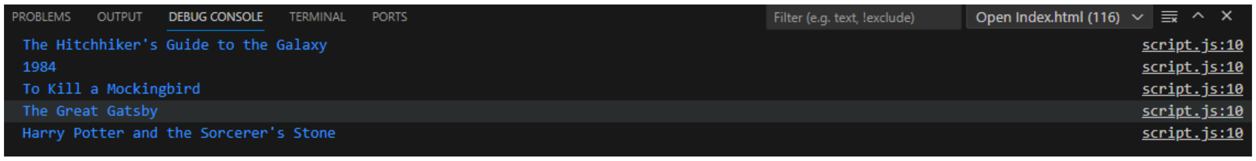
An array can hold many values under a single name, and you can access the values by referring to an index number.

An object is a collection of properties, and a property is an association between a name (or key) and a value. A property's value can be a function, in which case the property is known as a method.

Source Code:

```
const favoriteBooks = [
    "The Hitchhiker's Guide to the Galaxy",
    "1984",
    "To Kill a Mockingbird",
    "The Great Gatsby",
    "Harry Potter and the Sorcerer's Stone",
  ];
  function displayBooks(bookArray) {
    for (let i = 0; i < bookArray.length; i++) {
      console.log(bookArray[i]);
    }
  }
  displayBooks(favoriteBooks);
```

Output:


```
The Hitchhiker's Guide to the Galaxy          script.js:10
1984                                          script.js:10
To Kill a Mockingbird                         script.js:10
The Great Gatsby                              script.js:10
Harry Potter and the Sorcerer's Stone         script.js:10
```

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:

There are two types of scopes.

- Global Scope: Scope outside the outermost function attached to the window.
- Local Scope: Inside the function being executed.

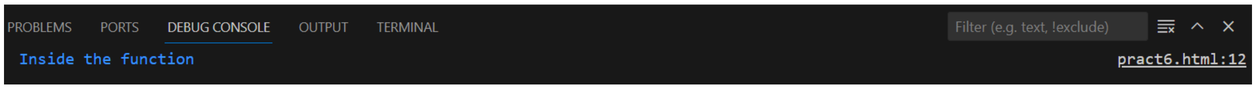
Hoisting: It is a concept that enables us to extract values of variables and functions even before initializing/assigning value without getting errors and this is happening due to the 1st phase (memory creation phase) of the Execution Context.

Source Code:

```
function testScope() {
      var variableInsideFunction = "I am inside the function.";
      console.log("Inside the function")
    }

    testScope();
    // Uncommenting the line below will result in an error
    // console.log(variableInsideFunction);
```

Output:



```
Inside the function                                                        pract6.html:12
```

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:

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

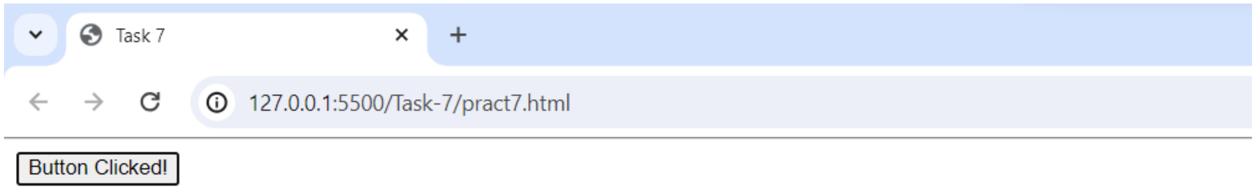
- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements.**

Source Code:

```
document.getElementById("myButton").addEventListener("click", function() {
      this.innerText = "Button Clicked!";
    });
```

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. To solve error we use following concepts:

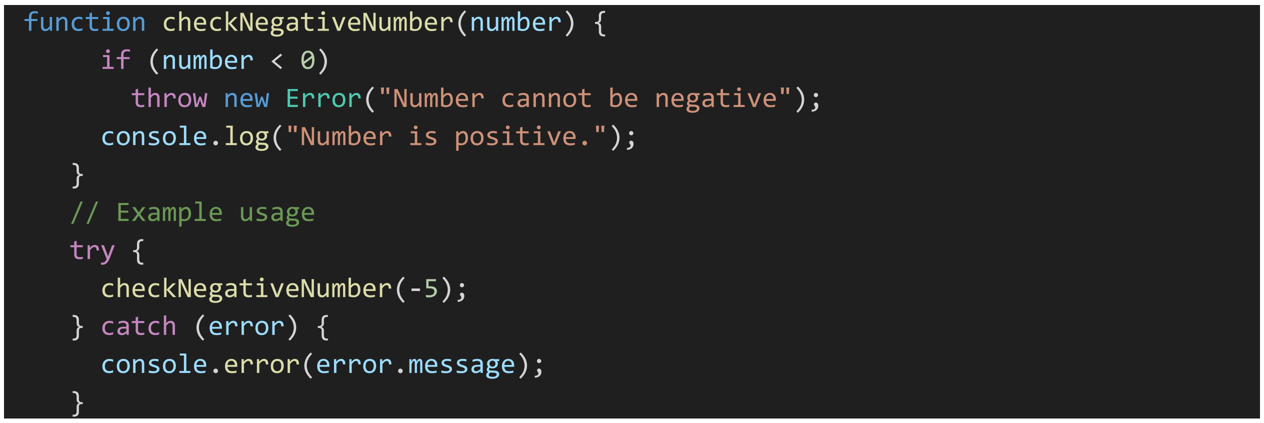
The `try` statement defines a code block to run (to try).

The `catch` statement defines a code block to handle any error.

The `finally` statement defines a code block to run regardless of the result.

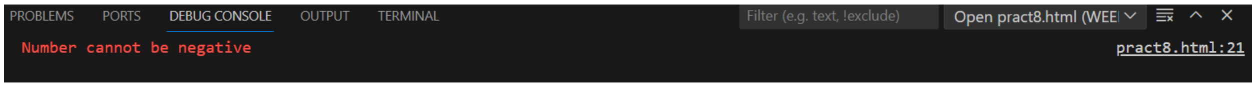
The `throw` statement defines a custom error.

Source Code:



```
function checkNegativeNumber(number) {
      if (number < 0)
        throw new Error("Number cannot be negative");
      console.log("Number is positive.");
    }
    // Example usage
    try {
      checkNegativeNumber(-5);
    } catch (error) {
      console.error(error.message);
    }
```

Output:



```
Number cannot be negative                                                    pract8.html:21
```

Task-9

Aim:

Asynchronous JavaScript

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

Description:

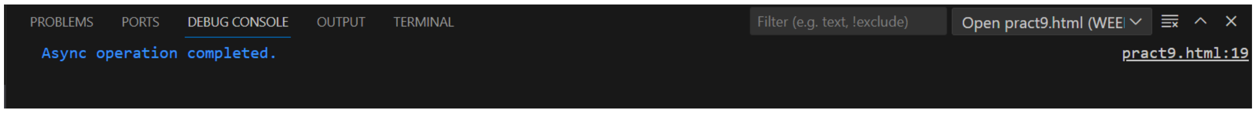
asynchronous programming is an essential concept in JavaScript that allows your code to run in the background without blocking the execution of other code. Developers can create more efficient and responsive applications by using features like callbacks, async/await, and promises.

Source Code:

```
function simulateAsyncOperation(callback) {
      setTimeout(function() {
        let result = "Async operation completed.";
        callback(result);
      }, 2000);
    }

    // Example usage
    simulateAsyncOperation(function(result) {
      console.log(result);
    });
```

Output:



```
Async operation completed.                                        pract9.html:19
```

Learning Outcome:

Overall, the provided code examples cover various important aspects of web development, including HTML structure, event handling, error management, and asynchronous JavaScript.