# Scopes

A Scope in JS defines the Accessibility or life or Visibility of Variables and Functions.

1. Global Scope:

   Variable declared globally (out side function) have globally. Scope means can be access from any where.

   Var have global Scope and Function Scope.

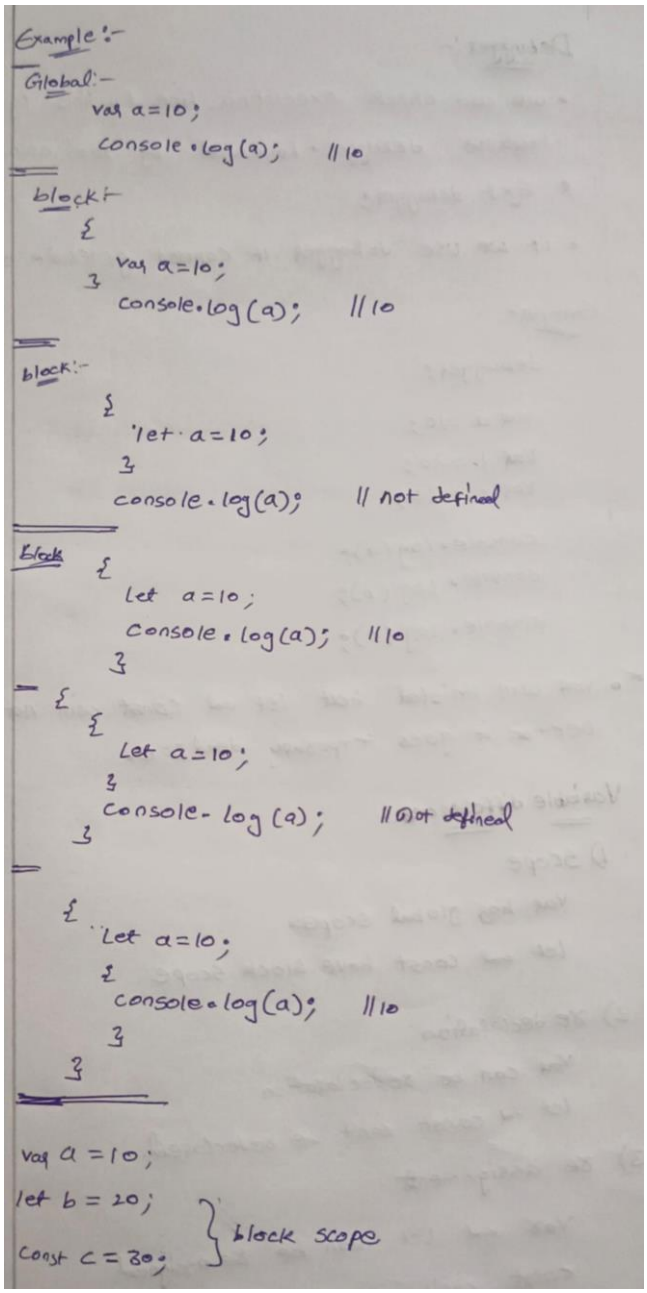
2. Block Scope:

   Variables declared in a block have block scope means that can't be accessed outside of the block.

   Only var have global scope the remaining let and const have block scope.

3. Local Scope:

   Variables declared within the function have local scope. They can only be accessed with in the function.



# Debugger

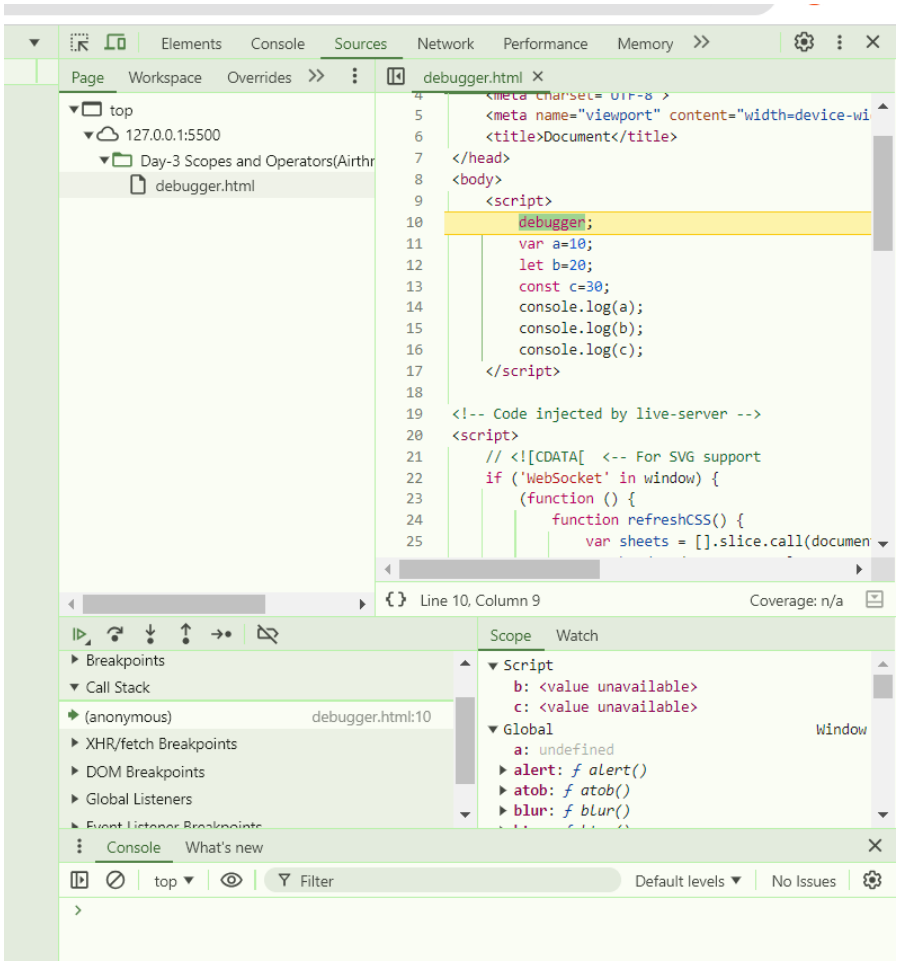
We can check Execution line by line by using keyword debugger followed by semi colon (:)

**syntax:**

```
debugger;
```

**Example:**

```
<script>
    debugger;
    var a=10;
    let b=20;
    const c=30;
    console.log(a);
    console.log(b);
    console.log(c);
</script>
```



## Variable Difference:

1) Scope
   var has global scope
   Let and const have block scope

2) Re declaration
   Var can be re declared
   Let and const can't be re declared

3) Re assignment
   Var and let can be re assigned
   Const can't be re assigned

# Operators

Javascript operators are used to perform different types of mathematical and logical computations.

(or)

In JavaScript, an **operator** is a symbol that performs an operation on one or more operands, such as variables or values, and returns a result. Let us take a simple expression **4 + 5** is equal to 9. Here 4 and 5 are called **operands**, and '+' is called the **operator**.

Types:

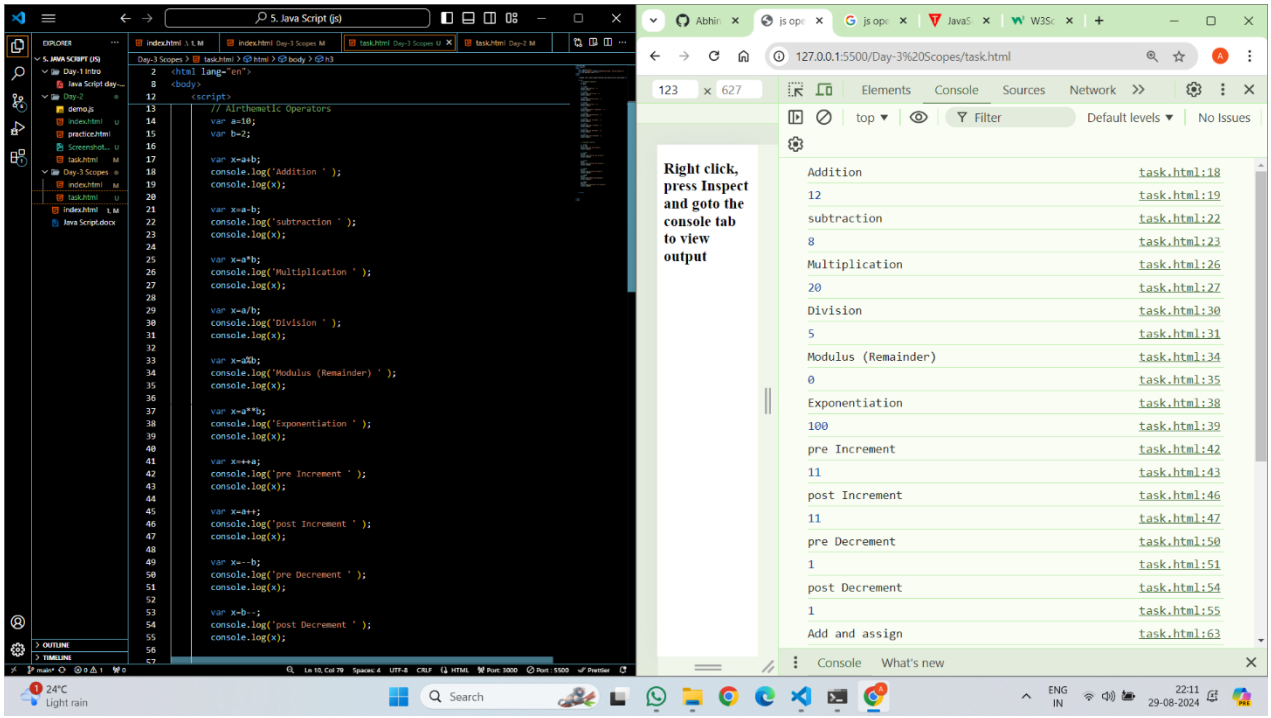
1. Airthmetic Operators
2. Assignment Operator
3. Comparision Operator
4. Logical Operator
5. Ternary Operator
6. Bitwise Oberator
7. String Operator
8. Typeof Operator

# Arithmetic operators

Arithmetic operators are used to perform **arithmetic operations** between variables or values.

| Operator | Name | Example |
|---|---|---|
| + | Addition | 3 + 4 // 7 |
| - | Subtraction | 5 - 3 // 2 |
| * | Multiplication | 2 * 3 // 6 |
| / | Division | 4 / 2 // 2 |
| % | Remainder | 5 % 2 // 1 |
| ++ | Increment (increments by **1**) | ++5 or 5++ // 6 |
| -- | Decrement (decrements by **1**) | --4 or 4-- // 3 |
| ** | Exponentiation (Power) | 4 ** 2 // 16 |

## Example:



# Assignment Operators:

We use assignment operators to **assign** values to variables.

| Operator | Name | Example |
|---|---|---|
| = | Assignment Operator | a = 7; |
| += | Addition Assignment | a += 5; // a = a + 5 |
| -= | Subtraction Assignment | a -= 2; // a = a - 2 |
| *= | Multiplication Assignment | a *= 3; // a = a * 3 |
| /= | Division Assignment | a /= 2; // a = a / 2 |
| %= | Remainder Assignment | a %= 2; // a = a % 2 |
| **= | Exponentiation Assignment | a **= 2; // a = a**2 |

**Example:**