

MERN Stack Training

Tasks

JavaScript Language - An Introduction to JavaScript, Code structure

1. An Introduction to JavaScript:

- Task 1: Write a simple script that displays "Hello, World!" on the web page using an alert box.
- Task 2: Experiment with different data types in JavaScript (e.g., string, number, boolean) by declaring and logging them in the console.
- Task 3: Use the console to perform basic math operations like addition, subtraction, multiplication, and division.
- Task 4: Declare two strings and concatenate them using the + operator.
- Task 5: Use the typeof operator to check the data type of various variables.

2. Code structure:

- Task 6: Write a multi-line JavaScript comment and a single-line comment. Explain the difference.
- Task 7: Create a script with both semicolon-separated and not separated lines. Note any differences in behavior.
- Task 8: Use proper indentation to format a nested loop.
- Task 9: Declare multiple variables in a single line.
- Task 10: Place a script tag at the top and bottom of an HTML document. Note any differences in behavior.

The modern mode, "use strict", Variables

1. The modern mode, "use strict":

- Task 11: Write a script without using "use strict" and try to assign a value to an undeclared variable. Note the result.
- Task 12: Enable "use strict" mode and repeat the above action, noting the difference.

- Task 13: In “use strict” mode, try to delete a variable, function, or function parameter.
- Task 14: Assign a value to an undeclared variable without “use strict” and then with “use strict”.
- Task 15: Declare a variable with a reserved keyword in “use strict” mode.

2. Variables:

- Task 16: Declare variables using let, const, and var. Discuss when each should be used.
- Task 17: Attempt to reassign a const variable and observe the result.
- Task 18: Declare a variable without initializing it and print its value.
- Task 19: Assign a number, string, and boolean value to a variable and print its type using typeof.
- Task 20: Rename a variable and observe the outcome.

Data types, Basic operators, maths

1. Data types:

- Task 21: Create variables of different data types (e.g., string, number, boolean, null, undefined, object).
- Task 22: Use the typeof operator to determine the type of various variables.
- Task 23: Declare a symbol and print its type.
- Task 24: Assign the value null to a variable and check its type using typeof.
- Task 25: Differentiate between declaring a variable using var and let in terms of scope.

2. Basic operators, maths:

- Task 26: Convert a string to a number using both implicit and explicit conversion.
- Task 27: Convert a boolean to a string and vice versa.
- Task 28: Practice basic arithmetic operators (+, -, *, /, %).
- Task 29: Use the ++ and -- operators on a numeric variable.
- Task 30: Explore the precedence of operators by combining multiple operators in a single expression.

Comparisons, Conditional branching: if, '?'

1. Comparisons:

- Task 31: Compare two numbers using relational operators (>, <, >=, <=).
- Task 32: Use equality (==) and strict equality (===) operators to compare different data types and note the differences.
- Task 33: Compare two strings lexicographically.
- Task 34: Use the inequality (!=) and strict inequality (!===) operators to compare values.
- Task 35: Compare null and undefined using both == and ===.

2. Conditional branching: if, '?':

- Task 36: Write an if statement that checks if a number is even or odd.
- Task 37: Use nested if statements to classify a number as negative, positive, or zero.
- Task 38: Use the conditional (ternary) operator '?' to rewrite a simple if...else statement.
- Task 39: Check the validity of a variable using the ? operator.
- Task 40: Use the conditional operator to assign a value to a variable based on a condition.

Logical operators, Functions

1. Logical operators:

- Task 41: Evaluate various combinations of logical operators (&&, ||, !).
- Task 42: Use logical operators to write a condition that checks if a number is in a given range.
- Task 43: Use the NOT (!) operator to invert a boolean value.
- Task 44: Evaluate the short-circuiting nature of logical operators.
- Task 45: Compare two non-boolean values using logical operators and observe the result.

2. Functions:

- Task 46: Write a function that takes two numbers as arguments and returns their sum.
- Task 47: Create a function that calculates the area of a rectangle.
- Task 48: Declare a function without parameters and call it.
- Task 49: Write a function that returns nothing and observe the default return value.
- Task 50: Declare a function with default parameters and call it with different arguments.

3. Arrow Functions:

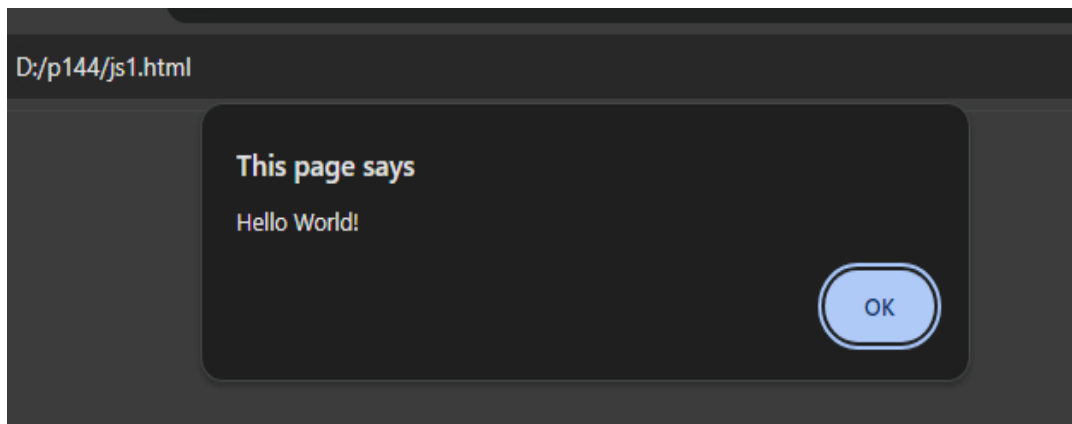
- Task 51: Declare a simple arrow function named greet that takes one parameter name and returns the string "Hello, name!". Test your function with various names.
- Task 52: Write an arrow function named add that takes two parameters and returns their sum. Validate your function with several pairs of numbers.
- Task 53: Declare an arrow function named isEven that checks if a number is even. If the number is even, it should return true; otherwise, false. Remember that if the arrow function body has a single statement, you can omit the curly braces.
- Task 54: Implement an arrow function named maxVal that takes two numbers as parameters and returns the larger number. Here, you'll need to use curly braces for the function body and the return statement.
- Task 55: Examine the behavior of the *this* keyword inside an arrow function vs a traditional function. Create an object named myObject with a property value set to 10 and two methods: multiplyTraditional using a traditional function and multiplyArrow using an arrow function. Both methods should attempt to multiply the value property by a number passed as a parameter. Check the value of this inside both methods.

1. An Introduction to JS:

Task 1:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let message="Hello World!";
    alert(message);
  </script>
</body>
</html>
```

Output:



Task 2:

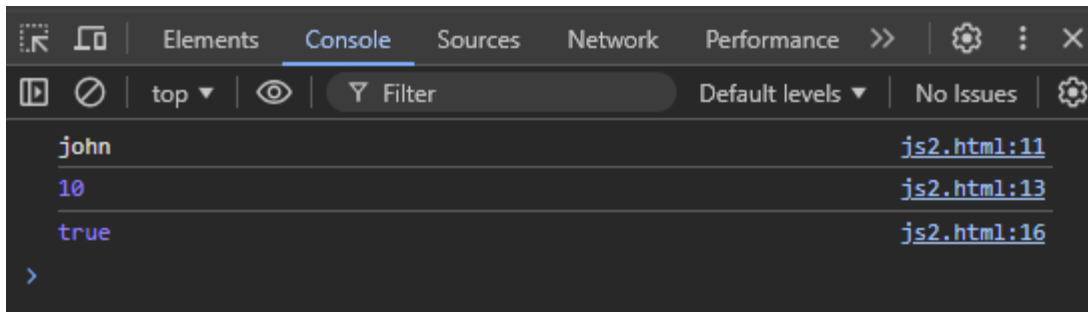
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
```

```

        var name = "john";
        console.log(name);
        let x= 10;
        console.log(x);
        let age = 20;
        let isAdult = age >=18;
        console.log(isAdult);
    </script>
</body>
</html>

```

Output:



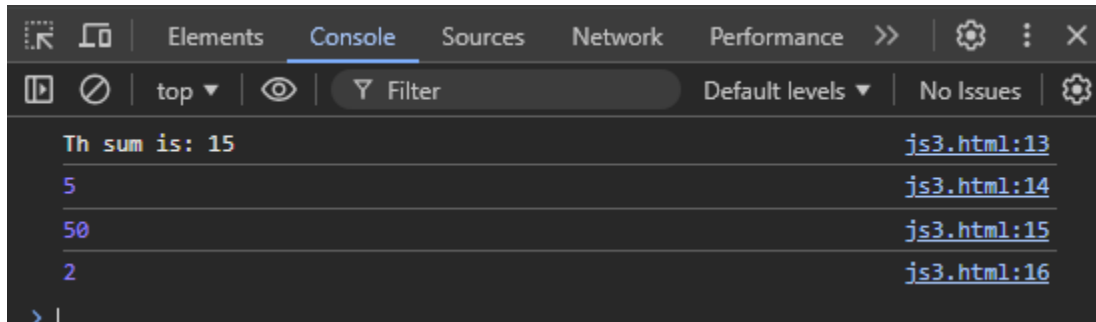
Task 3:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        let x=10;
        let y=5;
        let sum=x+y;
        console.log("Th sum is: "+sum);
        console.log("10" -5);
        console.log("10"*5);
        console.log("10"/5);
    </script>
</body>
</html>

```

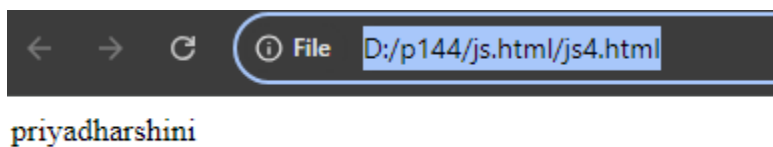
Output:



Task 4:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var name1="priya";
    var name2="dharshini";
    document.writeln(name1+name2);
  </script>
</body>
</html>
```

Output:

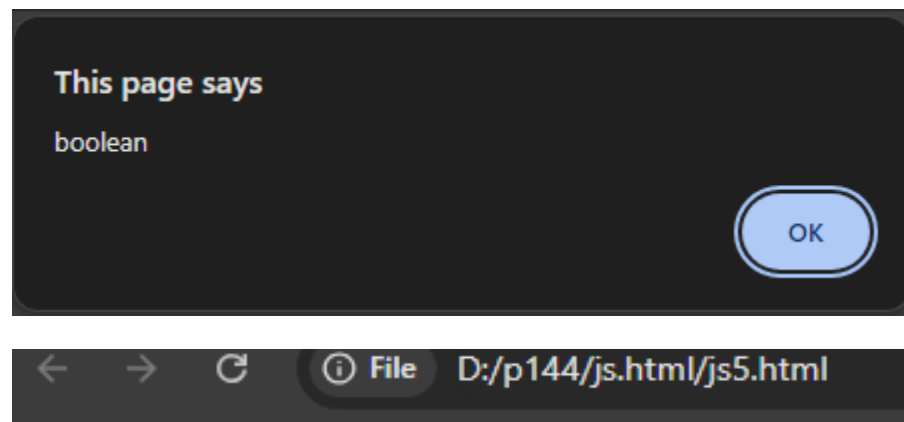


Task 5:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let val= true;
    alert(typeof val);
    document.writeln(typeof 7 + "<br>");
    document.writeln(typeof "hello");
  </script>
</body>
</html>
```

Output:



number
string

2.CODE STRUCTURE:

Task 6:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let p=50; //single line comment
    /* var name = "priya";
    console.log(name);
    document.writeln(name);*/
  </script>
</body>
</html>
```

1. Single-line Comment:

A **single-line comment** is used to comment out a single line of code or add a brief explanation to a line. It begins with `//`, and everything after `//` on that line will be treated as a comment.

```
// This is a single-line comment
```

2. Multi-line Comment:

A **multi-line comment** can span multiple lines. It begins with `/*` and ends with `*/`. Everything between these markers is treated as a comment, no matter how many lines it spans.

```
/*
```

This is a multi-line comment.

It can span multiple lines.

```
*/
```

Task 7:

1. Semicolon-separated statements:

When you explicitly use semicolons to separate statements, the behavior is predictable and clear.

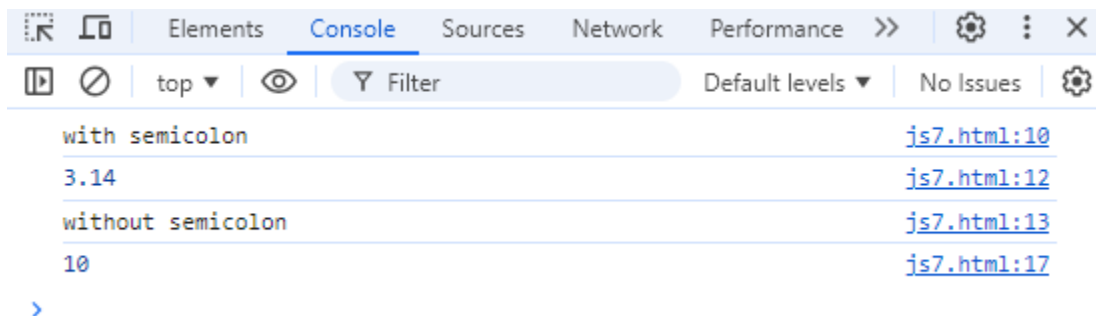
2. Non-semicolon-separated statements:

If you omit semicolons, JavaScript will often insert them automatically, but in some cases, it can lead to unexpected behavior.

Code:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    console.log("with semicolon");
    const PI=3.14;
    console.log(PI);
    console.log("without semicolon")
    let a = 5
    let b=5
    let sum = a+b
    console.log(sum)
  </script>
</body>
</html>
```

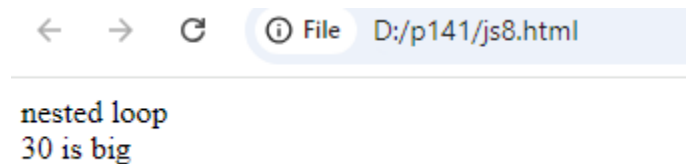
Output:



Task 8:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    document.writeln("nested loop");
    let a=10;
    let b=20;
    let c=30;
    if(a>b&&a>c){
      document.writeln("10 is big");
    }else{
      if(a<b&&b>c){
        document.writeln("20 is big");
      }else{
        document.writeln("30 is big");
      }
    }
  </script>
</body>
</html>
```

Output:



nested loop
30 is big

Task 9:

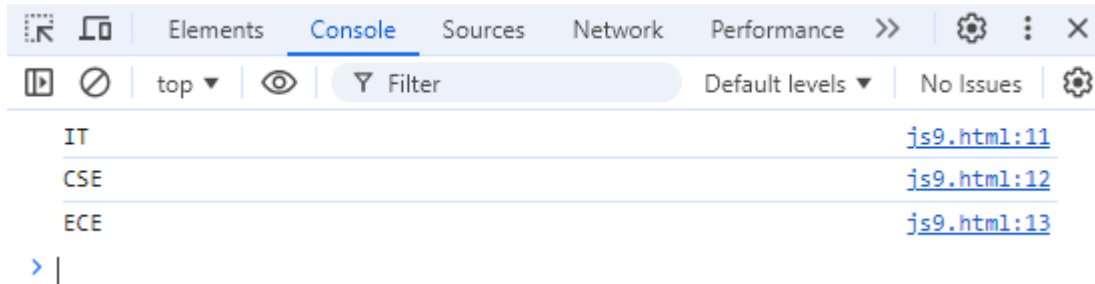
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var dept1="IT", dept2="CSE", dept3="ECE";
    console.log(dept1+"\n");
```

```

        console.log(dept2+"\n");
        console.log(dept3+"\n");
    </script>
</body>
</html>

```

Output:



Task 10:

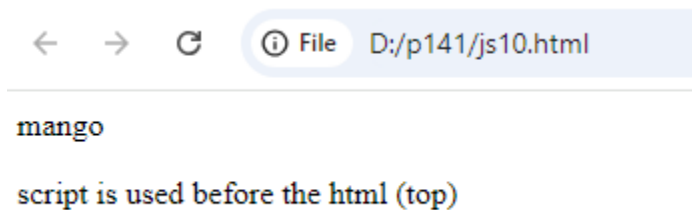
Script tag at the top

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
  <script>
    var fruit="mango";
    document.writeln(fruit+"<br>");
  </script>
</head>
<body>
  <p>script is used before the html (top)</p>
</body>
</html>

```

Output:

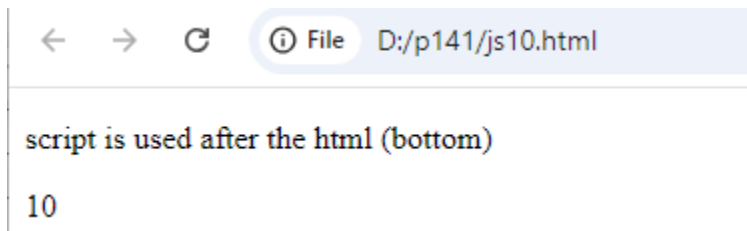


Script tag at the bottom

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>

</head>
<body>
  <p>script is used after the html (bottom)</p>
  <script>
    let x=10;
    document.writeln(x);
  </script>
</body>
</html>
```

Output:



Difference:

📌 **Script at the Top:** Useful for certain cases like preloading data or critical functionality, but can block the rendering of the page and cause delays.

📌 **Script at the Bottom:** Recommended for most general use cases, as it ensures that the HTML content is fully loaded before JavaScript is executed, improving page load performance and avoiding issues with DOM manipulation.

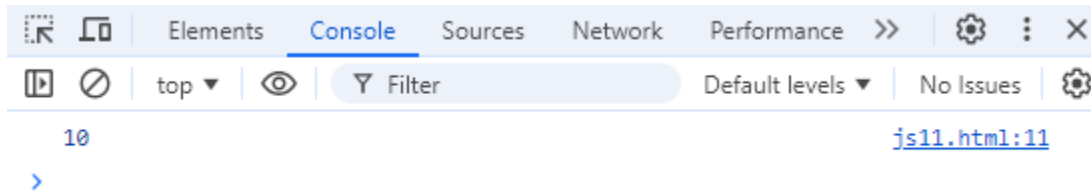
The modern mode, “use strict”:

Task 11:

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    x=10; //non-use strict mode
    console.log(x); //10
  </script>
</body>
</html>
```

Output:

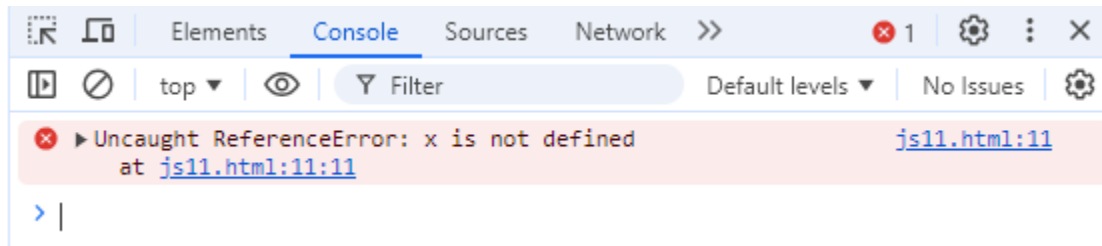


Task 12:

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    'use strict';
    x=10;
    console.log(x);
  </script>
</body>
</html>
```

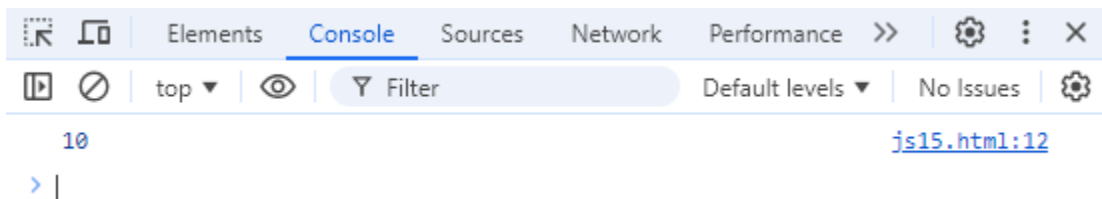
Output:



Task 15:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    'use strict';
    let x=10;
    console.log(x);
  </script>
</body>
</html>
```

Output:



Task 14:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
```

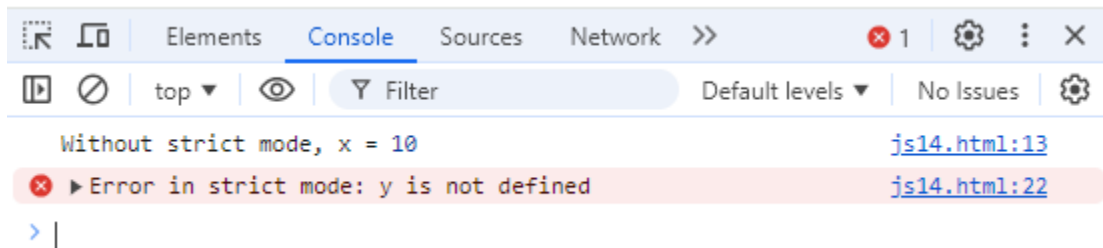
```
<script>
function withoutStrictMode() {
  x = 10;
  console.log("Without strict mode, x =", x);
}

function withStrictMode() {
  "use strict";
  try {
    y = 20;
    console.log("With strict mode, y =", y);
  } catch (error) {
    console.error("Error in strict mode:", error.message);
  }
}

withoutStrictMode();
withStrictMode();

</script>
</body>
</html>
```

Output:

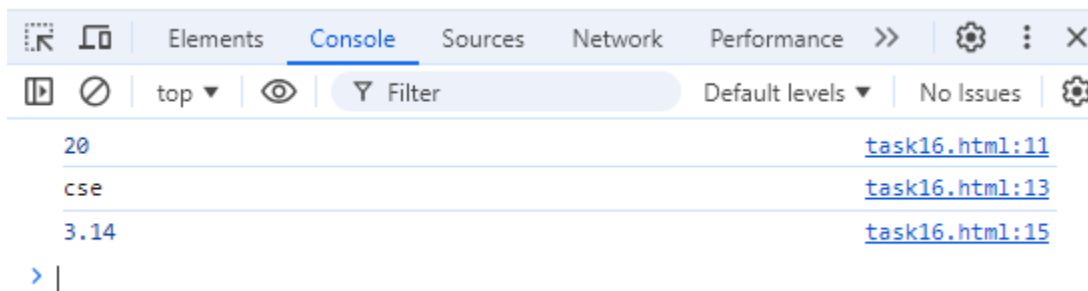


VARIABLES:

Task 16:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let x=20;
    console.log(x);
    var course='cse';
    console.log(course);
    const PI=3.14;
    console.log(PI);
  </script>
</body>
</html>
```

Output:



Task 17:

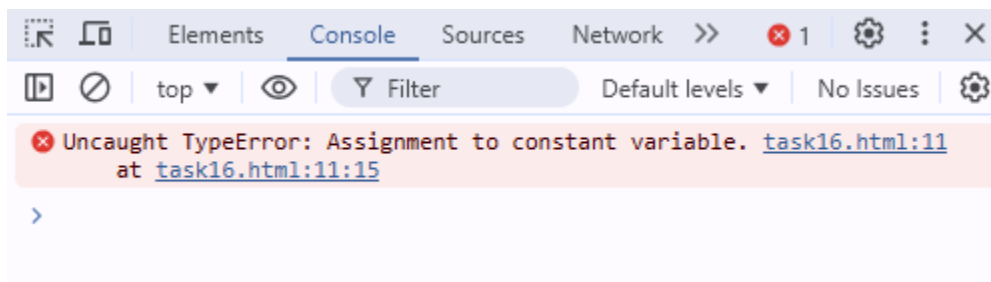
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
  <script>
    const PI=3.14;
    PI=3.75;
    console.log(PI);
  </script>
</body>
</html>

```

Output:



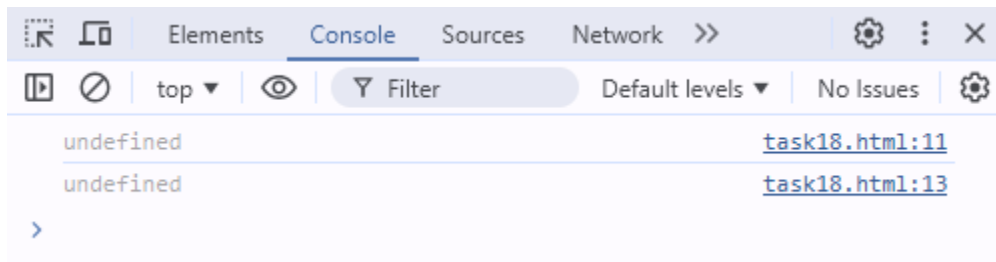
Task 18:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <string>
    let p;
    console.log(p);
    var dept;
    console.log(dept);
  </string>
</body>
</html>

```

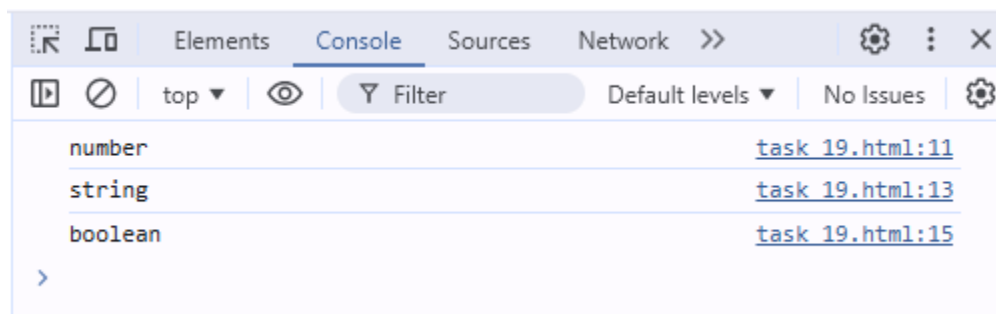
Output:



Task 19:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let x=5;
    console.log(typeof x);
    var name="priya";
    console.log(typeof name);
    let age=true;
    console.log(typeof age);
  </script>
</body>
</html>
```

Output:

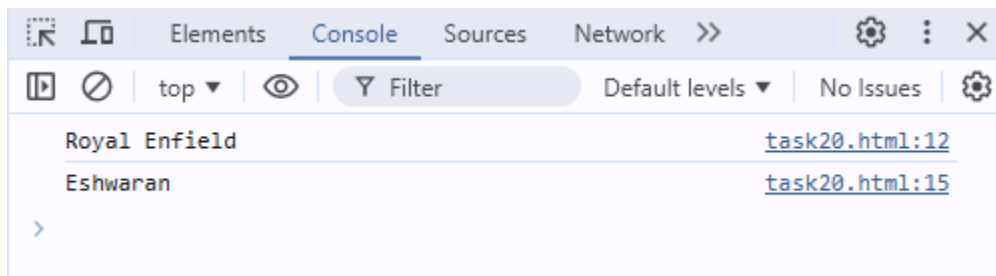


Task 20:

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    var bike="Royal Enfield";
    var twowheeler=bike;
    console.log(twowheeler);
    let name="Eshwaran";
    let myName=name;
    console.log(myName);
  </script>
</body>
</html>
```

Output:



DATA TYPES:

Task 21:

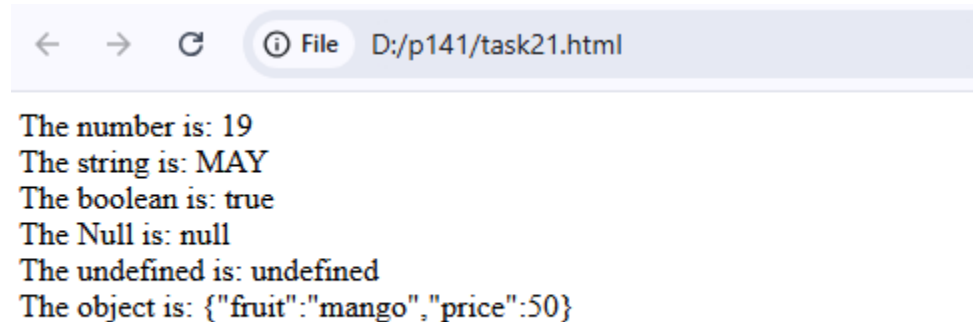
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
  <script>
    let number = 19;
    var month = "MAY";
    document.write("The number is: " + number + "<br>");
    document.write("The string is: " + month + "<br>");
    let boolean = true;
    document.write("The boolean is: " + boolean + "<br>");
    var mynull = null;
    document.write("The Null is: " + mynull + "<br>");
    let age;
    document.write("The undefined is: " + age + "<br>");
    let object = {
      fruit: 'mango',
      price: 50.0
    };
    document.write("The object is: " + object + "<br>");
  </script>
</body>
</html>

```

Output:



```

The number is: 19
The string is: MAY
The boolean is: true
The Null is: null
The undefined is: undefined
The object is: {"fruit": "mango", "price": 50}

```

Task 22:

```

<!DOCTYPE html>
<html lang="en">
<head>

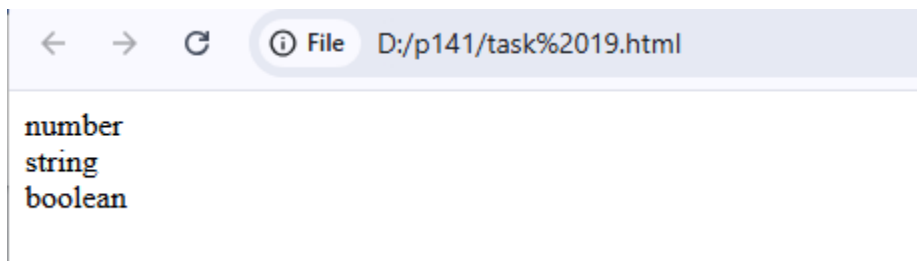
```

```

    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        let x=5;
        let y=x;
        document.write(typeof y+"<br>");
        var name="priya";
        let myname = name;
        document.write(typeof myname+"<br>");
        let age=true;
        let myage=age;
        document.write(typeof myage+"<br>");
    </script>
</body>
</html>

```

Output:



Task 23:

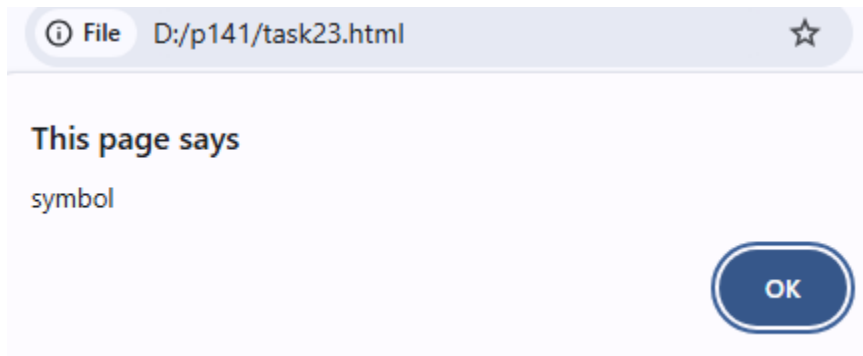
```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        let x = Symbol()

```

```
        alert(typeof x); // symbol
    </script>
</body>
</html>
```

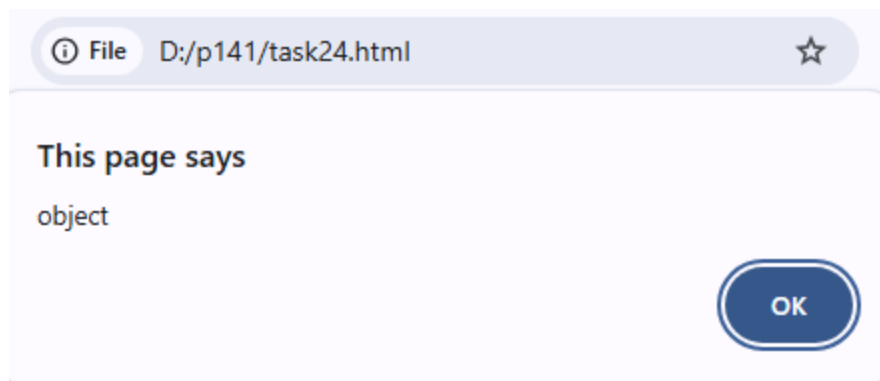
Output:



Task 24:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        var val=null;
        alert(typeof val);
    </script>
</body>
</html>
```

Output:



Task 25:

Var:

- Variables declared with `var` are **function-scoped** (if declared inside a function) or **globally-scoped** (if declared outside any function).
- If declared inside a block (like a loop or an if statement), the `var` variable is still accessible outside that block, within the function or globally, depending on where it was declared.

Code:

```
for (var i = 0; i < 3; i++) {  
  
    console.log(i);  
  
}  
  
console.log(i);
```

Let:

- Variables declared with `let` are **block-scoped**. This means they are only accessible within the block (i.e., the `{}`) where they are defined.

- Let respects the block boundaries such as in loops, if statements, etc., meaning they cannot be accessed outside the block in which they are defined.

Code:

```
function testLet() {  
  if (true) {  
    let y = 20; // let is block-scoped, so y is only accessible inside the  
    block  
  }  
  console.log(y); // ReferenceError: y is not defined  
}  
  
testLet();
```

Basic operators:

Task 26:

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-scale=1.0">  
  <title>Document</title>  
</head>  
<body>  
  <script>  
    console.log("implicit conversion"); //str to num  
    let str="1905";  
    let num=str*1;  
    console.log(num);
```

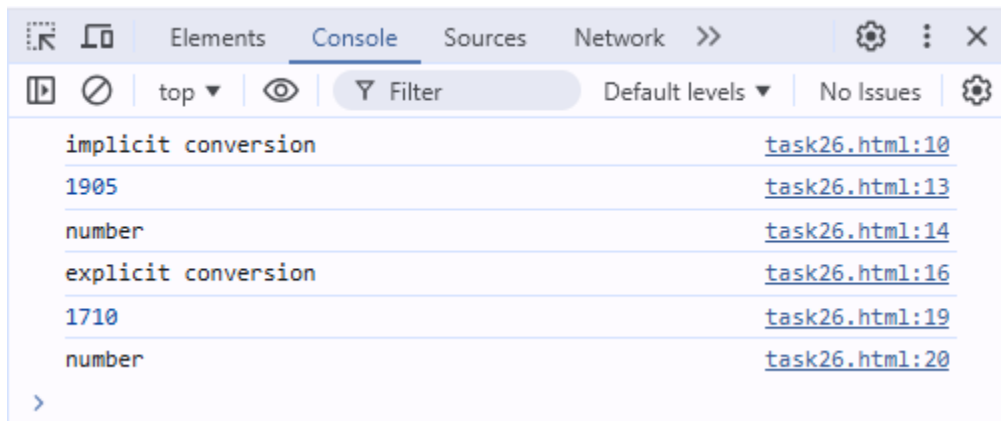
```

    console.log(typeof num);

    console.log("explicit conversion");
    let str1="1710";
    let num2=parseInt(str1);
    console.log(num2);
    console.log(typeof num2);
  </script>
</body>
</html>

```

Output:



Task 27:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    console.log("boolean to string");
    console.log("the boolean is true")
    let boolean =true;
    let val=String(boolean);
    console.log(val);
    console.log("the boolean is convert to:")
  </script>

```

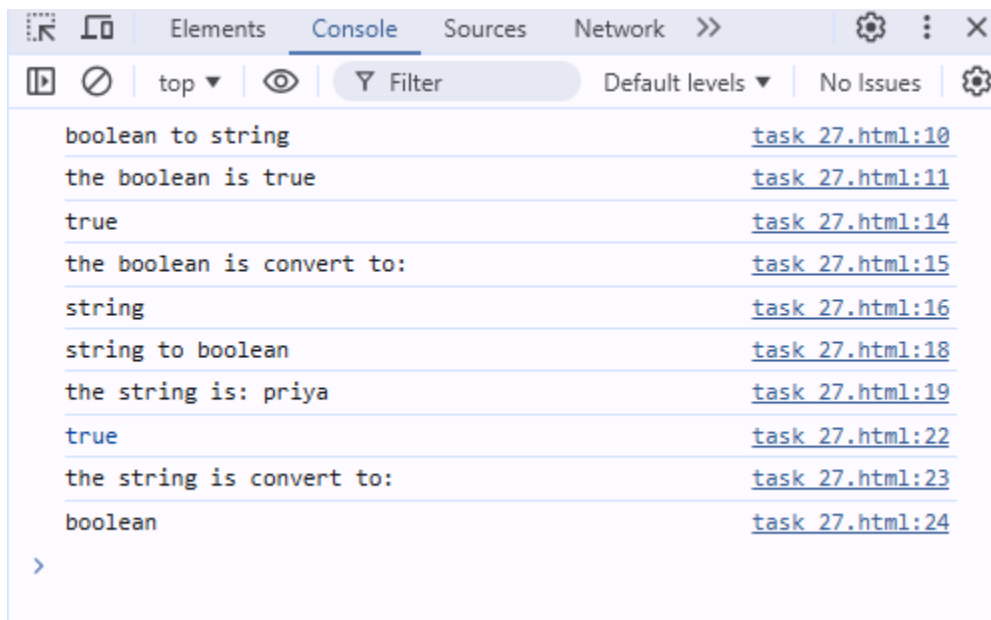
```

    console.log(typeof val);

    console.log("string to boolean");
    console.log("the string is: priya");
    let str="priya";
    let num=Boolean(str);
    console.log(num);
    console.log("the string is convert to:")
    console.log(typeof num);
  </script>
</body>
</html>

```

Output:



Task 28:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">

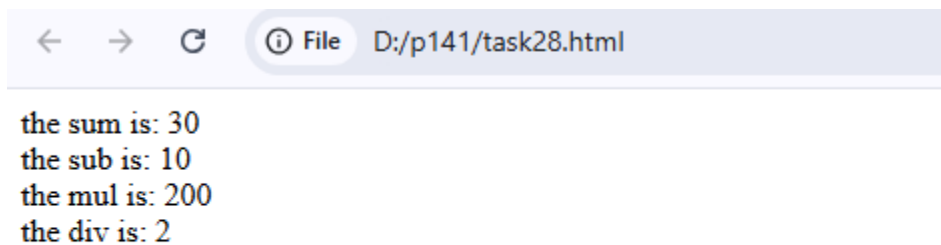
```

```

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        let p=20;
        let q=10;
        let sum=p+q;
        let sub=p-q;
        let mul=p*q;
        let div=p/q;
        document.write("the sum is: "+sum + "<br>");
        document.write("the sub is: "+sub + "<br>");
        document.write("the mul is: "+mul + "<br>");
        document.write("the div is: "+div + "<br>");
    </script>
</body>
</html>

```

Output:



Task 29:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>

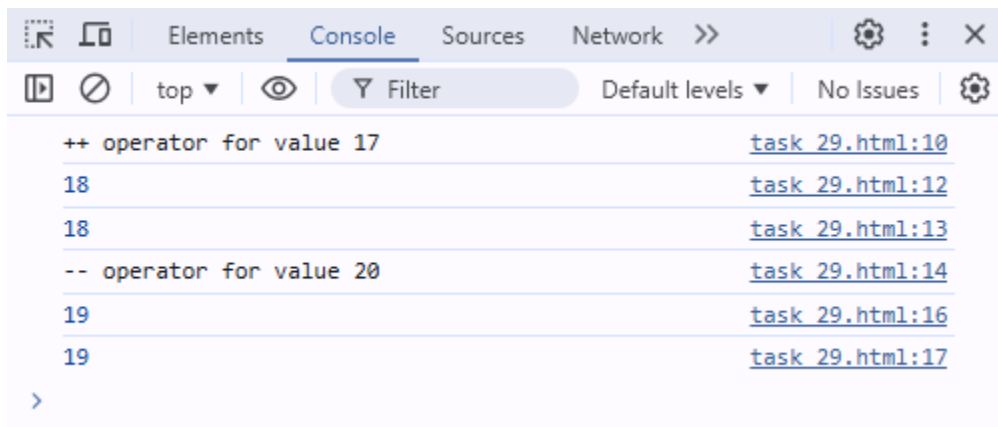
```

```

<script>
  console.log(++ operator for value 17");
  let s=17;
  console.log(++s);
  console.log(s++);
  console.log("-- operator for value 20");
  let p=20;
  console.log(--p);
  console.log(p--);
</script>
</body>
</html>

```

Output:



Task 30:

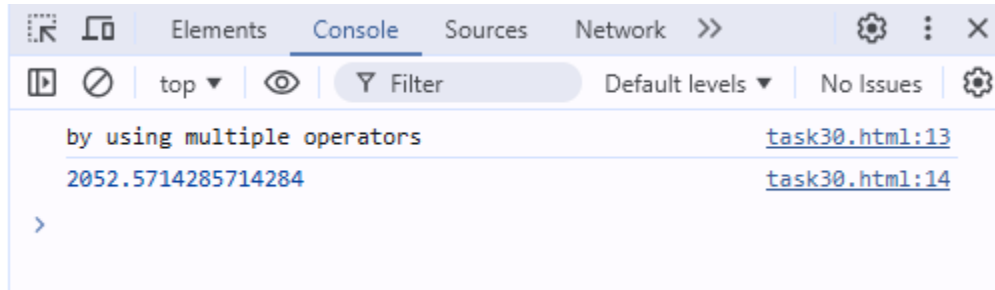
```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let a=90;
    let b=35;
    let ep=(a+b)+(a/b)-(b-a)*b;

```

```
        console.log("by using multiple operators");  
        console.log(ep);  
    </script>  
</body>  
</html>
```

Output:

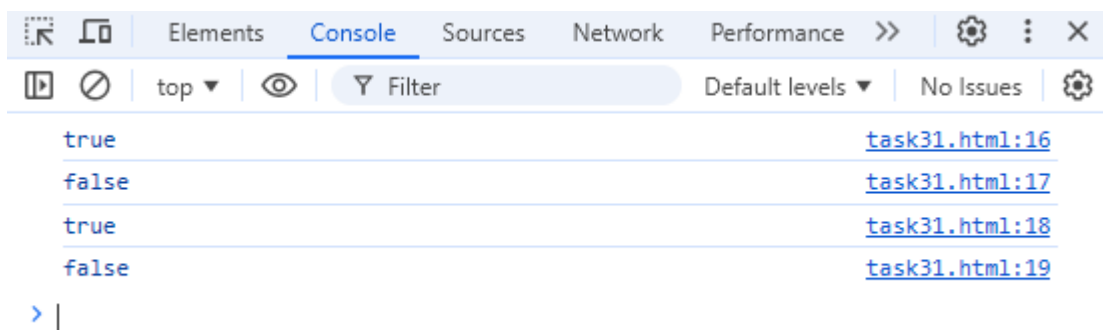


Comparisons:

Task 31:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let p=10;
    let q=5;
    let isGreater=p>q;
    let isSmaller=p<q;
    let isGreaterthanEqual=p>=q;
    let isSmallerthanEqual=p<=q;
    console.log(isGreater);
    console.log(isSmaller);
    console.log(isGreaterthanEqual);
    console.log(isSmallerthanEqual);
  </script>
</body>
</html>
```

Output:



Task 32:

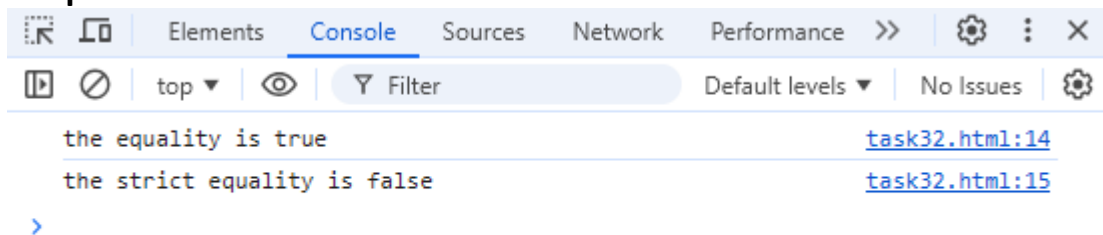
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
```

```

<body>
  <script>
    let p=10;
    var q="10";
    var equality=p==q;
    var strictEquality=p===q;
    console.log("the equality is "+equality);
    console.log("the strict equality is "+strictEquality);
  </script>
</body>
</html>

```

Output:



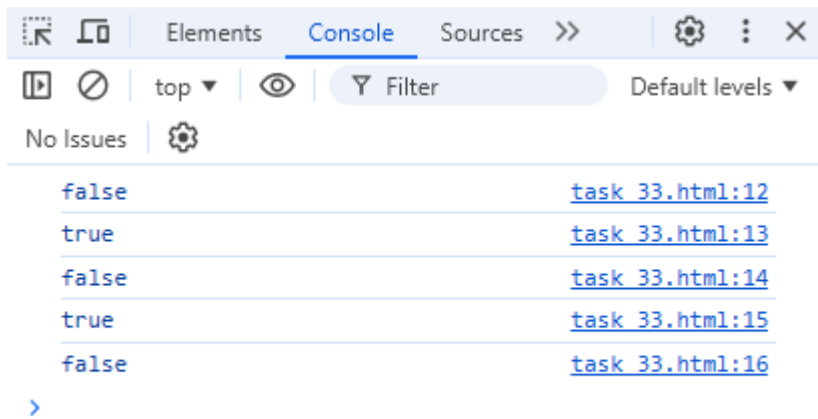
Task 33:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let str1='PRIYA';
    let str2='KOWSIK';
    console.log(str1<str2);
    console.log(str1>str2);
    console.log(str1<=str2);
    console.log(str1>=str2);
    console.log(str1==str2);
  </script>
</body>
</html>

```

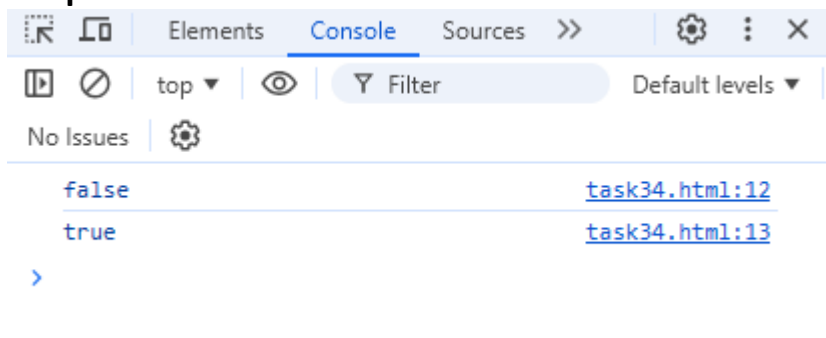
Output:



Task 34:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let p=19;
    var s='19';
    console.log(p!=s);
    console.log(p!==s);
  </script>
</body>
</html>
```

Output:



Task 35:

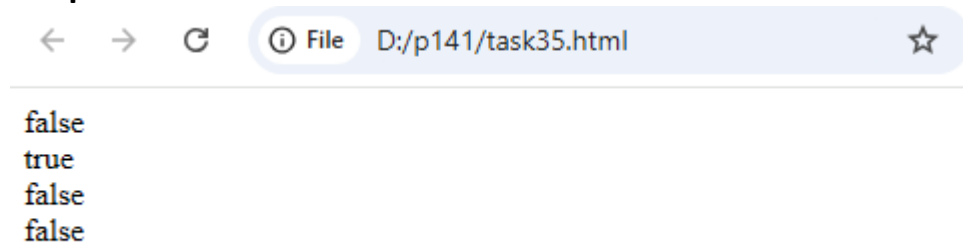
```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
```

```

    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        document.write((null=='')+ "<br>");
        document.write((null==undefined)+ "<br>");
        document.write((null===')+ "<br>");
        document.write((null===undefined)+ "<br>");
    </script>
</body>
</html>

```

Output:



Conditional branching: if, '?':

Task 36:

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
        document.write("check num is even or odd"+"<br>");
        let num=parseInt(prompt("enter the number"));
        if(num%2==0){
            document.write("the num is even"+"<br>");
        }else{
            document.write("the num is odd"+"<br>");
        }
    </script>
</body>
</html>

```

Output:

This page says

enter the number

OK

Cancel

← → ↻ ⓘ File D:/p141/task36.html

check num is even or odd
the num is even

Task 37:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    document.write("check the num is positive,negative,zero"+"<br>");
    let num = parseInt(prompt("enter a number"));
    if(num>0){
      document.write("The num is Positive"+"<br>");
    }if(num<0){
      document.write("The num is Negative"+"<br>");
    }else{
      document.write("The num is Zero"+"<br>");
    }
  </script>
</body>
</html>
```

Output:

This page says

enter a number

OK Cancel

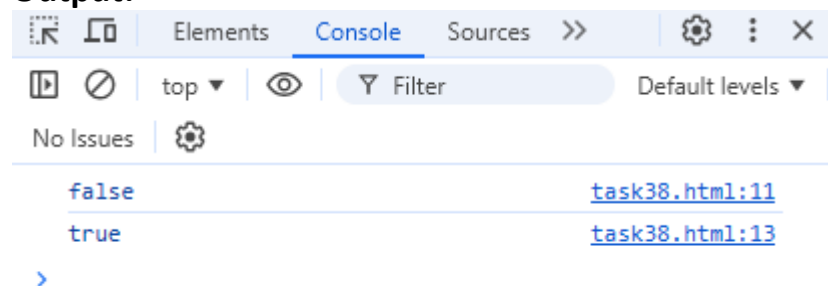
← → ↻ ⓘ File D:/p141/task37.html

check the num is positive,negative,zero
The num is Negative

Task 39:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let res=((6<3)?true:false);
    console.log(res);
    let res2=((19>17)?true:false);
    console.log(res2);
  </script>
</body>
</html>
```

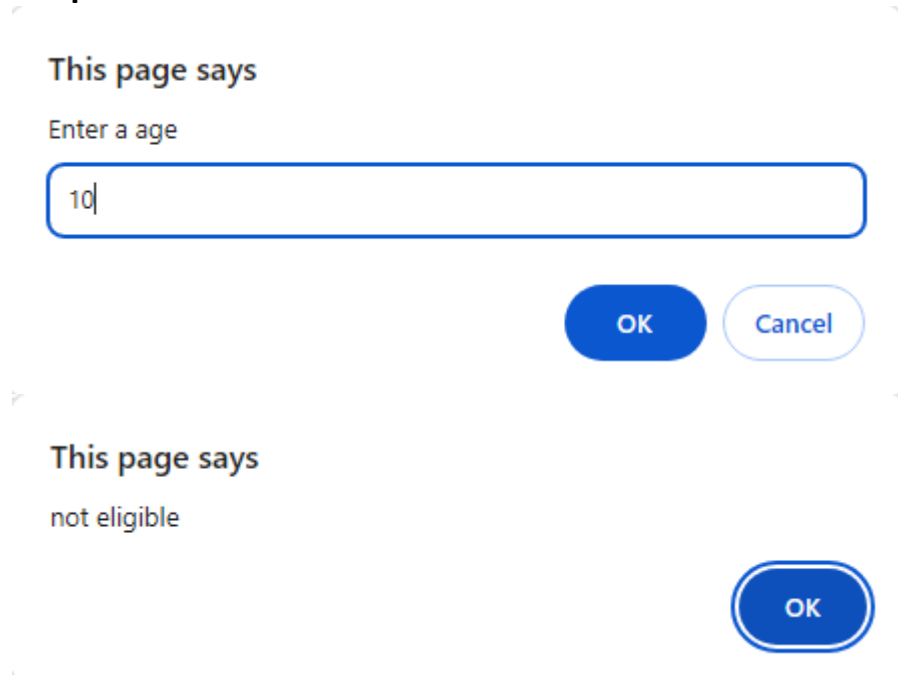
Output:



Task 38:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let age=prompt("Enter a age");
    if((age>=18)?true:false){
      document.write("Eligible to vote");
    }else{
      alert("not eligible")
    }
  </script>
</body>
</html>
```

Output:

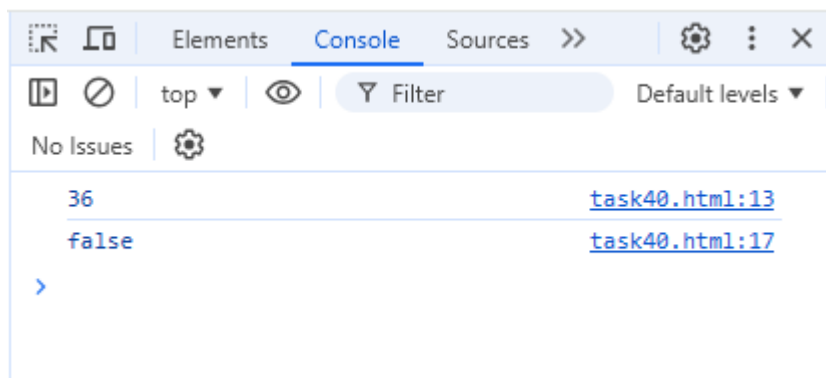


Task 40:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
```

```
<body>
  <script>
    let a=20;
    let b=16;
    let ans=((a>b)?a+b:a-b);
    console.log(ans);
    var dept1='cs';
    var dept2='ec';
    let res=((dept1==dept2)?true:false);
    console.log(res);
  </script>
</body>
</html>
```

Output:

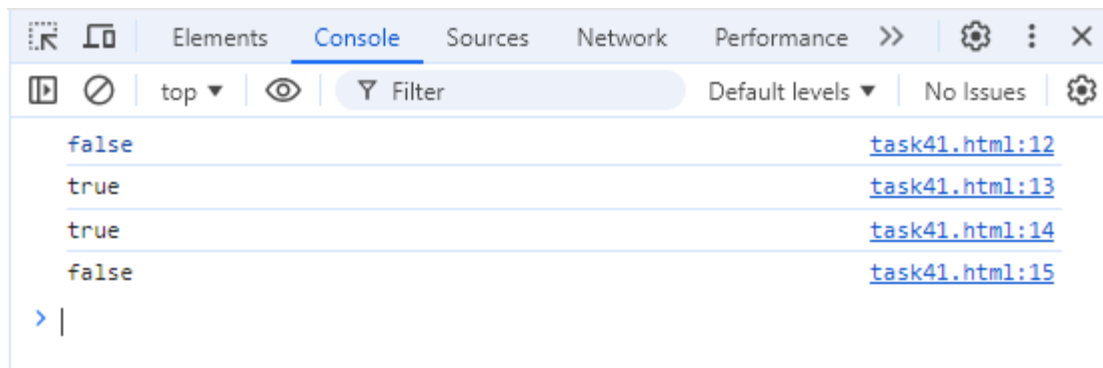


1. Logical operators:

Task 41:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let a = false;
      let b = true;
      console.log(a && b + "\n");
      console.log(a || b + "\n");
      console.log(!a + "\n");
      console.log(!b + "\n");
    </script>
  </body>
</html>
```

Output:



Task 42:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
  <meta charset="UTF-8" />
  <meta name="viewport" content="width=device-width, initial-scale=1.0" />
  <title>Document</title>
</head>
<body>
  <script>
    let p = 50;
    let s = 60;
    let num = parseInt(prompt("enter a number"));
    if (p <= num && s >= num) {
      document.write("The num is in range" + "<br>");
    } else {
      alert("The num is not in range");
    }
  </script>
</body>
</html>
```

Output:

The screenshot shows a web browser window. At the top, it says "This page says". Below that, there is a text input field with the placeholder text "enter a number". The input field contains the number "57". Below the input field, there are two buttons: "OK" and "Cancel". The "OK" button is highlighted in blue. At the bottom of the browser window, there is a navigation bar with back, forward, and refresh icons, followed by a "File" icon and the path "D:/p141/task42.html". Below the browser window, the text "The num is in range" is displayed.

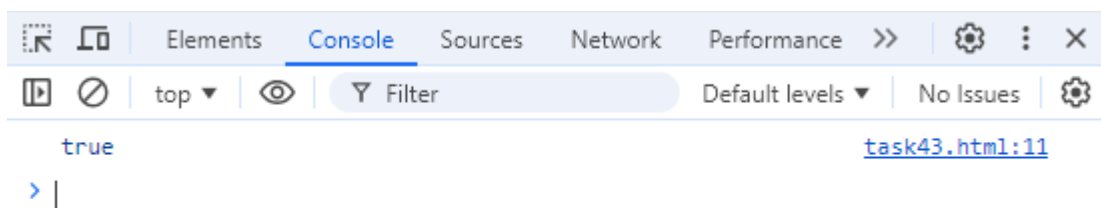
Task 43:


```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let boolean = false;
      console.log(!boolean);
    </script>
  </body>
</html>

```

Output:



Task 44:

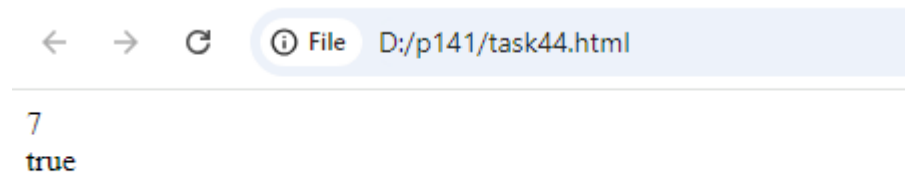
```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let p = 5;
      let s = 7;
      let res = p && s;
      document.write(res + "<br>");
      var op1 = "true";
      var op2 = "false";
    </script>
  </body>
</html>

```

```
    let ans = op1 || op2;
    document.write(ans);
  </script>
</body>
</html>
```

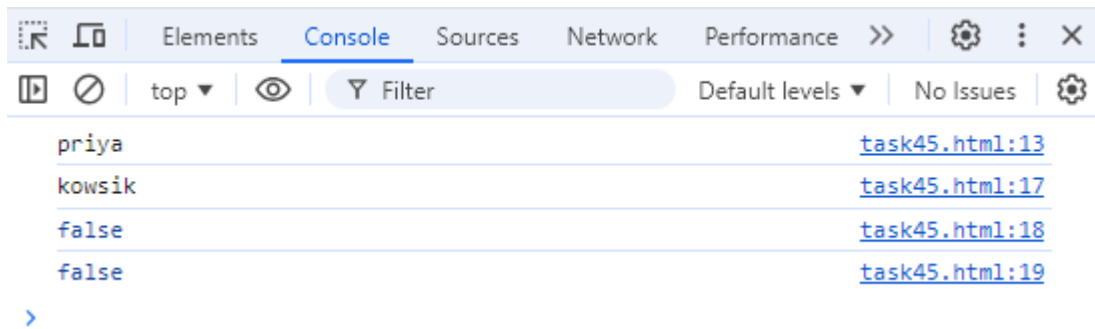
Output:



Task 45:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let str1 = "mathi";
      let str2 = "priya";
      let res = str1 && str2;
      console.log(res);
      let x = 0;
      let y = "kowsik";
      let res2 = x || y;
      console.log(res2);
      console.log(!str1);
      console.log(!str2);
    </script>
  </body>
</html>
```

Output:

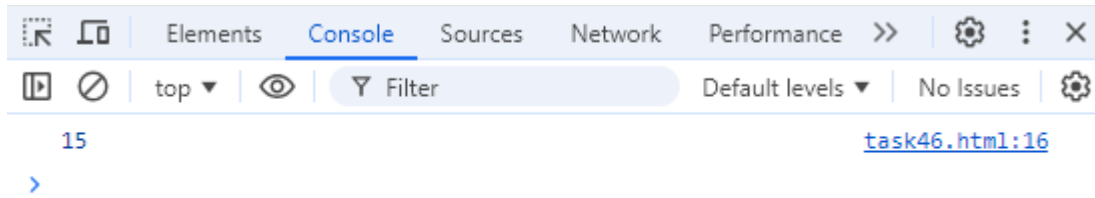


2. Functions:

Task 46:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let num1 = 10;
      let num2 = 5;
      function Sum(num1, num2) {
        return num1 + num2;
      }
      let res = Sum(num1, num2);
      console.log(res);
    </script>
  </body>
</html>
```

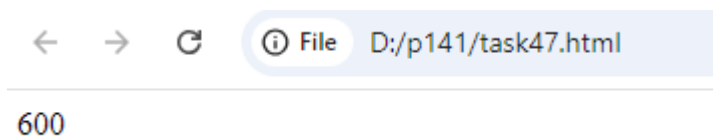
Output:



Task 47:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let length = 20;
      let breadth = 30;
      function rectangle() {
        return length * breadth;
      }
      let res = rectangle();
      document.write(res);
    </script>
  </body>
</html>
```

Output:



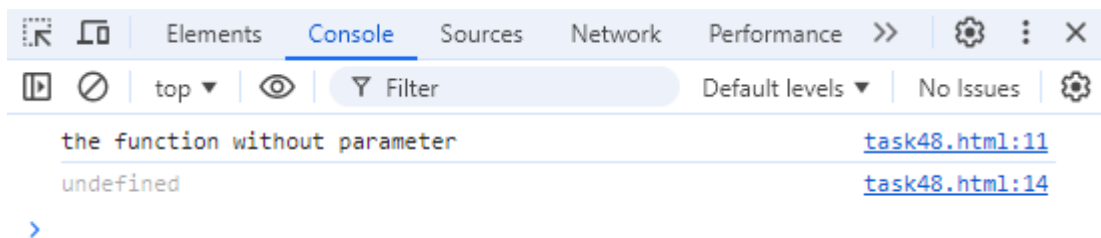
Task 48:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function mul() {
        console.log("the function without parameter");
      }
      let res = mul();
      console.log(res);
    </script>
  </body>
</html>

```

Output:



Task 49:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      function def() {
        document.write("The function that returns nothing "+"<br>")
      }
    </script>
  </body>
</html>

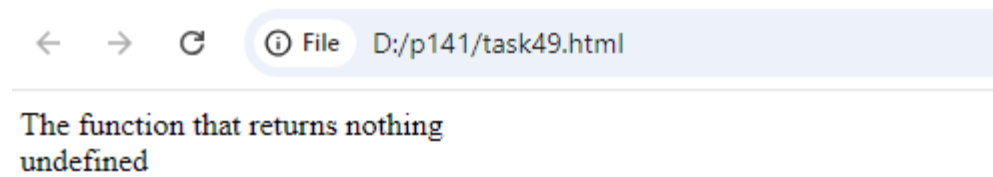
```

```

);
    }
    let res = def();
    document.write(res);
</script>
</body>
</html>

```

Output:



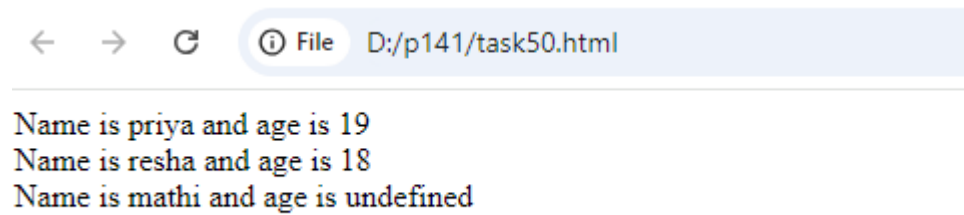
Task 50:

```

<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let name = "priya";
      let age = 19;
      function fun(name, age) {
        document.write(`Name is ${name} and age is ${age}` + "<br>");
      }
      fun(name, age);
      fun("resha", 18);
      fun("mathi");
    </script>
  </body>
</html>

```

Output:

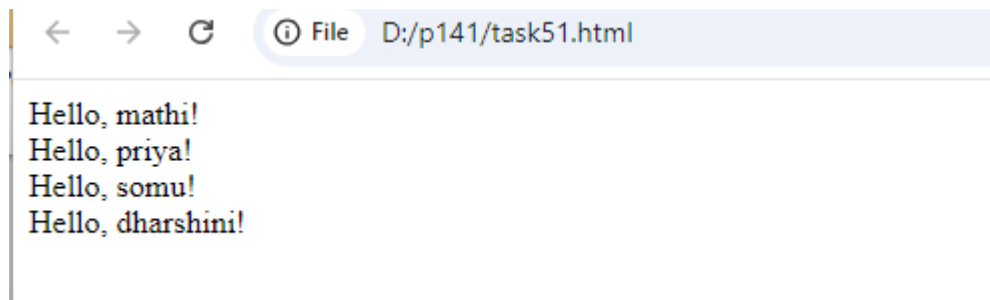


3. Arrow Functions:

Task 51:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      let greet = (name) => {
        return `Hello, ${name}!`;
      };
      document.write(greet("mathi") + "<br>");
      document.write(greet("priya") + "<br>");
      document.write(greet("somu") + "<br>");
      document.write(greet("dharshini") + "<br>");
    </script>
  </body>
</html>
```

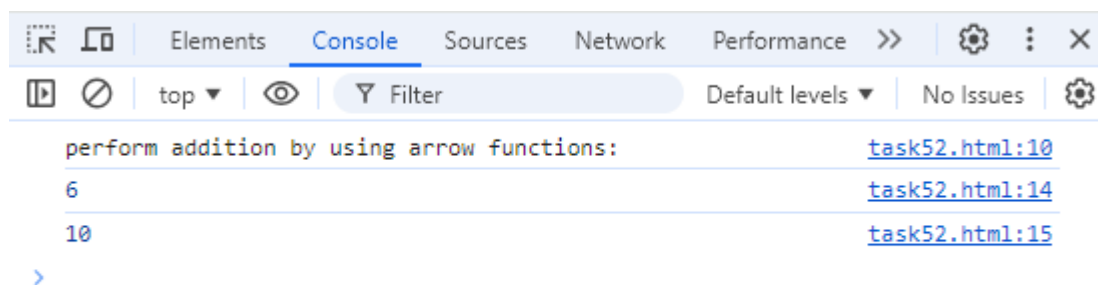
Output:



Task 52:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      console.log("perform addition by using arrow functions:");
      let add = (num1, num2) => {
        return num1 + num2;
      };
      console.log(add(4, 2));
      console.log(add(5, 5));
    </script>
  </body>
</html>
```

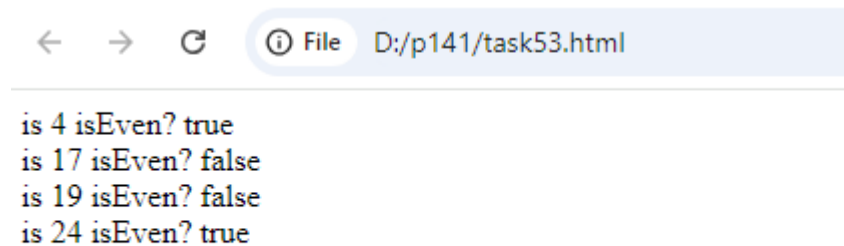
Output:



Task 53:

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Document</title>
  </head>
  <body>
    <script>
      console.log("Check the num is by using arrow functions:");
      let a;
      const isEven = (a) => a % 2 == 0;
      document.write(`is 4 isEven? ${isEven(4)} <br>`);
      document.write(`is 17 isEven? ${isEven(17)} <br>`);
      document.write(`is 19 isEven? ${isEven(19)} <br>`);
      document.write(`is 24 isEven? ${isEven(24)} <br>`);
    </script>
  </body>
</html>
```

Output:



is 4 isEven? true
is 17 isEven? false
is 19 isEven? false
is 24 isEven? true

Task 54:

```
<!DOCTYPE html>

<html lang="en">
<head>
```

```

<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<title>Document</title>
</head>
<body>
  <script>
    document.write("to fnd a maximum number" + "<br>"
);
    let num1=parseInt(prompt("Enter a num1"));
    let num2=parseInt(prompt("Enter a num2"));
    let maxValue = (num1, num2) => {
      if(num1>num2){
        return `${num1} is bigger`;
      }else{
        return num2;
      }
    }
    document.writeln(maxValue(num1,num2));

  </script>
</body>
</html>

```

Output:

This page says

Enter a num1

20

OK Cancel

This page says

Enter a num2

10

OK Cancel

← → ↻

File D:/p141/task54.html

to fnd a maximum number

20 is bigger

Task 55:

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Document</title>
</head>
<body>
  <script>
    let multiplyTraditional;
    let multiplyArrow;
    const myObject={
      value:10,
      multiplyTraditional:function(number){
        console.log('Traditional function',this);
        return this.value*number;
      },
      multiplyArrow:(number)=>{
        console.log('arrow function',this);
        return this.value*number;
      },
    };
    console.log(myObject.multiplyTraditional(4));
    console.log(myObject.multiplyArrow(4));
  </script>
</body>
</html>
```

Output:

