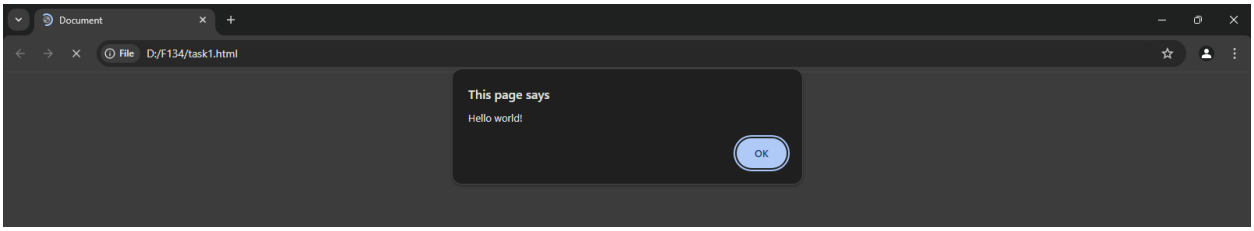
Task 1: Write a simple script that displays “Hello, World!” on the web page using an alert box.

```
<!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>
        alert("Hello world!");
    </script>
</body>
</html>
```

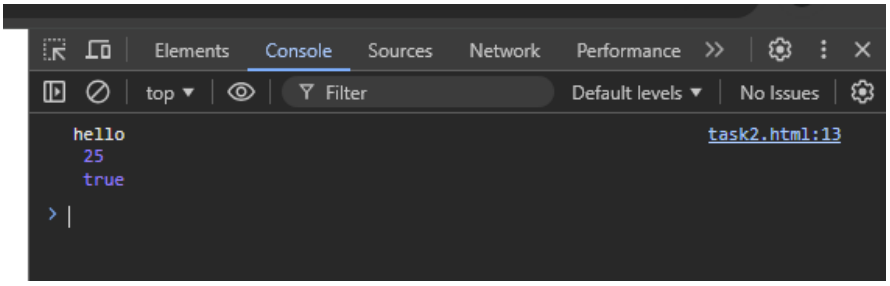
Output:



Task 2: Experiment with different data types in JavaScript (e.g., string, number, boolean) by declaring and logging them in the console.

```
<!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 a="hello";
        var b=25;
        var c=true;
        console.log(a,"\n",b,"\n",c);
    </script>
</body>
</html>
```

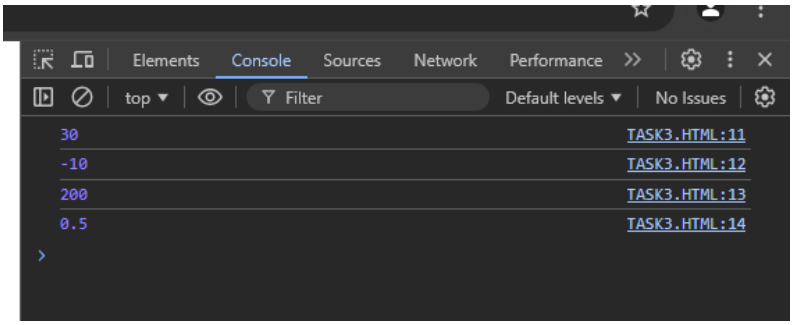
Output:



Task 3: Use the console to perform basic math operations like addition, subtraction, multiplication, and division.

```
<!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 a=10,b=20;
        console.log(a+b);
        console.log(a-b);
        console.log(a*b);
        console.log(a/b);

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

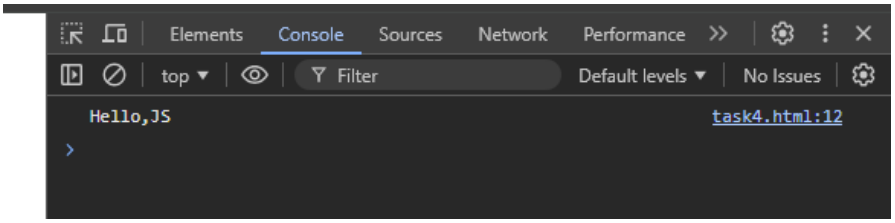
Output:



Task 4: Declare two strings and concatenate them using the + operator.

```
<!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 n="Hello,";
    var m="JS";
    console.log(n+m);
</script>
    </body>
</html>
```

Output:

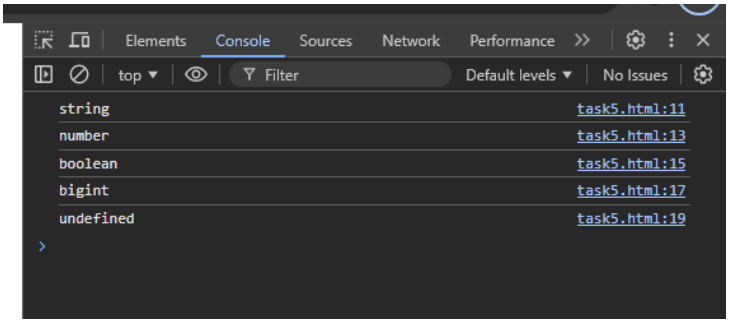


Task 5: Use the typeof operator to check the data type of various variables.

```
<!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 a="One";
        console.log(typeof a);
        var b=12;
        console.log(typeof b);
        var c=true;
        console.log(typeof c);
        var d=2345674564444447890n;
        console.log(typeof d);
        var e;
```

```
        console.log(typeof e);
    </script>
</body>
</html>
```

Output:



Task 6: Write a multi-line JavaScript comment and a single-line comment. Explain the difference.

```
<!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>
        //This is single line comment ,used for commenting one line

        /*This is multi line comment ,used for commenting
        two or more lines*/
    </script>
</body>
</html>
```

Task 7: Create a script with both semicolon-separated and not separated lines. Note any differences in behavior.

```
<!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>
        // Using semicolons to separate multiple statements on one line
let x = 5; let y = 10; let z = x + y; console.log("Sum:", z);

// Non-separated lines (standard behavior)
let a = 20;
let b = 25;
let c = a + b;
console.log("Sum:", c);
     </script>
</body>
</html>
```

- **Semicolon-Separated Lines**: Multiple statements on a single line, separated by semicolons, works but can reduce readability and clarity, especially in longer or more complex code.

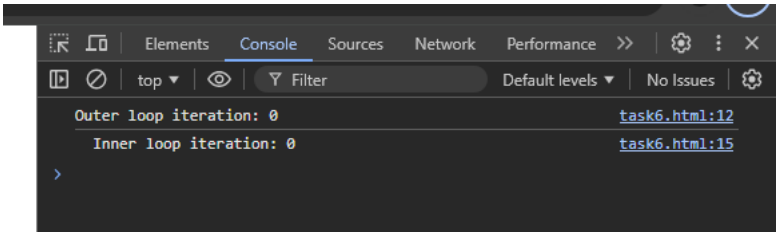
- **Non-Separated Lines**: Each statement on its own line makes the code more readable, easier to debug, and more maintainable, which is why it's the recommended approach in JavaScript.

Task 8: Use proper indentation to format a nested loop

```
<!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>
        // Outer loop
for (let i = 0; i < 1; i++) {
    console.log("Outer loop iteration: " + i);
    // Inner loop
    for (let j = 0; j < 1; j++) {
        console.log("  Inner loop iteration: " + j);
    }
}
    </script>
</body>
```

```
</html>
```

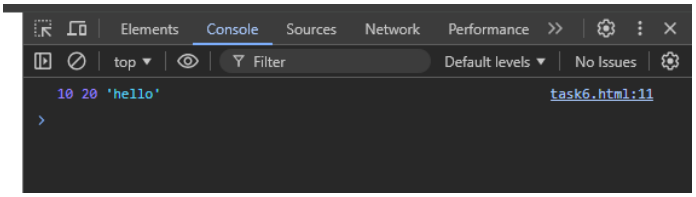
Output:



Task 9: Declare multiple variables in a single line.

```
<!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 n=10,m=20,name="hello";
       console.log(n,m,name);
    </script>
</body>
</html>
```

Output:

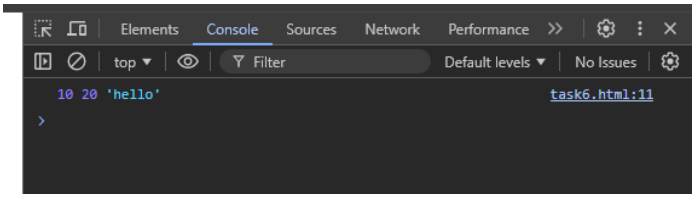


Task 10: Place a script tag at the top and bottom of an HTML document. Note any differences in behavior.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script>
        var n=10,m=20,name="hello";
        console.log(n,m,name);
```

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

Output:



Task 16: Declare variables using let, const, and var. Discuss when each should be used.

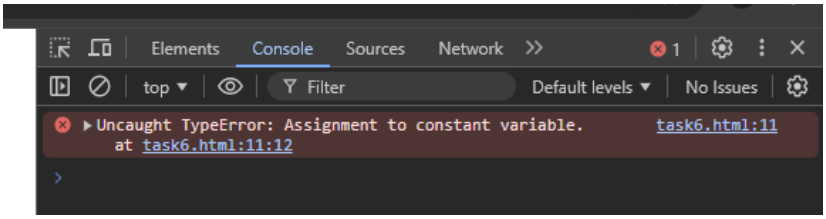
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
          if (true) {
        var name = "Alice";
        var name="john";
    }
    console.log(name); /*"Alice" (because `var` is function-scoped, it is
accessible outside the `if` block),
    the variable can be re-declared and re-initialized*/

    let age = 25;
console.log(age); // 25

age = 30; // Reassigning the value can be possible and it is a block scope
console.log(age); // 30

const birthYear = 1996;
console.log(birthYear); // 1996
```

```
// Reassigning will result in an error
// birthYear = 2000;
// Uncaught TypeError: Assignment to constant variable.
     </script>
</body>
</html>
```

Task 17: Attempt to reassign a const variable and observe the result.

```
<!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 b=10;
          b=20;
          console.log(b);
     </script>
</body>
</html>
```

Output:

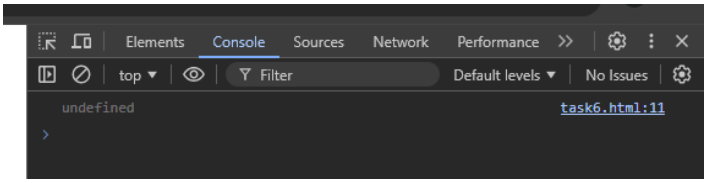


Task 18: Declare a variable without initializing it and print its value.

```
<!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 n;
          console.log(n);
     </script>
</body>
</html>
```

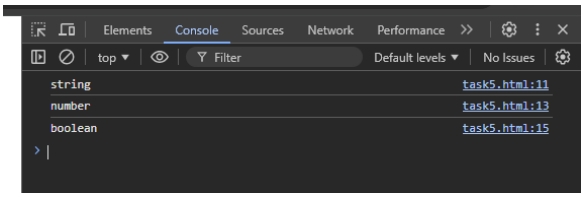
Output:



Task 19: Assign a number, string, and boolean value to a variable and print its type using typeof.

```
<!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 a="One";
        console.log(typeof a);
        var b=12;
        console.log(typeof b);
        var c=true;
        console.log(typeof c);
    </script>
</body>
</html>
```

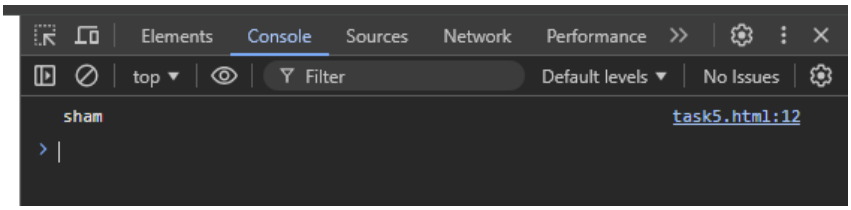
Output:



Task 20: Rename a variable and observe the outcome.

```
<!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";
        var name="sham";
        console.log(name);
    </script>
</body>
</html>
```

Output:



Task 21: Create variables of different data types (e.g., string, number, boolean, null, undefined, object).

```
<!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>
 // 1. String (Text data type)
let name = "John Doe";
console.log(typeof name); // Output: "string"
console.log(name);        // Output: "John Doe"

// 2. Number (Numeric data type)
let age = 30;
console.log(typeof age);  // Output: "number"
```

```
console.log(age);         // Output: 30

// 3. Boolean (True/False values)
let isActive = true;
console.log(typeof isActive); // Output: "boolean"
console.log(isActive);       // Output: true

// 4. Undefined (A variable that has been declared but not assigned a value)
let greeting;
console.log(typeof greeting); // Output: "undefined"
console.log(greeting);        // Output: undefined

// 5. Object (A collection of key-value pairs)
let person = {
  name: "Alice",
  age: 25,
  isEmployed: true
};
console.log(typeof person);  // Output: "object"
console.log(person);          // Output: { name: "Alice", age: 25, isEmployed:
true }
    </script>
</body>
</html>
```

Task 22: Use the typeof operator to determine the type of various variables.

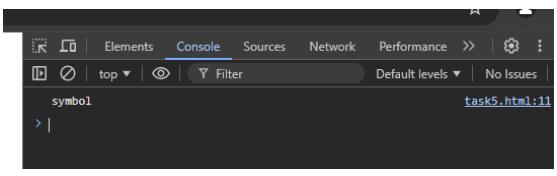
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
</head>
<body>
    <script>
 // 1. String (Text data type)
let name = "John Doe";
console.log(typeof name); // Output: "string"

// 2. Number (Numeric data type)
let age = 30;
console.log(typeof age);  // Output: "number"
```

```
// 3. Boolean (True/False values)
let isActive = true;
console.log(typeof isActive); // Output: "boolean"

// 4. Undefined (A variable that has been declared but not assigned a value)
let greeting;
console.log(typeof greeting); // Output: "undefined"

// 5. Object (A collection of key-value pairs)
let person = {
  name: "Alice",
  age: 25,
  isEmployed: true
};
console.log(typeof person);  // Output: "object"
    </script>
</body>
</html>
```

Task 23: Declare a symbol and print its type

```
<!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 uniqueSymbol = Symbol("description");
console.log(typeof uniqueSymbol);
    </script>
</body>
</html>
```

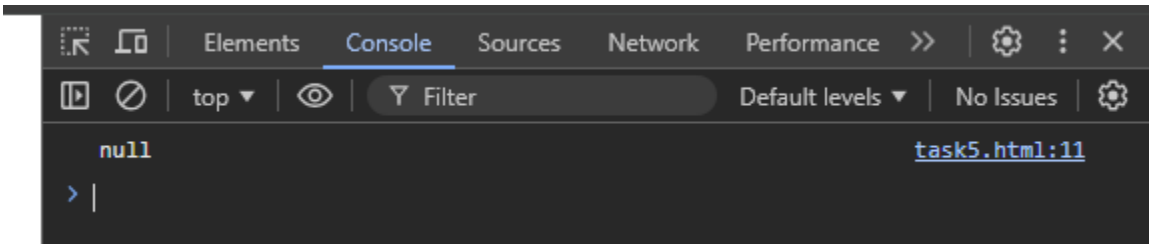
Output:



Task 24: Assign the value null to a variable and check its type using typeof.

```
<!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=null;
console.log(name);
    </script>
</body>
</html>
```

Output:



Task 25: Differentiate between declaring a variable using var and let in terms of scope.

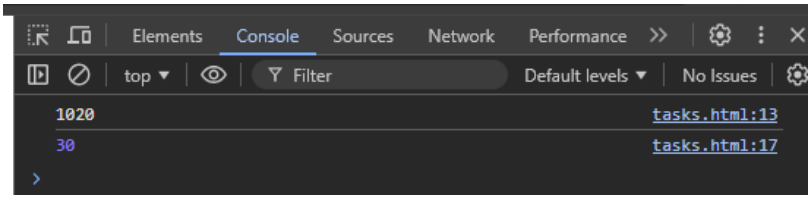
- **var** is **function-scoped**, meaning it is accessible within the entire function where it's declared, even if declared inside blocks like loops or conditionals.
- **let** is **block-scoped**, meaning it is only accessible within the specific block (e.g., `{}`) where it is declared, providing better control over variable visibility.

Task 26: Convert a string to a number using both implicit and explicit conversion.

```
<!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>
        //implicit type conversion
        var a=10;
        var b="20";
        console.log(a+b);
        //explicit type conversion
        var a=10;
        var b=Number("20");
        console.log(a+b);
    </script>
</body>
</html>
```

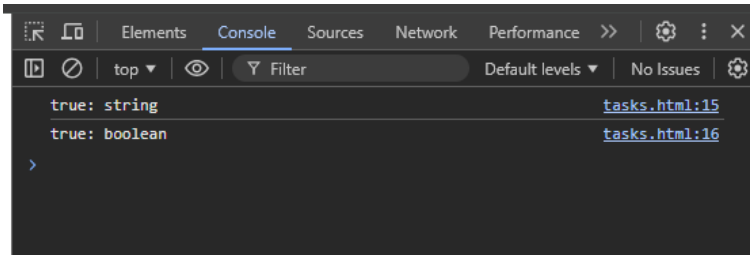
Output:



Task 27: Convert a boolean to a string and vice versa.

```
<!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>
       //bollean to string
       var bool=true;
       var h="hello";
       var res=String(bool);
       var res1=Boolean(h);
       console.log(res+": "+typeof res);
       console.log(res1+": "+typeof res1);
    </script>
</body>
</html>
```

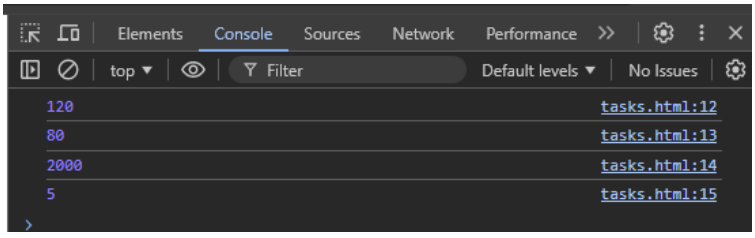
Output:



Task 28: Practice basic arithmetic operators (+, -, *, /, %)

```
<!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 a=100;
      var b=20;
      console.log(a+b);
      console.log(a-b);
      console.log(a*b);
      console.log(a/b);
    </script>
</body>
</html>
```

Output:

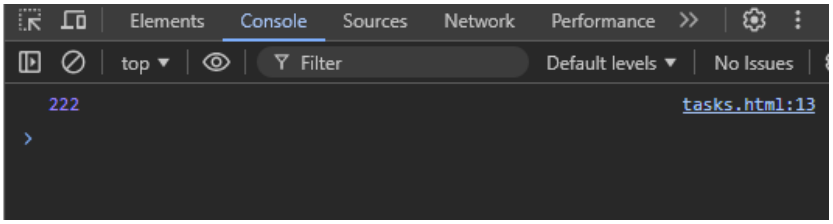


Task 29: Use the ++ and -- operators on a numeric variable.

```
<!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 a=100;
      var b=20;
      var res=a++ + b-- + ++a;
      console.log(res);
    </script>
</body>
</html>
```

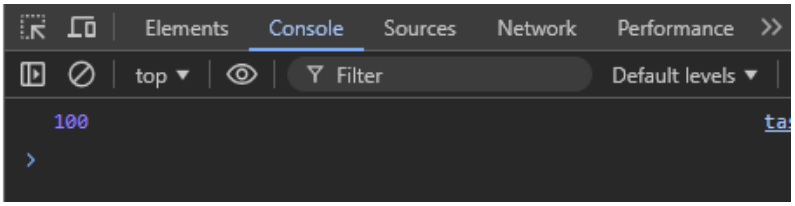
Output:



Task 30: Explore the precedence of operators by combining multiple operators in a single expression.

```
<!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 a=100;
      var b=20;
      var res=a+b-a*b/a;
      console.log(res);
    </script>
</body>
</html>
```

Output:

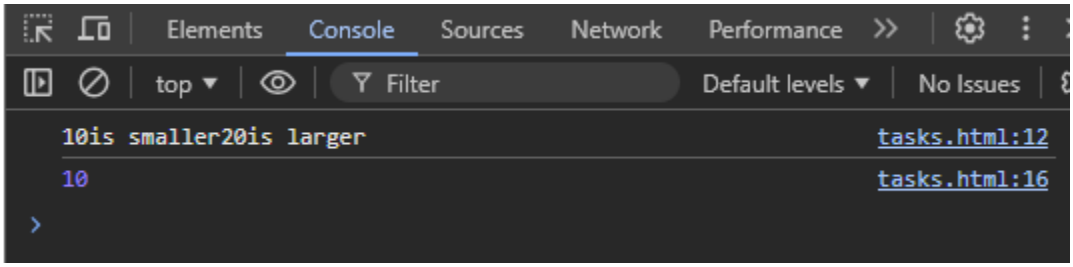


Comparisons, Conditional branching: if, '?'

Task 31: Compare two numbers using relational operators (>, =, <=).

```
<!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 a=10,b=20;
     if(a<=b)
     console.log(a +"is smaller"+b +"is larger");
    else if(a>=b)
    console.log(b +"is smaller and"+ a + "is larger");
    var res=a<b?a:b;
    console.log(res);
    </script>
</body>
</html>
```

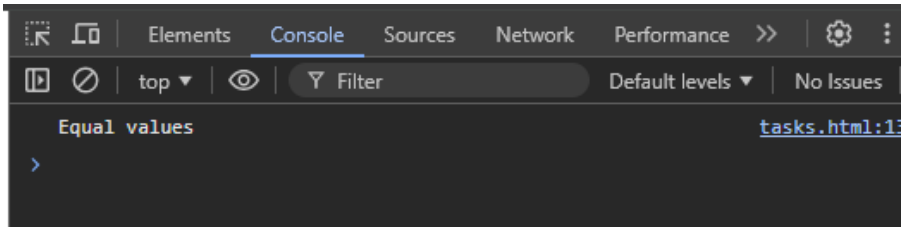
Output:



Task 32: Use equality () and strict equality (=) operators to compare different data types and note the differences.

```
<!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 a=10;
        var b="10";
    if(a==b)//compares values only
    console.log("Equal values");
    a=10;
    b=10;
   if(a===b)//compares values and types
   console.log("equal in types");
    </script>
</body>
</html>
```

Output:

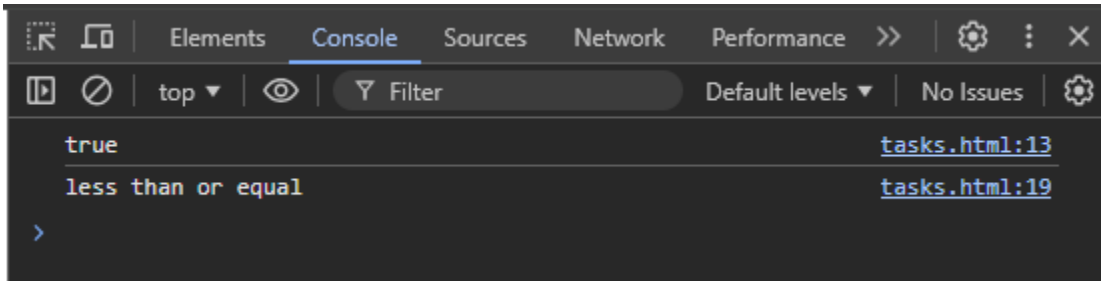


Task 33: Compare two strings lexicographically.

```
<!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 str="Apple";
        var str2="apple";
```

```
        if(str<str2)
        console.log("true");
    if(str==str2)
    console.log("both are equal");
   str="apple";
   str2="apple";
   if(str<=str2)
   console.log("less than or equal ");
    </script>
</body>
</html>
```

Output:

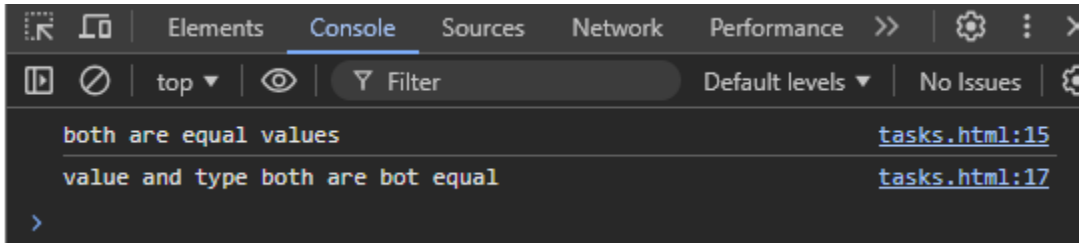


Task 34: Use the inequality (!=) and strict inequality (!==) operators to compare values.

```
<!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 n=10;
        var m="10";
        if(n!=m)
        console.log("value is not equal");
        else
        console.log("both are equal values");
    if(n!==m)
    console.log("value and type both are bot equal");
    </script>
</body>
```

```
</html>
```

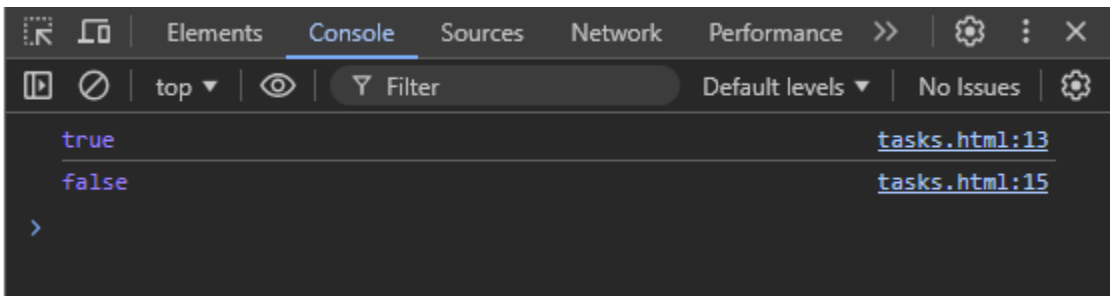
Output:



Task 35: Compare null and undefined using both == and ===

```
<!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 n=null;
        var m;
        var res=(n==m)
        console.log(res);
        res=(n===m)
        console.log(res);
    </script>
</body>
</html>
```

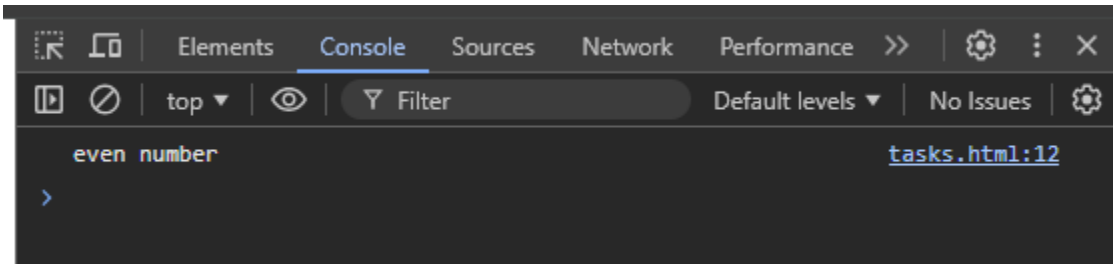
Output:



Task 36: Write an if statement that checks if a number is even or odd.

```
<!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 n=10;
       if(n%2==0)
       console.log("even number");
       else
       console.log("odd number");
    </script>
</body>
</html>
```

Output:

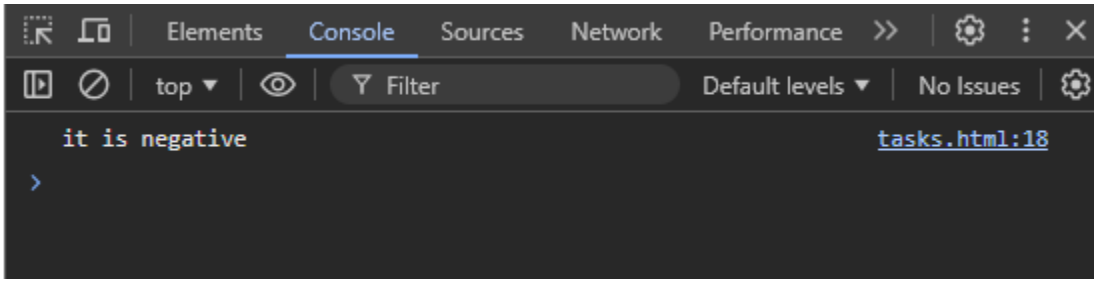


Task 37: Use nested if statements to classify a number as negative, positive, or zero.

```
<!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 m=-5;
      if(m>=0){
```

```
        if(m==0)
        console.log("it is zero");
    else
    console.log("it is positive")
      }
      else
      console.log("it is negative");
    </script>
</body>
</html>
```

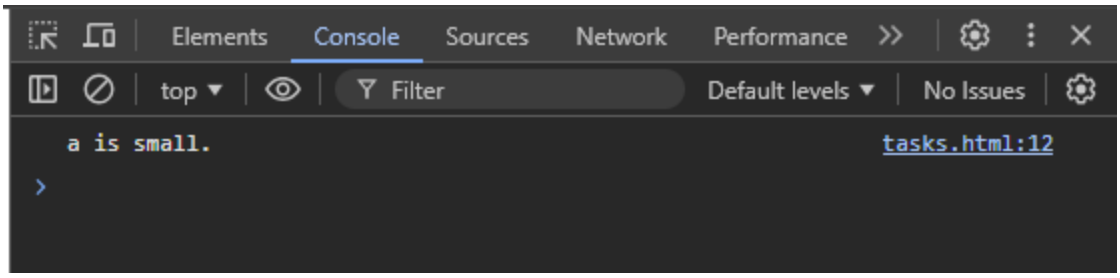
Output:



Task 38: Use the conditional (ternary) operator '?' to rewrite a simple if…else statement.

```
<!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 a=10,b=20;
      var res=a<b?"a is small.":"b is small";
      console.log(res);
    </script>
</body>
</html>
```

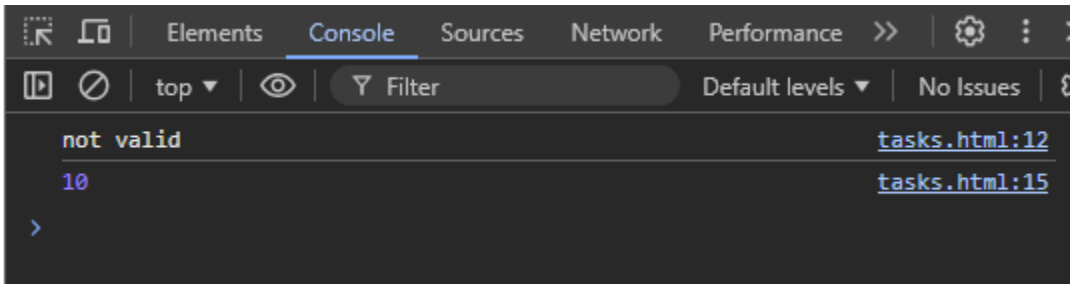
Output:



Task 39: Check the validity of a variable using the ? operator

```
<!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 a;
      var res=(a??"not valid");
      console.log(res);
      var b=10;
      var res=(b??"not valid");
      console.log(res);
    </script>
</body>
</html>
```

Output:

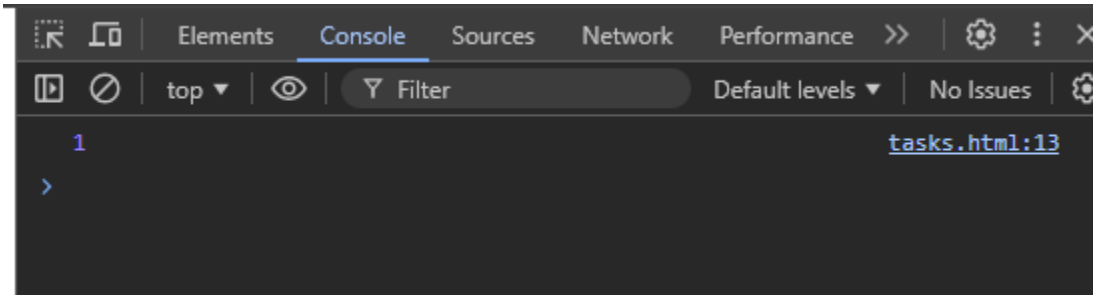


Task 40: Use the conditional operator to assign a value to a variable based on a condition.

```
<!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 a=10;
      var b;
      var res=a==10?b=1:b=0;
      console.log(res);
    </script>
</body>
</html>
```

Output:

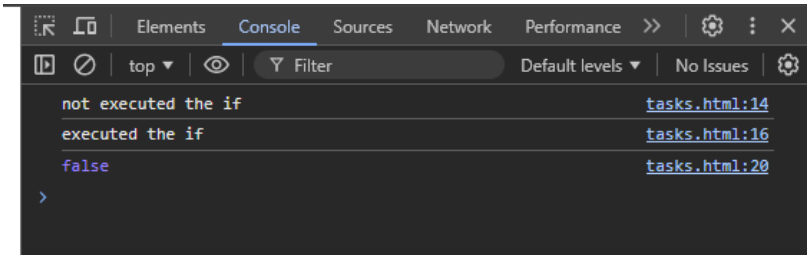


Task 41: Evaluate various combinations of logical operators (&&, ||, !).

```
<!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 a=10,b=20;
      if(a!=0&&a>b)
      console.log("executed the if");
    else
    console.log("not executed the if");
    if(a==10||b>a)
      console.log("executed the if");
    else
```

```
    console.log("not executed the if");
var res=a%2==0?true:false;
console.log(!res);
    </script>
</body>
</html>
```

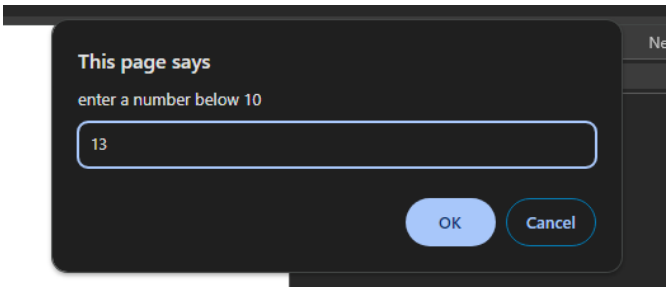
Output:



Task 42: Use logical operators to write a condition that checks if a number is in a given range

```
<!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 inp=prompt("enter a number below 10");
        var res=inp>0&&inp<=10?true:false;
       alert(res);
    </script>
</body>
</html>
```

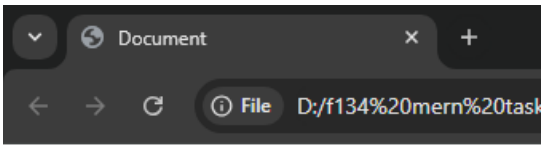
Output:



Task 43: Use the NOT (!) operator to invert a boolean value.

```
<!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 n=true;
        document.writeln("The given value before inverting: "+n+"<br>");
        document.write("The value after inverting: "+!n);
    </script>
</body>
</html>
```

Output:

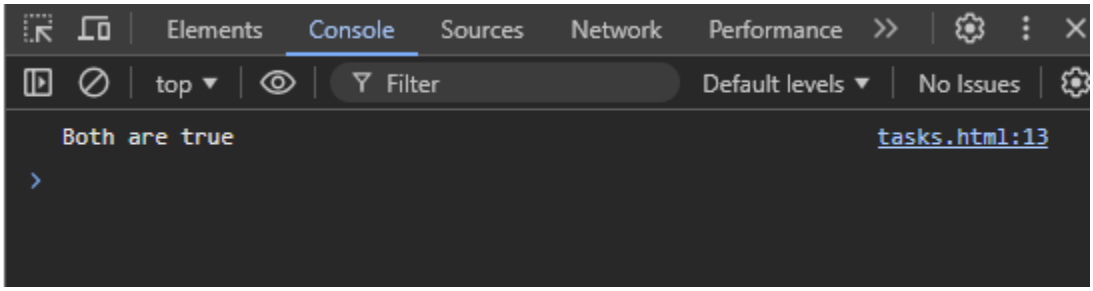


The given value before inverting: true
The value after inverting: false

Task 44: Evaluate the short-circuiting nature of logical operators.

```
<!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 a=true;
       var b=true;
       if(a&&b)
       console.log("Both are true");
    else
    console.log("any on econdition failed");
    </script>
</body>
</html>
```

Output:

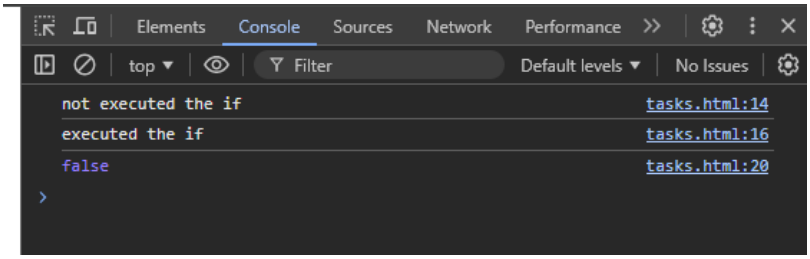


Task 45: Compare two non-boolean values using logical operators and observe the result.

```
<!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 a=10,b=20;
      if(a!=0&&a>b)
      console.log("executed the if");
    else
    console.log("not executed the if");
    if(a==10||b>a)
      console.log("executed the if");
    else
    console.log("not executed the if");
var res=a%2==0?true:false;
console.log(!res);
    </script>
</body>
</html>
```

Output:

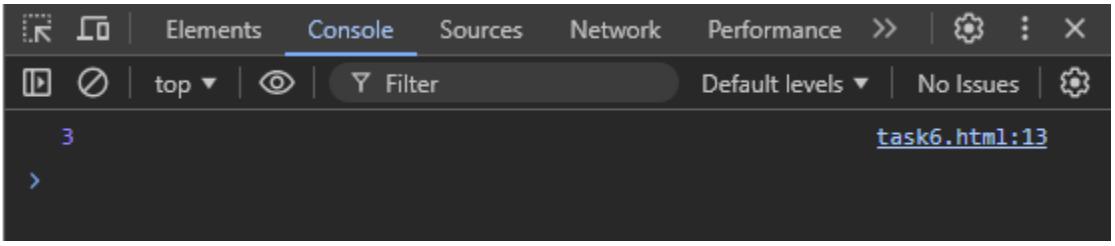


Task 46: Write a function that takes two numbers as arguments and returns their sum.

```
<!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 sum(a,b){
        return a+b;
       }
       console.log(sum(1,2));
     </script>
</body>
```

```
</html>
```

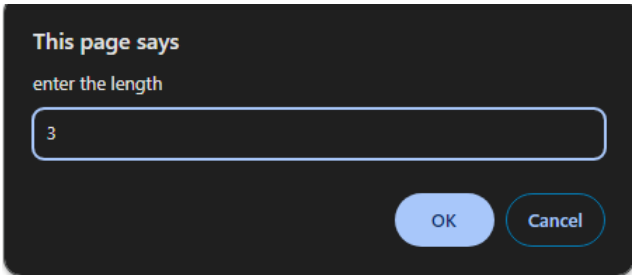
Output:

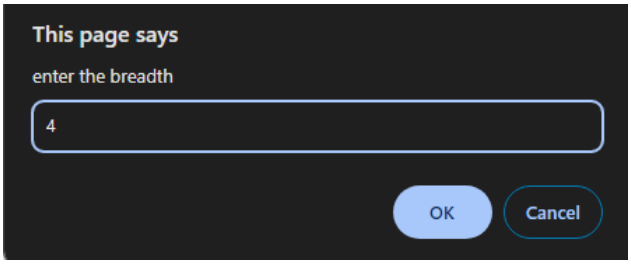


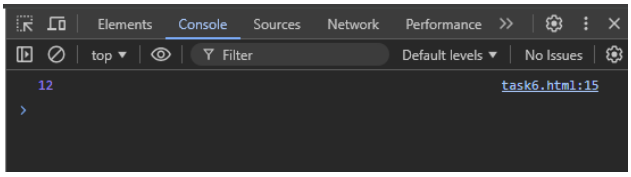
Task 47: Create a function that calculates the area of a rectangle.

```
<!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 area(a,b){
        return a*b;
       }
       let a=Number(prompt("enter the length"));
       let b=Number(prompt("enter the breadth"));
       console.log(area(a,b));
</script>
</body>
</html>
```

Output:



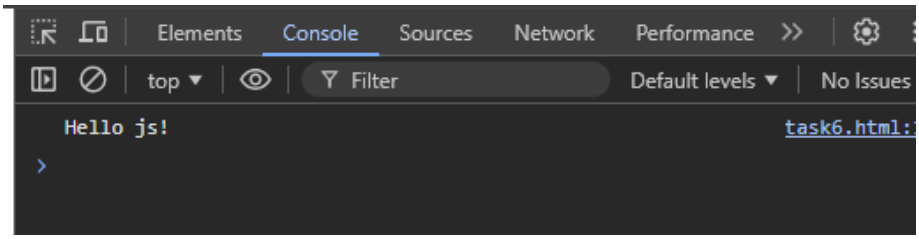




Task 48: Declare a function without parameters and call it.

```
<!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 a(){
          return "Hello js!";
       }
       console.log(a());
</script>
</body>
</html>
```

Output:

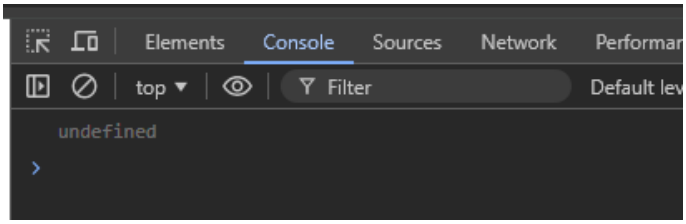


Task 49: Write a function that returns nothing and observe the default return value.

```
<!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 a(){

       }
       console.log(a());
</script>
</body>
</html>
```

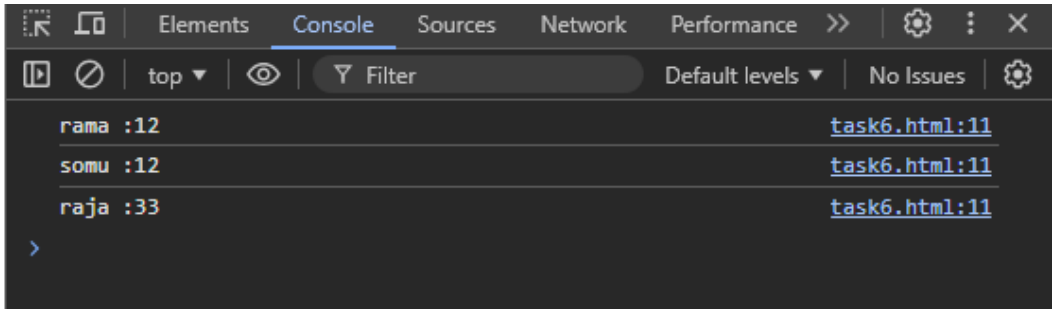
Output:



Task 50: Declare a function with default parameters and call it with different arguments.

```
<!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 a(name="rama",age=12){
          console.log(name+" :"+age);
       }
      a();
      a("somu");
      a("raja",33);
</script>
</body>
</html>
```

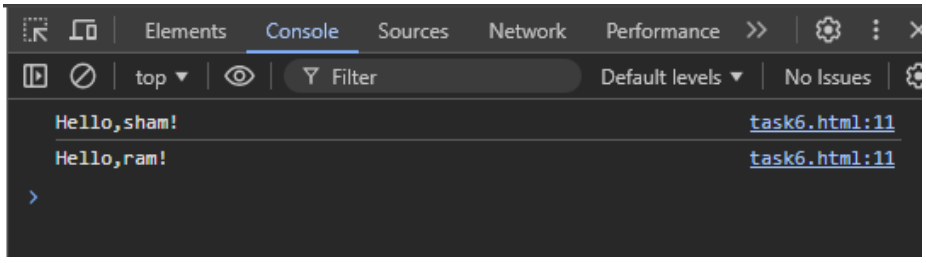
Output:



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.

```
<!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 fun=(name)=>{
        console.log("Hello,"+name+"!");
       }
       fun("sham");
       fun("ram");
</script>
</body>
</html>
```

Output:

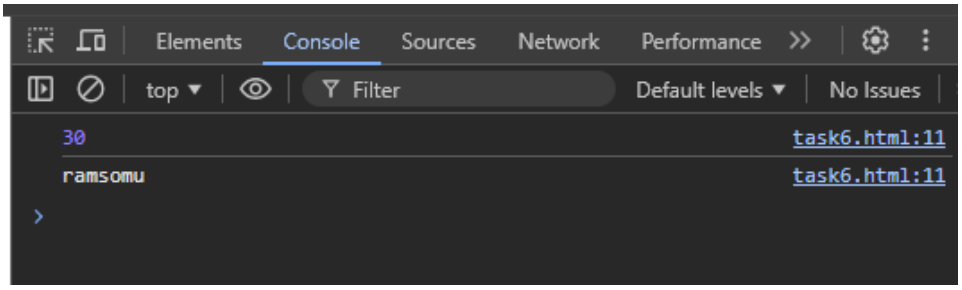


Task 52: Write an arrow function named add that takes two parameters and returns their sum. Validate your function with several pairs of numbers.

```
<!DOCTYPE html>
```

```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 fun=(m,n)=>{
        console.log(m+n);
       }
       fun(10,20);
       fun("ram","somu");
</script>
</body>
</html>
```

Output:

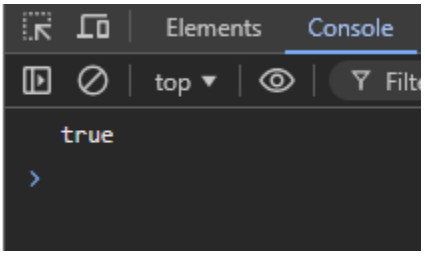


Task 53: Declare an arrow function named isEven that checks if a number is even. If the number is even, it should return true; otherwise, false. Remember that if the arrow function body has a single statement, you can omit the curly braces.

```html
<!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>
        add=(a)=>{
            if(a%2==0)
            console.log("true");
        else
```

```
        console.log("false");
        }
        add(2);
    </script>
</body>
</html>
```

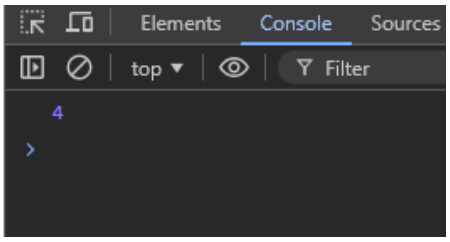
Output:



Task 54: Implement an arrow function named maxValue 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

```
<!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>
        maxvalue = (a,b)=>{
           if(a>b)
           return a;
        else
        return b;
        }
   console.log(maxvalue(2,4));
</script>
</body>
</html>
```

Output:



Task 55: Examine the behavior of the *this* keyword inside an arrow function vs atraditional 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.

```
<!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 = {
            num: 10,
            multiplytraditional: function () {
                console.log(this.num * 2);
            },
            multiplyarrow: () => {
                console.log(a.num * 2);
            },

            c() {
                this.multiplytraditional();
            },
            b() {
                this.multiplyarrow();
            },
        }
        a.b();
        a.c();
    </script>
</body>

</html>
```

Output: