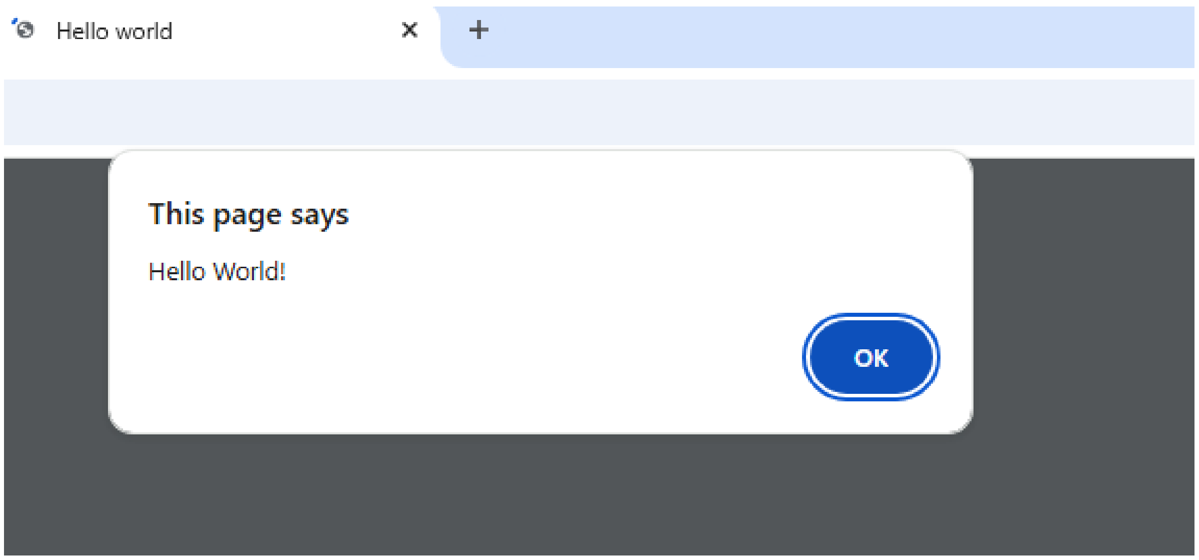
TASK 1

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

OUTPUT:



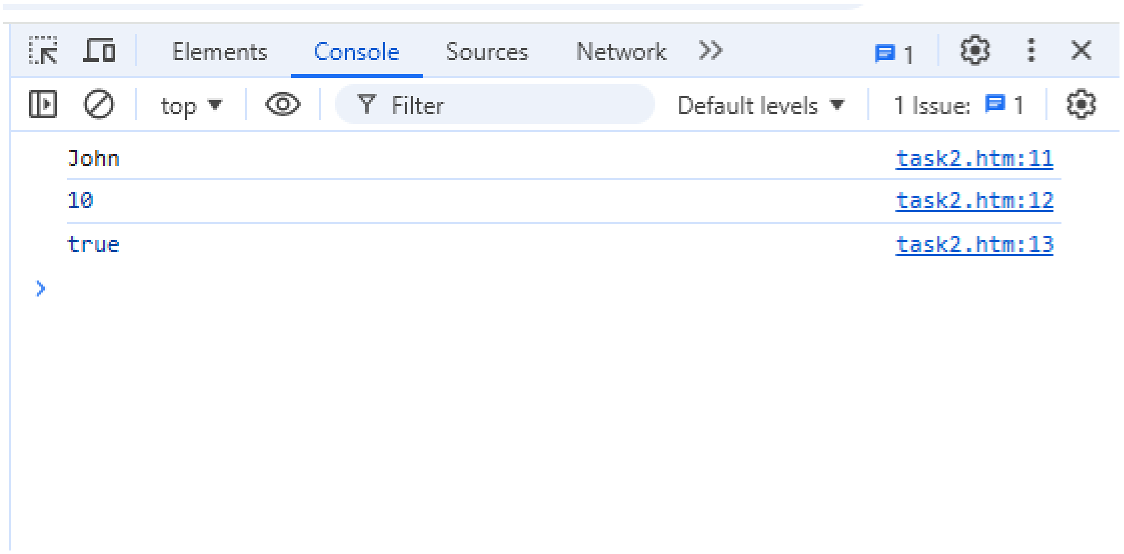
This page says

Hello World!

TASK 2

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
            let name="John";
            let num=10;
            let value=true;
            console.log(name);
            console.log(num);
            console.log(value);
            </script>
        </body>
</html>
```

OUTPUT:

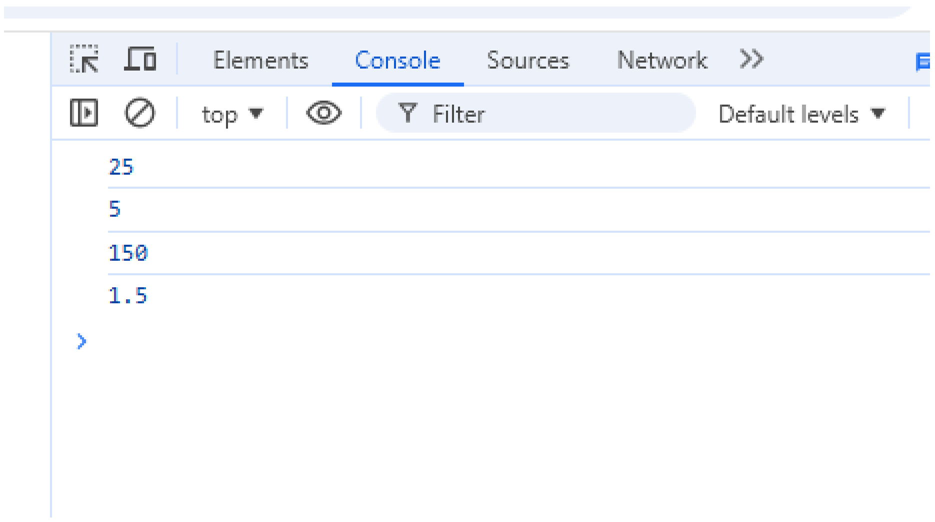


```
John                                              task2.htm:11
10                                                task2.htm:12
true                                              task2.htm:13
```

TASK 3

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
            let a=15;
            let b=10;
            console.log(a+b);
            console.log(a-b);
            console.log(a*b);
            console.log(a/b);
            </script>
        </body>
</html>
```

OUTPUT:

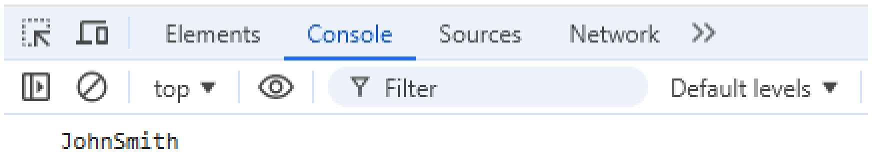


```
25
5
150
1.5
```

TASK 4

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
            let fname="John";
            let lname="Smith";
            console.log(fname+lname);
            </script>
        </body>
</html>
```

OUTPUT:

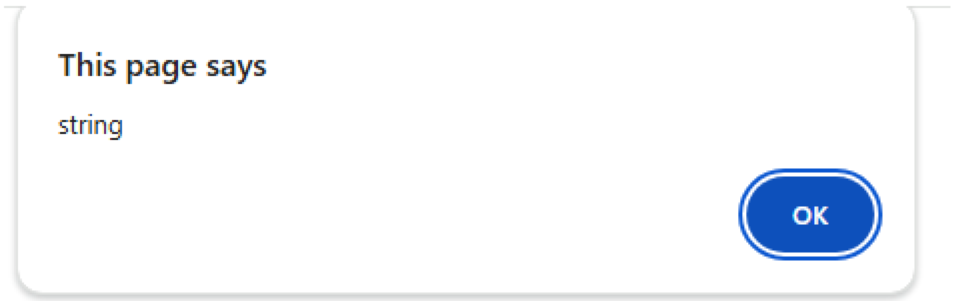


```
JohnSmith
```

TASK 5

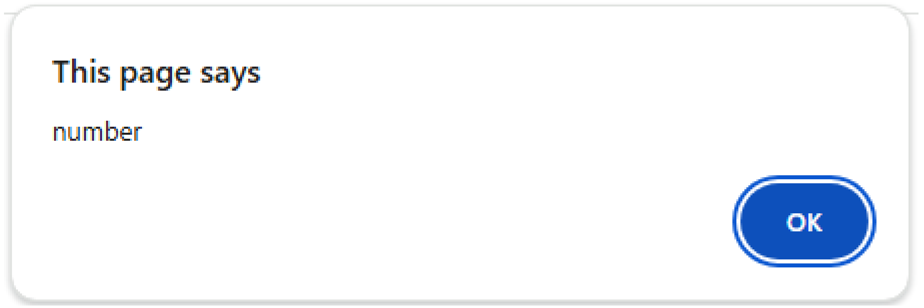
```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
            let fname="John";
            let num=100;
            alert(typeof fname);
            alert(typeof num);
            </script>
        </body>
</html>
```

OUTPUT:



This page says

string



This page says

number

TASK 6

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
          //This is a single line comment
          /* This is a multiline comment
            -where you can write the multiple lines of comment*/
            </script>
        </body>
</html>
```

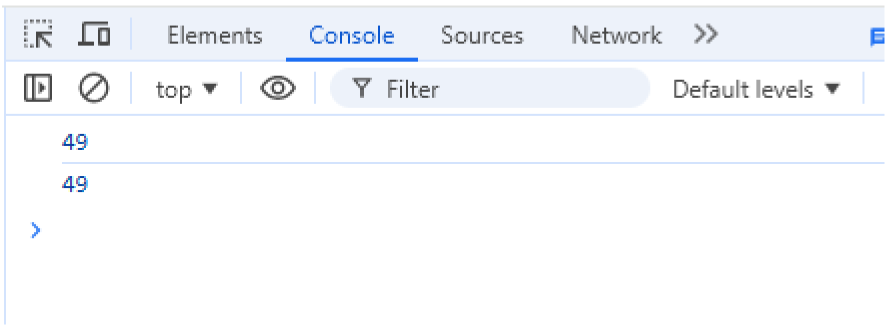
DIFFERENCE

- Single-line comments in JavaScript are pretty straightforward. They start with two slashes (//). Anything you write after these slashes on the same line won't run as code. It's just for you or others to read.
- For longer notes that take up more than one line, you use multi-line or block comments. These are surrounded by /* */. Everything between these markers is a comment, no matter how many lines it covers.

TASK 7

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
       let a=24;
       let b=25;
       console.log(a+b);
       let sum=a+b
       console.log(sum)
            </script>
        </body>
</html>
```

OUTPUT:



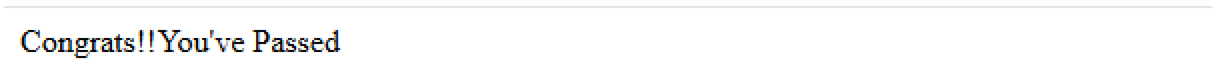
```
49
49
```

- **Semicolon-separation** makes the end of each statement explicit, reducing the chance of unexpected behavior, especially in complex or minified code.
- **Non-semicolon-separation** relies on ASI to automatically insert semicolons, which works in most simple cases, but can lead to subtle bugs or unexpected behavior, especially when multiple statements are written on a single line or in edge cases (e.g., `return` statements).

TASK 8

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
           let mark=89;
           if(isNaN(mark)){
              alert("Please enter valid mark");
           }
           else {
              if(mark<50){
                document.writeln("Fail");
              }
              else if(mark>50){
                document.writeln("Congrats!!You've Passed");
              }
           }
            </script>
        </body>
</html>
```

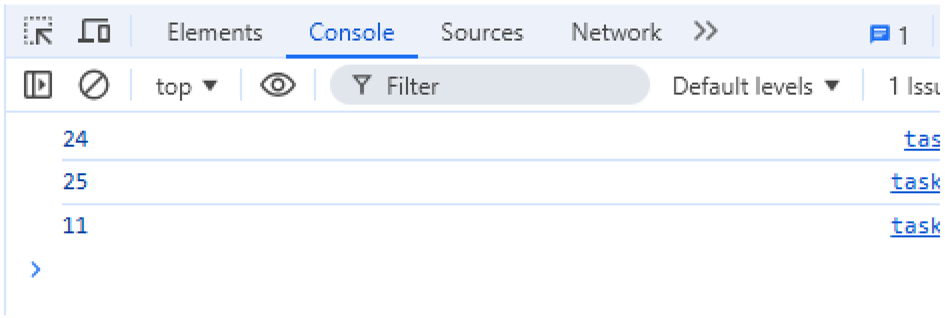
OUTPUT:


Congrats!!You've Passed

TASK 9

```
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        </head>
        <body>
            <script>
           let num1=24,num2=25,num3=11;
           console.log(num1);
           console.log(num2);
           console.log(num3);
            </script>
        </body>
</html>
```

OUTPUT:



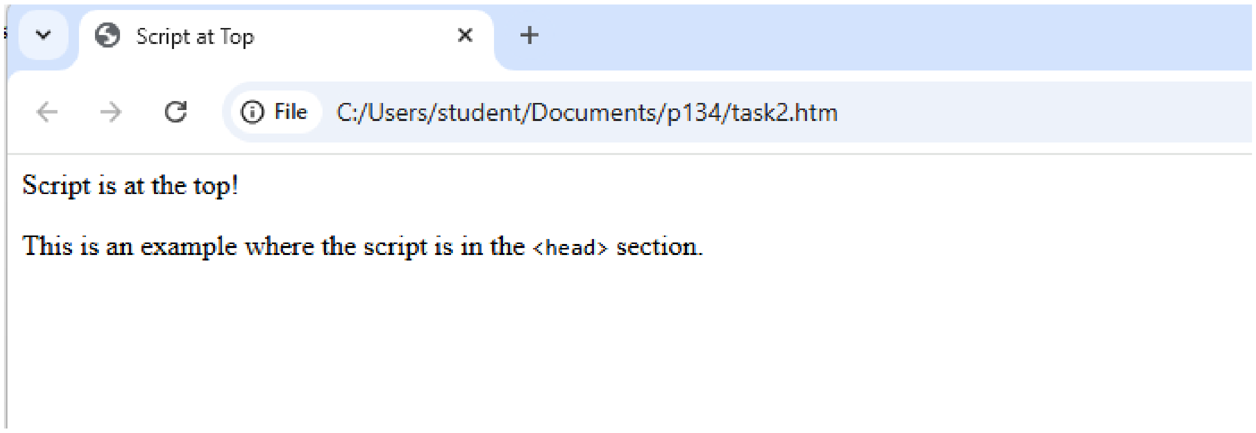
```
24          tas
25          task
11          task
>
```

TASK 10

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Script at Top</title>
    <script>
        document.write("Script is at the top!");
    </script>
```

```html
</head>
<body>
    <p>This is an example where the script is in the <code>&lt;head&gt;</code>
section.</p>
</body>
</html>
```

OUTPUT:



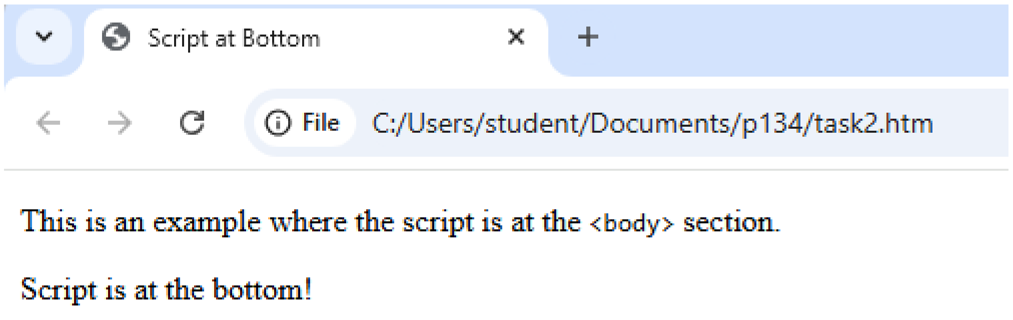
```
C:/Users/student/Documents/p134/task2.htm
```

Script is at the top!

This is an example where the script is in the `<head>` section.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Script at Bottom</title>
</head>
<body>
    <p>This is an example where the script is at the <code>&lt;body&gt;</code>
section.</p>
    <script>
        document.write("Script is at the bottom!");
    </script>
</body>
</html>
```

OUTPUT:



This is an example where the script is at the `<body>` section.

Script is at the bottom!

DIFFERENCE :

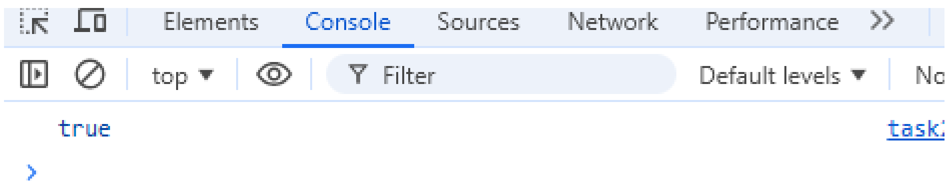
- **Script in the `<head>`**: This can be useful for essential scripts that need to be loaded early (e.g., polyfills, analytics scripts), but generally, it may slow down the page load and can cause issues with DOM manipulation.
- **Script at the Bottom of the `<body>`**: This is the best practice for most use cases because it avoids blocking the page rendering and ensures that the DOM is ready when the script runs.

For better performance and fewer potential issues, it's generally recommended to place `<script>` tags just before the closing `</body>` tag, unless there's a specific need to load the script earlier.

TASK 11

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

OUTPUT:

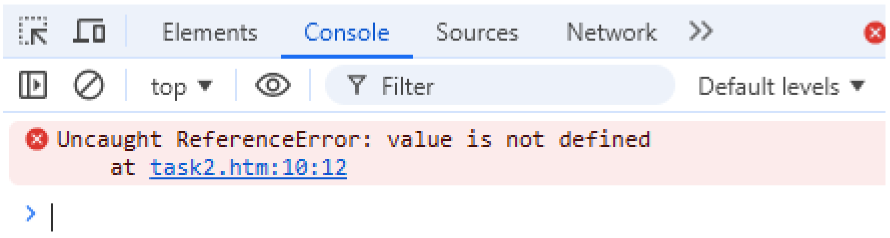


```
true                                                  task1
>
```

TASK 12

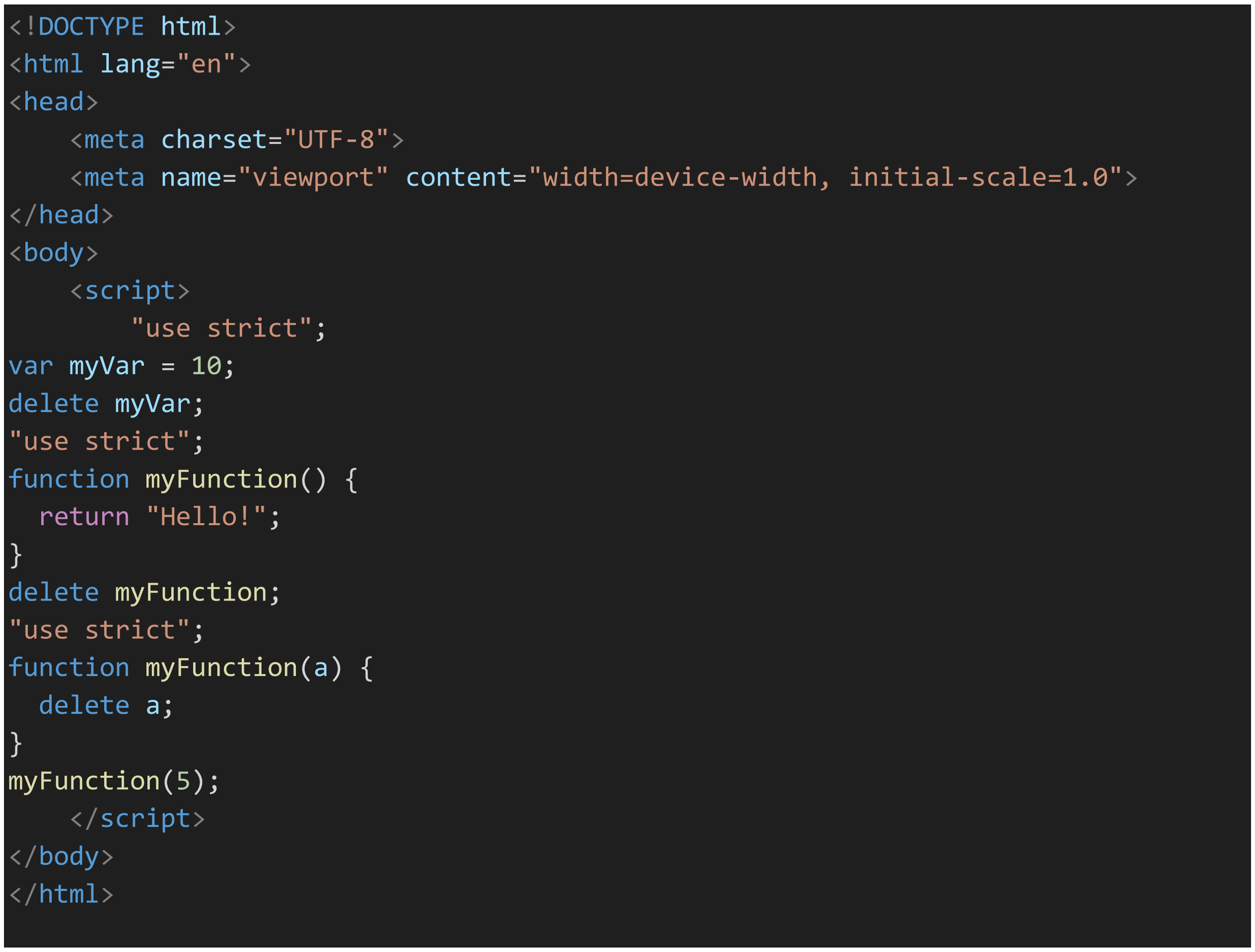
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
        "use strict";
      value=true;
      console.log(value);
    </script>
</body>
</html>
```

OUTPUT:



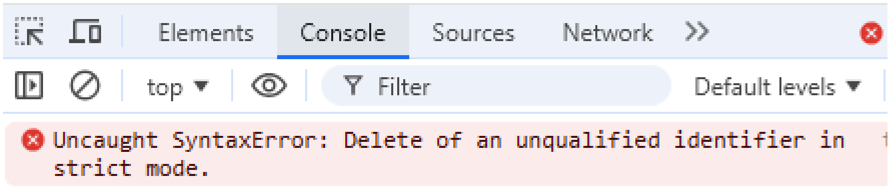
```
Uncaught ReferenceError: value is not defined
    at task2.htm:10:12
```

TASK 13



```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
        "use strict";
var myVar = 10;
delete myVar;
"use strict";
function myFunction() {
  return "Hello!";
}
delete myFunction;
"use strict";
function myFunction(a) {
  delete a;
}
myFunction(5);
    </script>
</body>
</html>
```

OUTPUT:



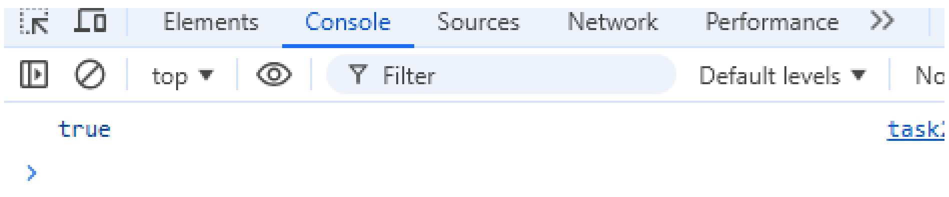
```
Uncaught SyntaxError: Delete of an unqualified identifier in
strict mode.
```

TASK 14:

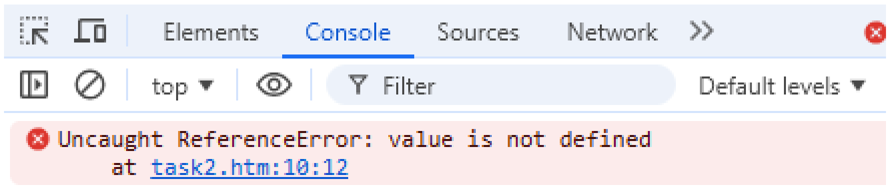
```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
      value=true;
      console.log(value);
    </script>
</body>
</ html>

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

OUTPUT:



```
true                                                    task
>
```

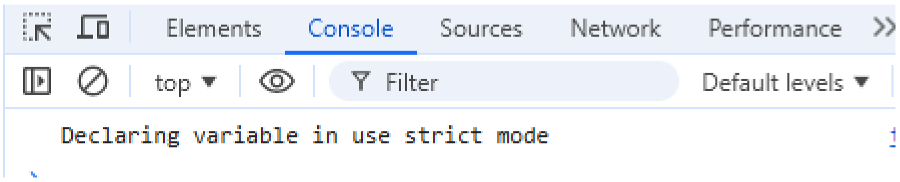


```
Uncaught ReferenceError: value is not defined
    at task2.htm:10:12
```

TASK 15

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
        "use strict";
var Line="Declaring variable in use strict mode";
console.log(Line);
    </script>
</body>
</html>
```

OUTPUT:

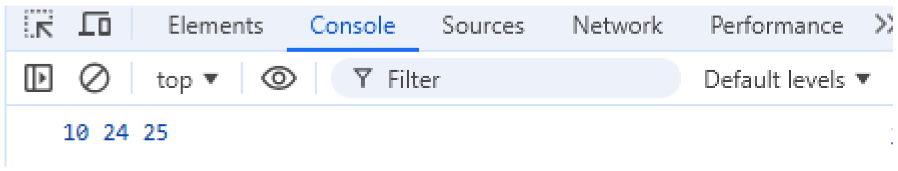


```
Declaring variable in use strict mode
```

TASK 16

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
        let a=10;
        var b=24;
        const c=25;
        console.log(a,b,c);
    </script>
</body>
</html>
```

OUTPUT:



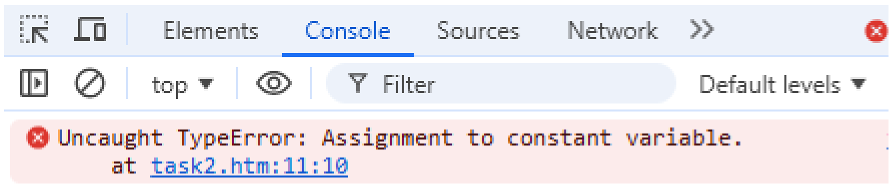
```
10 24 25
```

- **Avoid using var** in modern JavaScript because it has some tricky behaviors (like hoisting) and doesn't have block scope.
- If you need a variable to be scoped only to the function (not block), and you're working with older JavaScript code, you might still encounter var frequently.
- **Use let for variables that need to be reassigned** but should be **confined to a specific block scope** (e.g., loops, conditional statements).
- Use let when you need to declare variables inside a block of code that should not leak into the surrounding code.
- **Use const for variables that should never be reassigned**. This is the preferred choice for constants, function references, and other values that should remain unchanged after initialization.
- **Use const for all variables** whose reference or value should remain constant (e.g., configuration settings, array or object references, etc.).

TASK 17

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
        const c=25;
        c=24;
        console.log(a,b,c);
    </script>
</body>
</html>
```

OUTPUT:

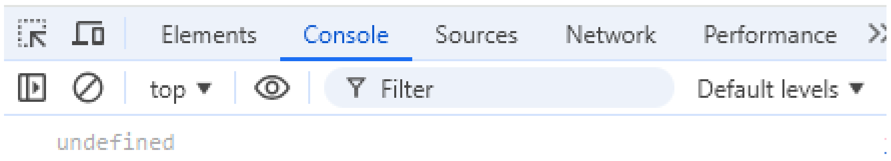


```
Uncaught TypeError: Assignment to constant variable.
    at task2.htm:11:10
```

TASK 18

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

OUTPUT:

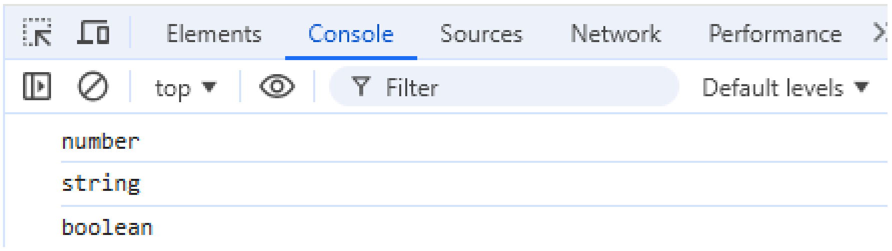


```
undefined
```

TASK 19

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

OUTPUT:

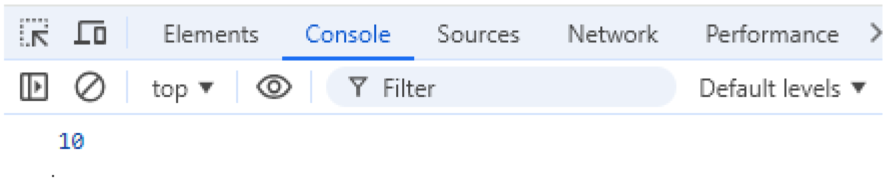


```
number
string
boolean
```

TASK 20

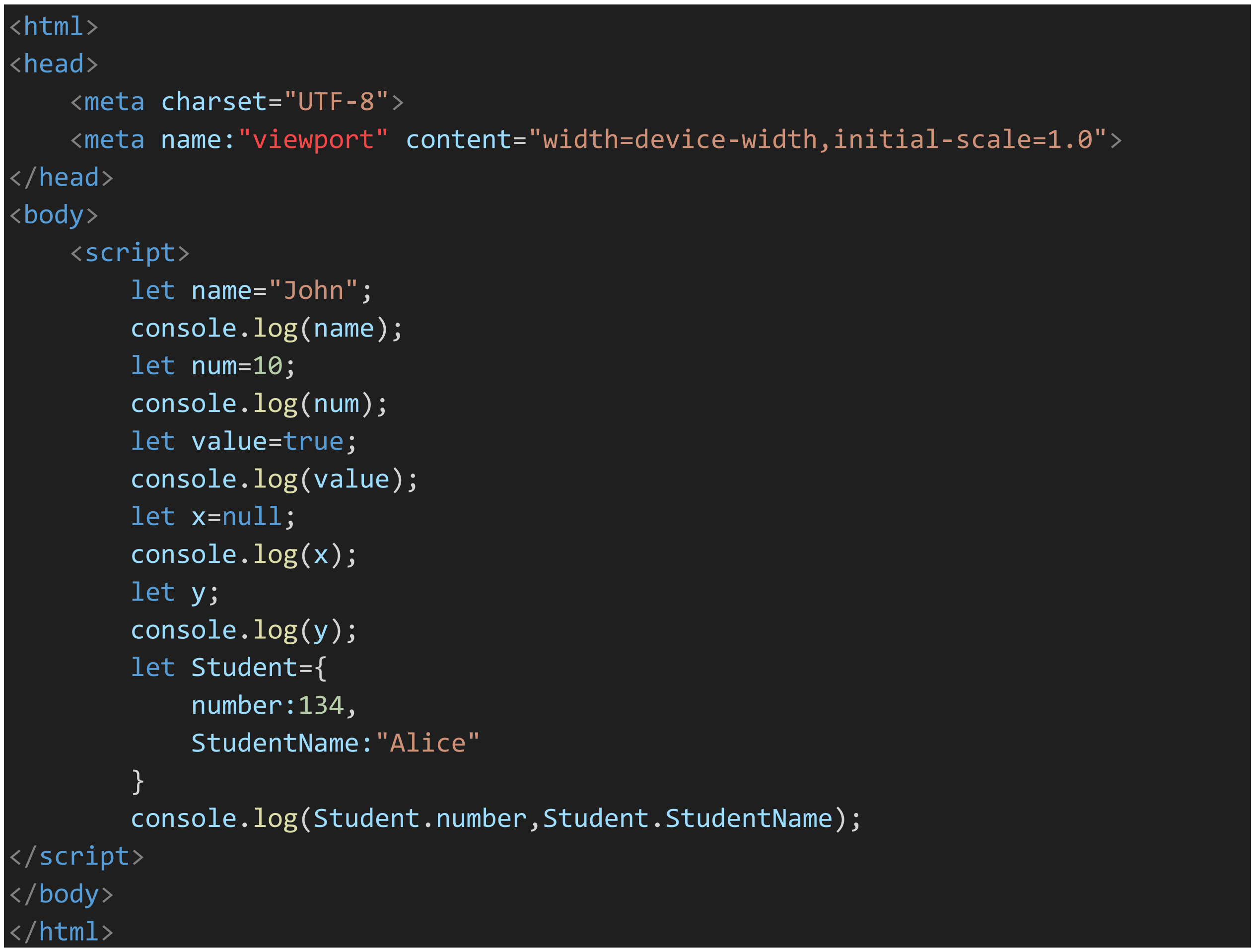
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
    <script>
        var x=10;
        var y=x;
        console.log(y);
    </script>
</body>
</html>
```

OUTPUT:



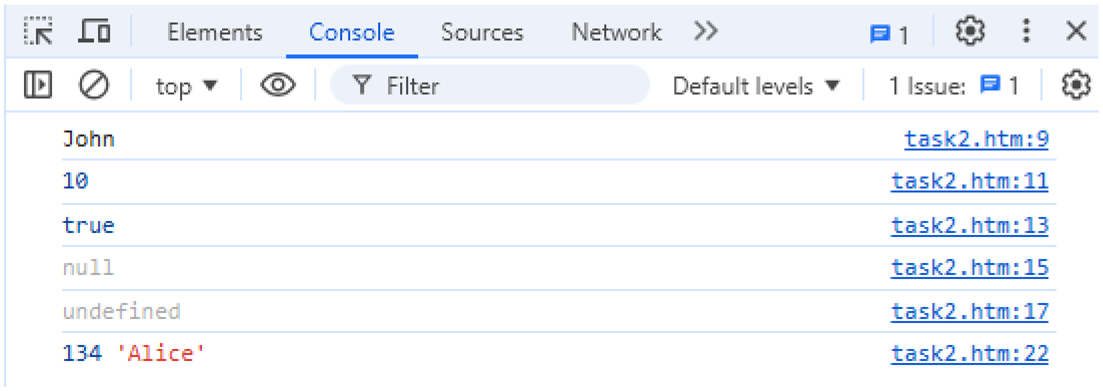
```
10
```

TASK 21



```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
        let name="John";
        console.log(name);
        let num=10;
        console.log(num);
        let value=true;
        console.log(value);
        let x=null;
        console.log(x);
        let y;
        console.log(y);
        let Student={
            number:134,
            StudentName:"Alice"
        }
        console.log(Student.number,Student.StudentName);
</script>
</body>
</html>
```

OUTPUT:

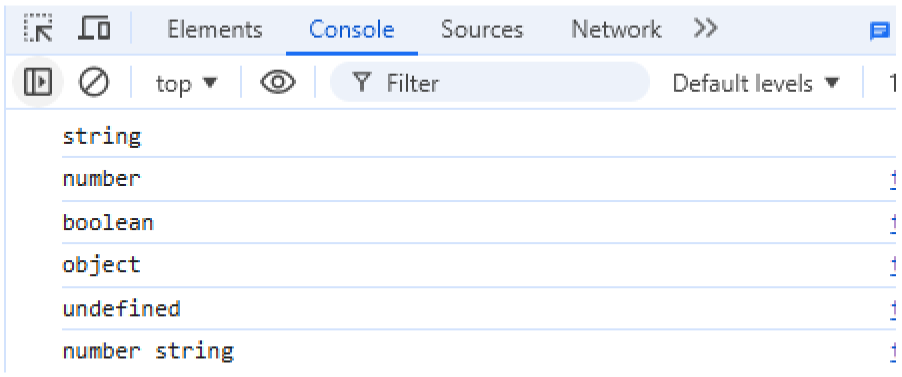


```
John                                   task2.htm:9
10                                     task2.htm:11
true                                   task2.htm:13
null                                   task2.htm:15
undefined                              task2.htm:17
134 'Alice'                            task2.htm:22
```

TASK 22

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
        let name="John";
        console.log(typeof name);
        let num=10;
        console.log(typeof num);
        let value=true;
        console.log(typeof value);
        let x=null;
        console.log(typeof x);
        let y;
        console.log(typeof y);
        let Student={
            number:134,
            StudentName:"Alice"
        }
        console.log(typeof Student.number,typeof Student.StudentName);
</script>
</body>
</html>
```

OUTPUT:

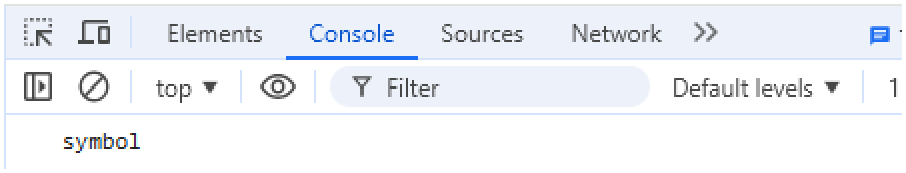


```
string
number
boolean
object
undefined
number string
```

TASK 23

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
        let a=Symbol("W");
        console.log(typeof a);
</script>
</body>
</html>
```

OUTPUT:

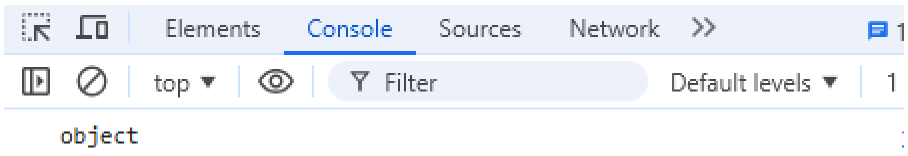


```
symbol
```

TASK 24

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
        let a=null;
        console.log(typeof a);
</script>
</body>
</html>
```

OUTPUT:



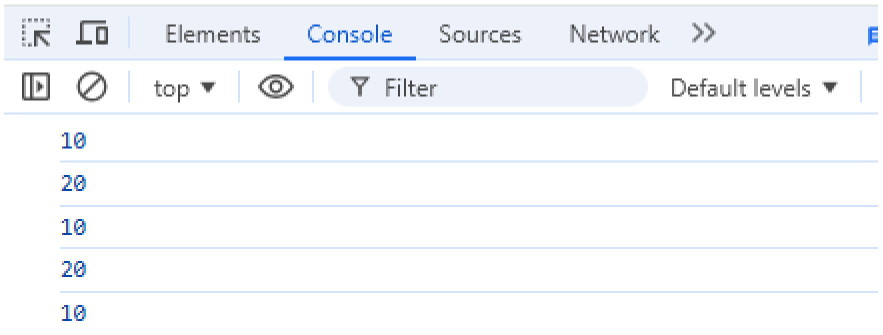
```
object
```

TASK 25

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    function exampleVar() {
    var a = 10;
    if (true) {
        var b = 20;
    }
    console.log(a);
    console.log(b);
}
exampleVar();

function exampleLet() {
    let a = 10;
    if (true) {
        let b = 20;
        console.log(a);
        console.log(b);
    }
    console.log(a);
}
exampleLet();
</script>
</body>
</html>
```

OUTPUT:



```
10
20
10
20
10
```

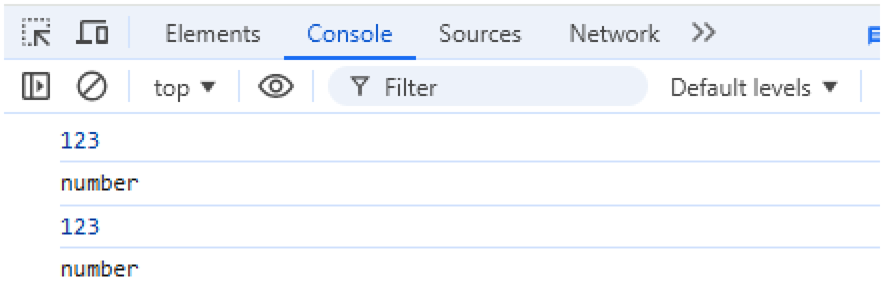
- **var** is **function-scoped**, meaning it is accessible throughout the entire function, regardless of block boundaries. It has potential pitfalls with hoisting and unintentional variable overwrites.
- **let** is **block-scoped**, making it a more predictable choice for modern JavaScript development, especially when dealing with blocks, loops, or conditionals. It reduces the chance of variable leakage and improves code clarity.

In modern JavaScript, it's generally recommended to use `let` (or `const` for constants) instead of `var` for better scoping and fewer errors.

TASK 26

```html
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport"  content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
    let str = "123";
let num = str * 1;
console.log(num);
console.log(typeof num);
let str1 = "123";
let num1 = Number(str1);
console.log(num);
console.log(typeof num1);
</script>
</body>
</html>
```

OUTPUT:

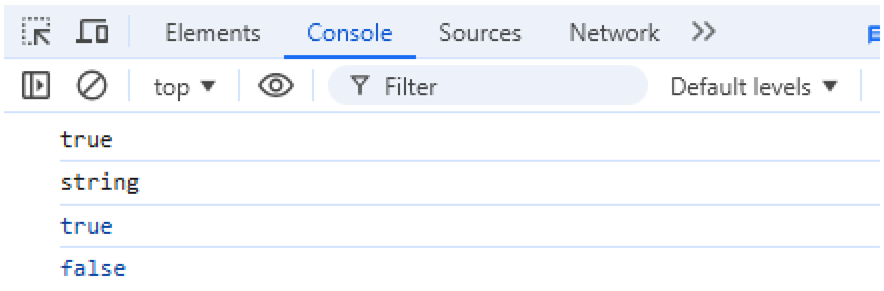


```
123
number
123
number
```

TASK 27

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
   let boolValue = true;
let strValue = boolValue.toString();
console.log(strValue);
console.log(typeof strValue);
let strValue1 = "hello";
let boolValue1 = Boolean(strValue1);
console.log(boolValue1);
let emptyStr = "";
let boolEmpty = Boolean(emptyStr);
console.log(boolEmpty);
</script>
</body>
</html>
```

OUTPUT:

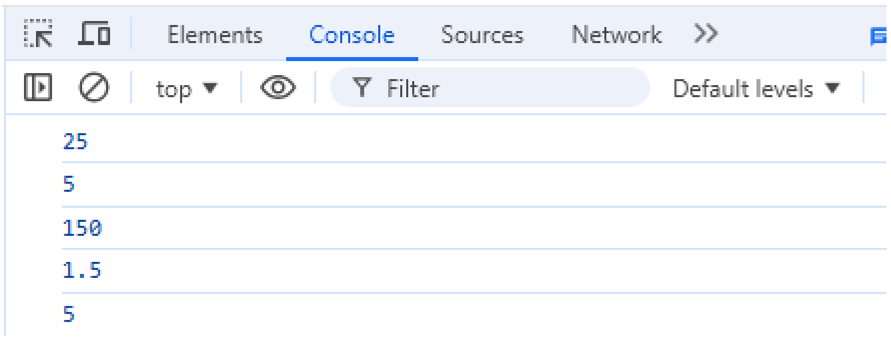


```
true
string
true
false
```

TASK 28

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
   let a=15;
   let b=10;
    console.log(a+b);
    console.log(a-b);
    console.log(a*b);
    console.log(a/b);
    console.log(a%b);
</script>
</body>
</html>
```

OUTPUT:



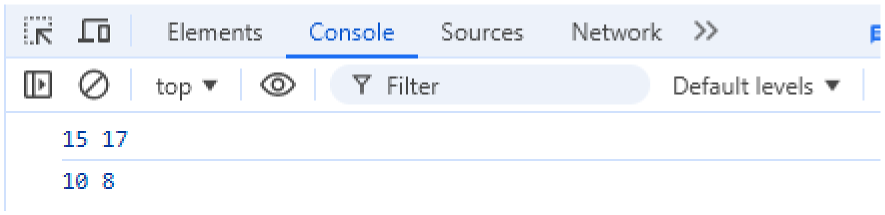
```
25
5
150
1.5
5
```

TASK 29

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
   let a=15;
   let b=10;
    console.log(a++,++a);
    console.log(b--,--b);
```

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

OUTPUT:

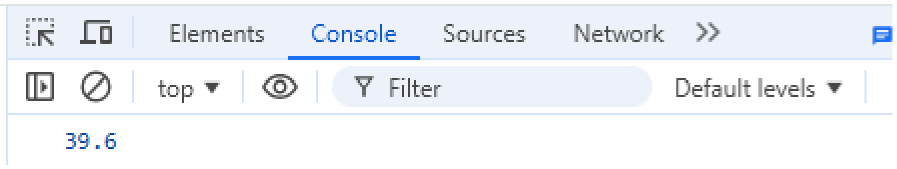


```
15 17
10 8
```

TASK 30

```
<html>
<head>
    <meta charset="UTF-8">
    <meta name:"viewport" content="width=device-width,initial-scale=1.0">
</head>
<body>
    <script>
   let a=15;
   let b=10;
   let c=24;
   let d=25;
   let result=a+b*(c/d)+a;
   console.log(result);
</script>
</body>
</html>
```

OUTPUT:

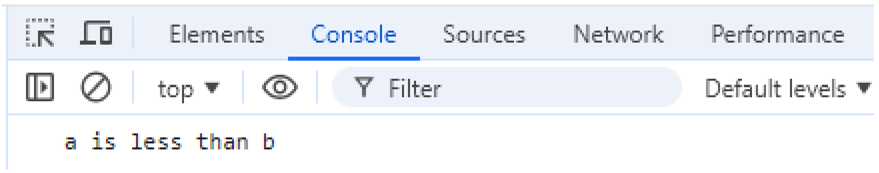


```
39.6
```

TASK 31

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
            let a=10;
            let b=15;
            if(a<b){
                console.log("a is less than b");
            }
            else if(a>b){
                console.log("a is grater than b");
            }
            else if(a<=b){
                console.log("a is less than equal to b ");
            }
            else {
                console.log("a is grater than equal to b");
            }
        </script>
    </body>
</html>
```

OUTPUT:

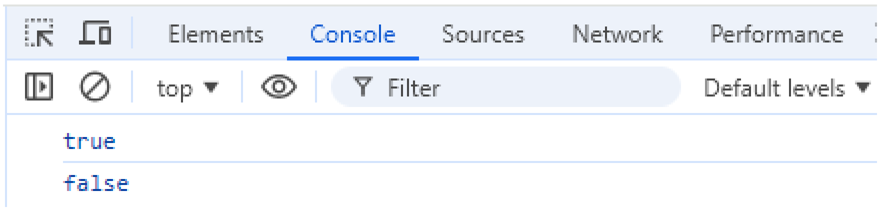


```
a is less than b
```

TASK 32

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
            let a=10;
            let b="10";
           console.log(a==b);
           console.log(a===b);
        </script>
    </body>
</html>
```

OUTPUT:



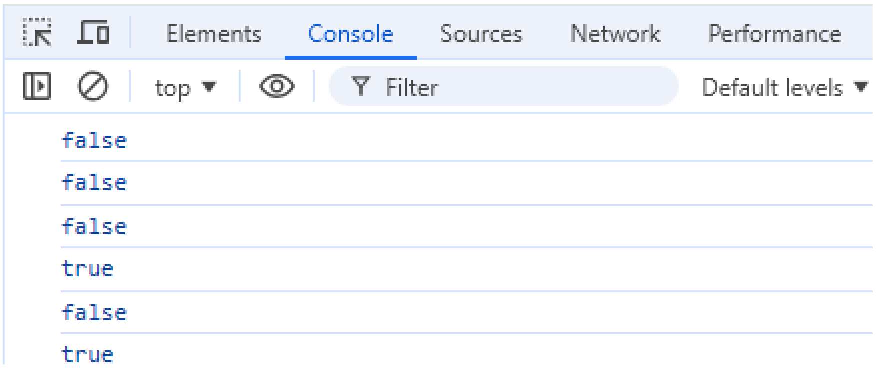
```
true
false
```

- == allows **type coercion**, making comparisons "looser" but sometimes unpredictable.
- === is **strict** and avoids type coercion, making comparisons more reliable.

TASK 33

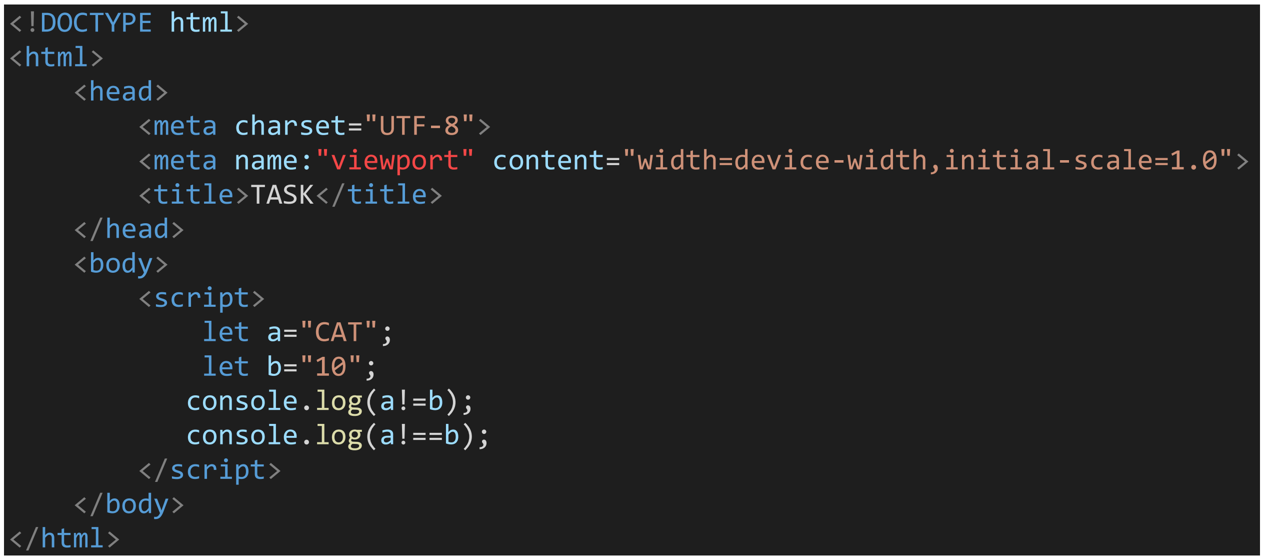
```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
            let a="CAT";
            let b="10";
           console.log(a==b);
           console.log(a===b);
           console.log(a<b);
           console.log(a>b);
           console.log(a<=b);
           console.log(a>=b);
        </script>
    </body>
</html>
```

OUTPUT:



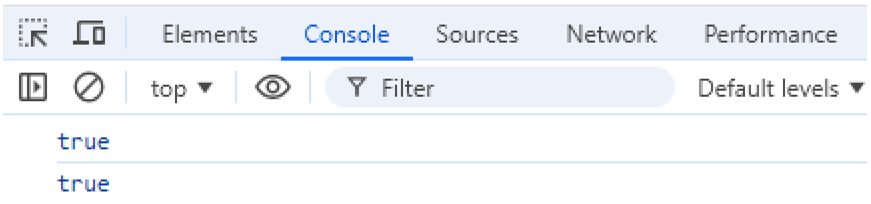
```
false
false
false
true
false
true
```

TASK 34



```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
            let a="CAT";
            let b="10";
           console.log(a!=b);
           console.log(a!==b);
        </script>
    </body>
</html>
```

OUTPUT:

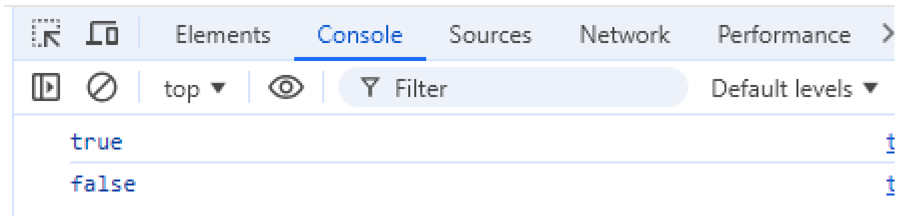


```
true
true
```

TASK 35

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
           console.log(null==undefined);
           console.log(null===undefined);
        </script>
    </body>
</html>
```

OUTPUT:

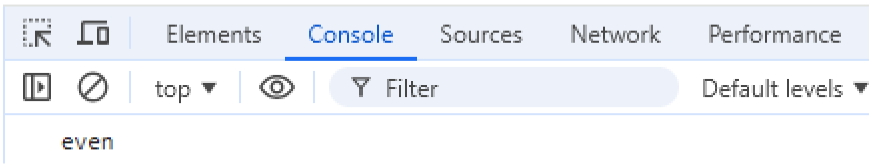


```
true
false
```

TASK 36

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
            let a=24;
             if(a%2==0){
                console.log("even");
             }
             else{
                console.log("odd");
             }
        </script>
    </body>
</html>
```

OUTPUT:

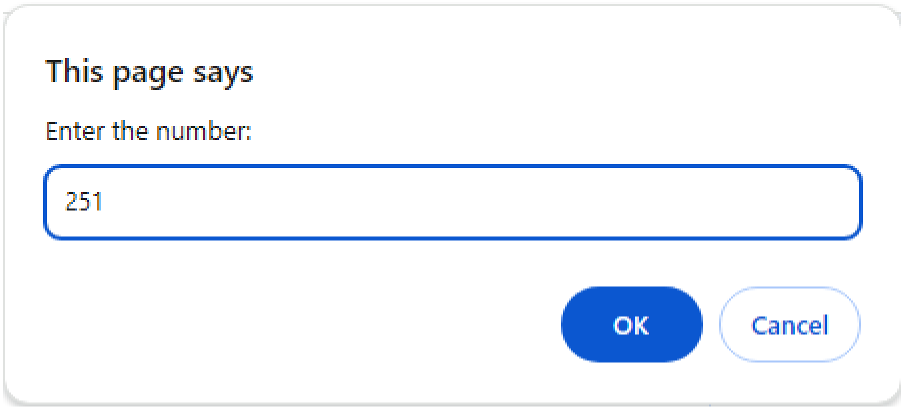


```
even
```

TASK 37

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num=prompt("Enter the number:");
          if(num<0){
            alert("Negative");
          }
          else if(num>0){
            alert("Positive");
          }
          else{
            alert("zero");
          }
        </script>
    </body>
</html>
```

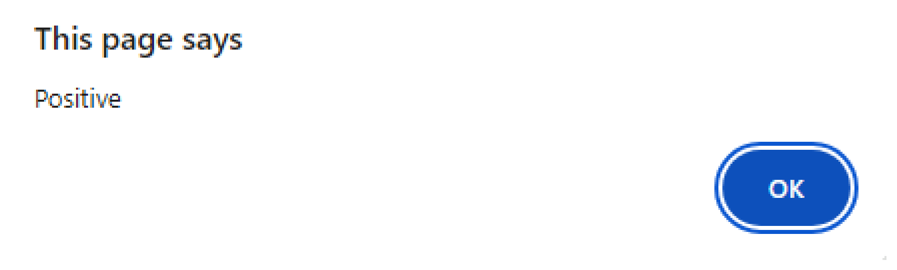
OUTPUT:



This page says

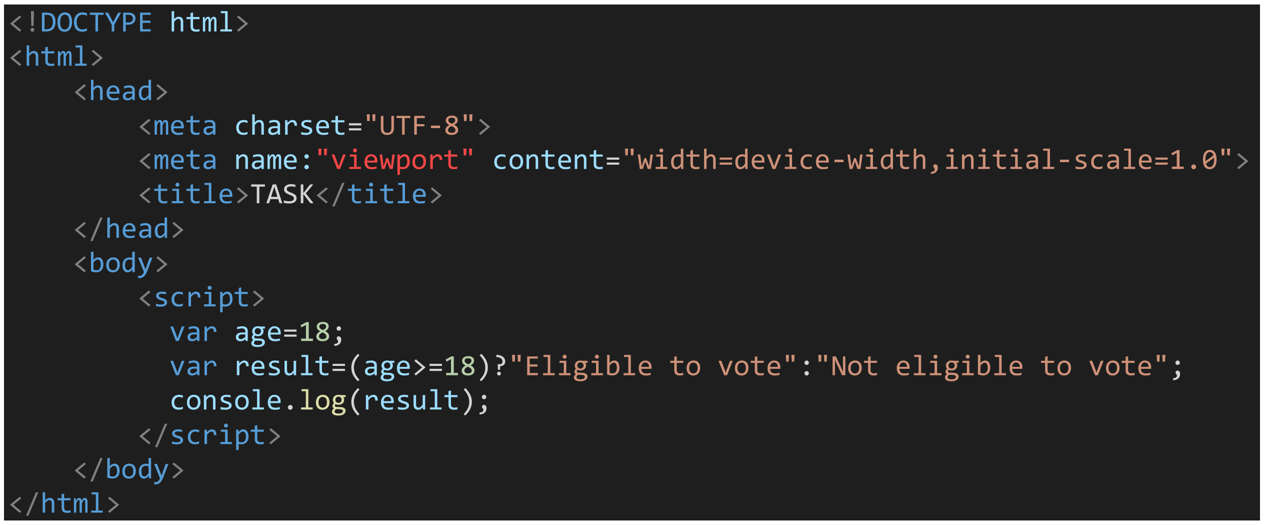
Enter the number:

251

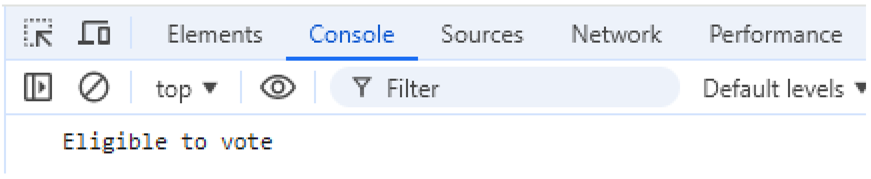

This page says

Positive

TASK 38

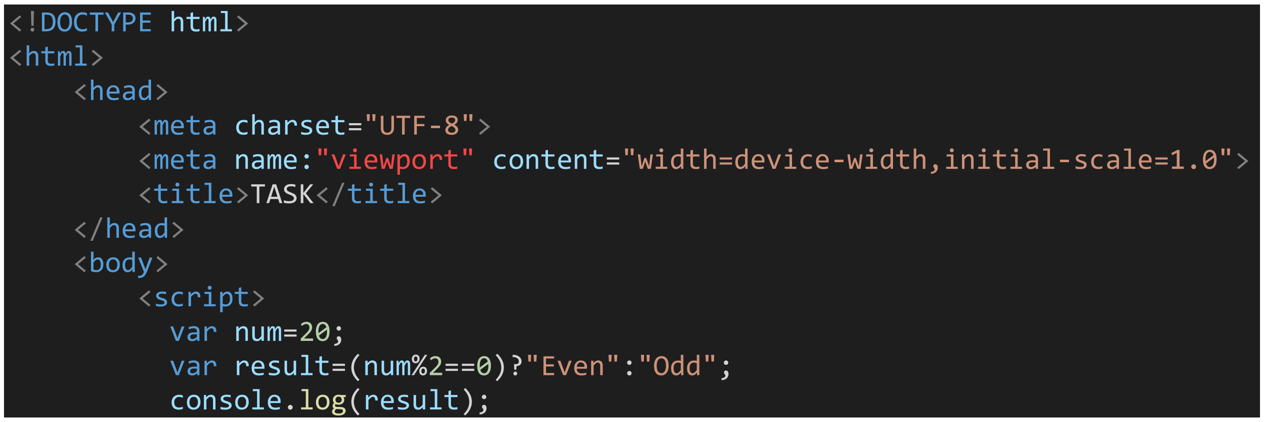

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var age=18;
          var result=(age>=18)?"Eligible to vote":"Not eligible to vote";
          console.log(result);
        </script>
    </body>
</html>
```

OUTPUT:

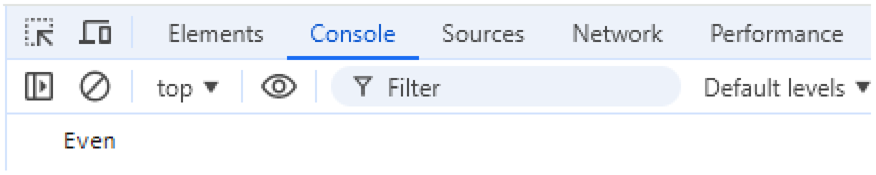


```
Eligible to vote
```

TASK 39


```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num=20;
          var result=(num%2==0)?"Even":"Odd";
          console.log(result);
        </script>
    </body>
</html>
```

OUTPUT:

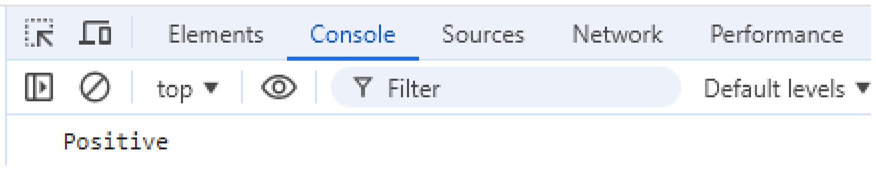


```
Even
```

TASK 40

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num=20;
          var result=(num>=0)?"Positive":"Negative";
          console.log(result);
        </script>
    </body>
</html>
```

OUTPUT:

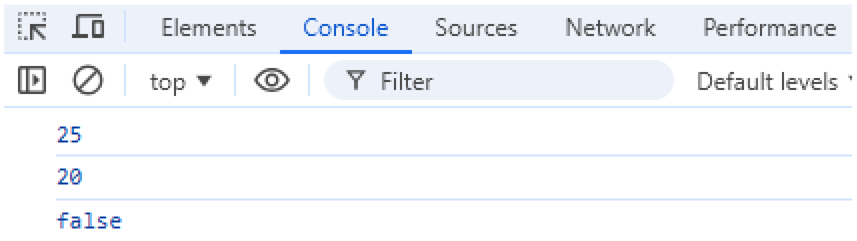


```
Positive
```

TASK 41

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num1=20,num2=25;
          console.log(num1&&num2);
          console.log(num1||num2);
          console.log(!num2);
        </script>
    </body>
</html>
```

OUTPUT:

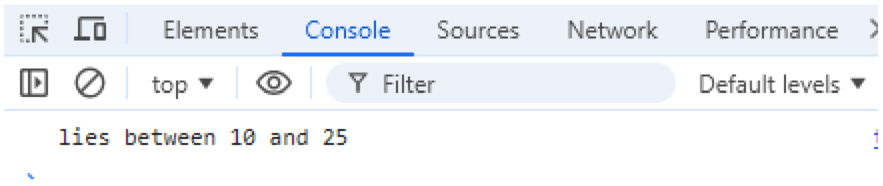


```
25
20
false
```

TASK 42

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num1=20;
          if(num1>=10&&num1<=25){
            console.log("lies between 10 and 25");
          }
          else {
            console.log("Not in the range");
          }
        </script>
    </body>
</html>
```

OUTPUT:

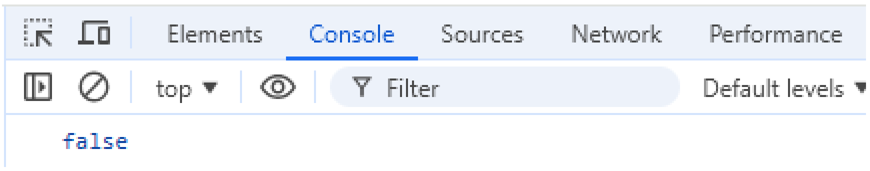


```
lies between 10 and 25
```

TASK 43

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var value=true;
          console.log(!value);
        </script>
    </body>
</html>
```

OUTPUT:

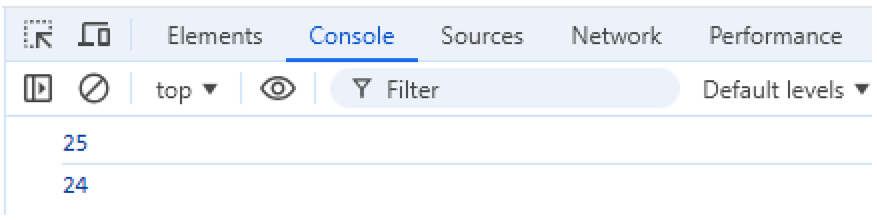


```
false
```

TASK 44

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num=24,num1=25;
          console.log(num&&num1);
          console.log(num||num1);
        </script>
    </body>
</html>
```

OUTPUT:

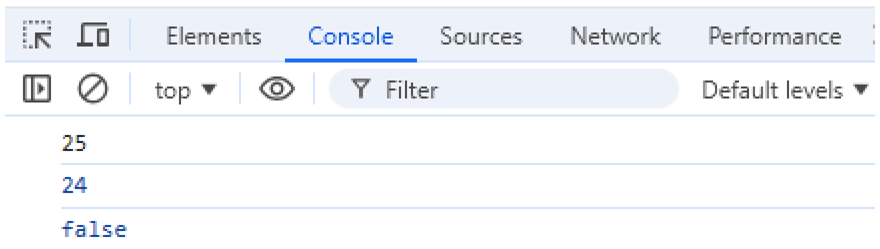


```
25
24
```

TASK 45

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <meta name:"viewport" content="width=device-width,initial-scale=1.0">
        <title>TASK</title>
    </head>
    <body>
        <script>
          var num=24,num1="25";
          console.log(num&&num1);
          console.log(num||num1);
          console.log(!num);
        </script>
    </body>
</html>
```

OUTPUT:

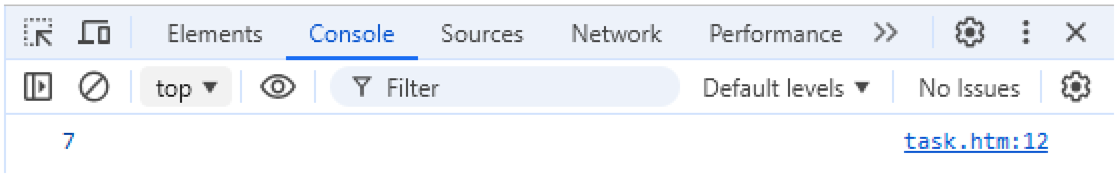


```
25
24
false
```

TASK 46

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
let add=function sum(a,b){
    return a+b;
}
console.log(add(2,5));
</script>
</body>
</html>
```

OUTPUT:

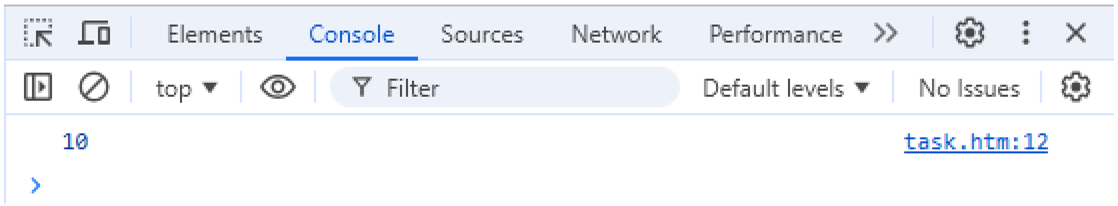


```
7                                   task.htm:12
```

TASK 47

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
let area=function rectArea(a,b){
    return a*b;
}
console.log(area(2,5));
</script>
</body>
</html>
```

OUTPUT:



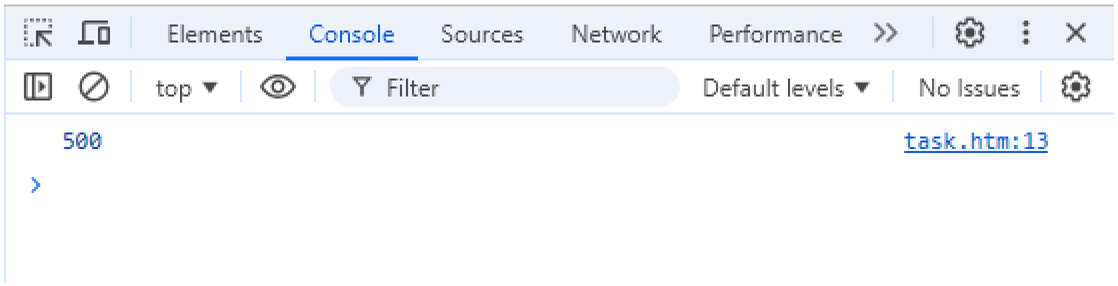
```
10                                  task.htm:12
>
```

TASK 48

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

```
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
function rectArea(){
   let a=20,b=25;
   return a*b;
}
console.log(rectArea());
</script>
</body>
</html>
```

OUTPUT:

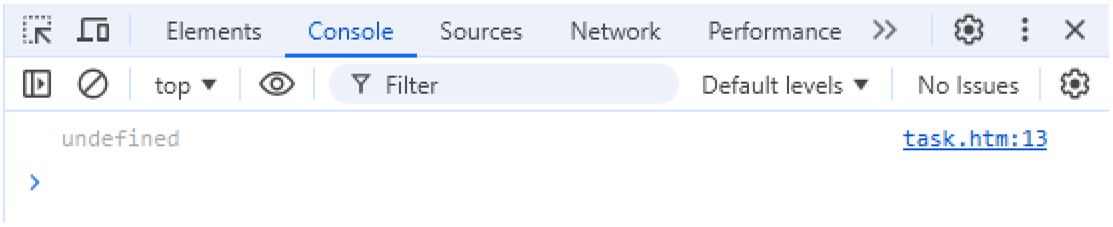


```
500                                                    task.htm:13
```

TASK 49

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
function rectArea(){
   let a=20,b=25;
   return ;
}
console.log(rectArea());
</script>
</body>
</html>
```

OUTPUT:

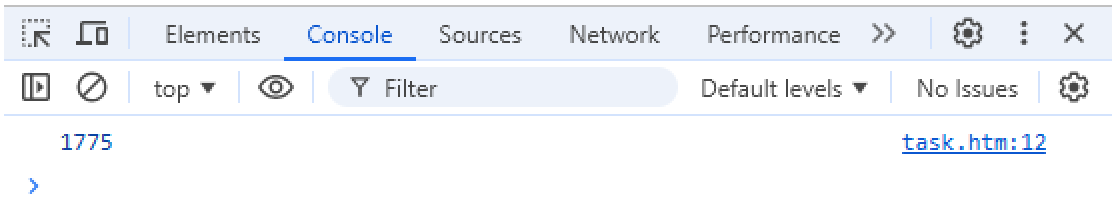


```
undefined                                              task.htm:13
```

TASK 50

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
function mul(a,b=5){
    return a*b;
}
console.log(mul(355));
</script>
</body>
</html>
```

OUTPUT:

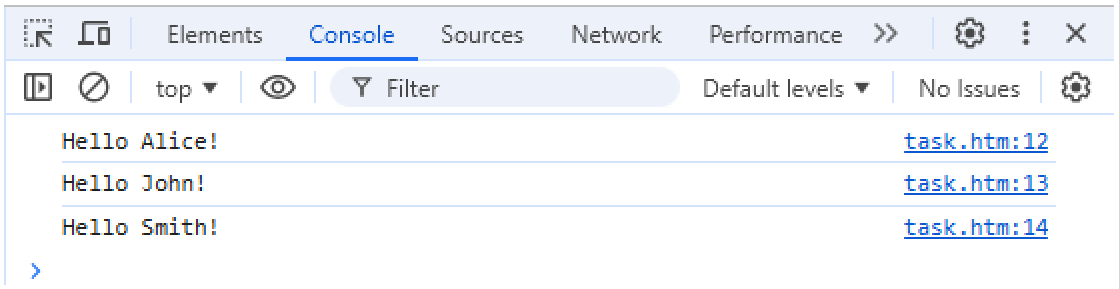


```
1775                                                    task.htm:12
>
```

TASK 51

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
let greet=(name)=>{
    return "Hello "+name+"!";
}
console.log(greet("Alice"));
console.log(greet("John"));
console.log(greet("Smith"));
</script>
</body>
</html>
```

OUTPUT:

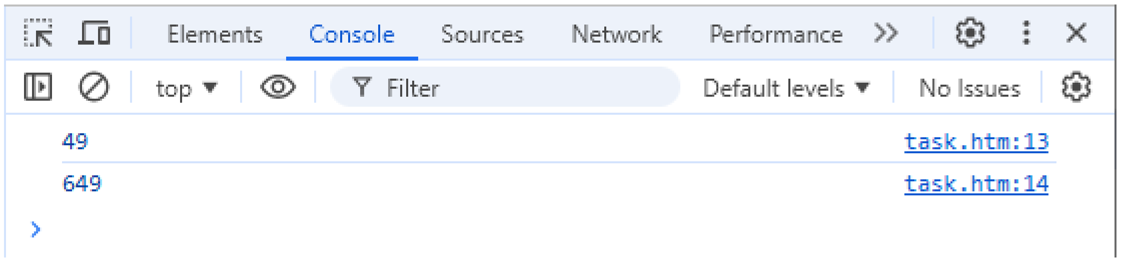


```
Hello Alice!                                   task.htm:12
Hello John!                                    task.htm:13
Hello Smith!                                   task.htm:14
```

TASK 52

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
let add=(num1,num2)=>{
	return num1+num2;
}
console.log(add(43,6));
console.log(add(4,645));
</script>
</body>
</html>
```

OUTPUT:

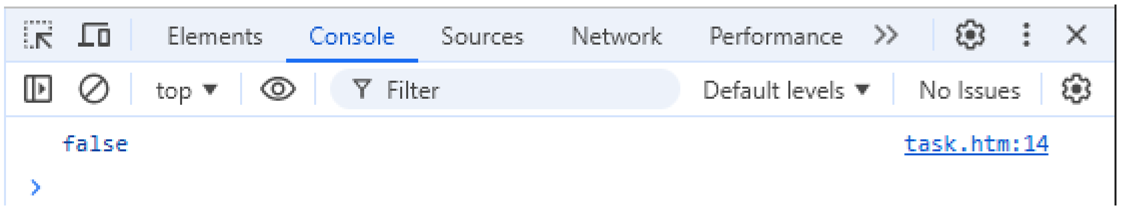


```
49                                             task.htm:13
649                                            task.htm:14
```

TASK 53

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
let isEven=(num)=>{
   if(num%2==0)
   return true;
else return false;
}
console.log(isEven(4365));
</script>
</body>
</html>
```

OUTPUT:

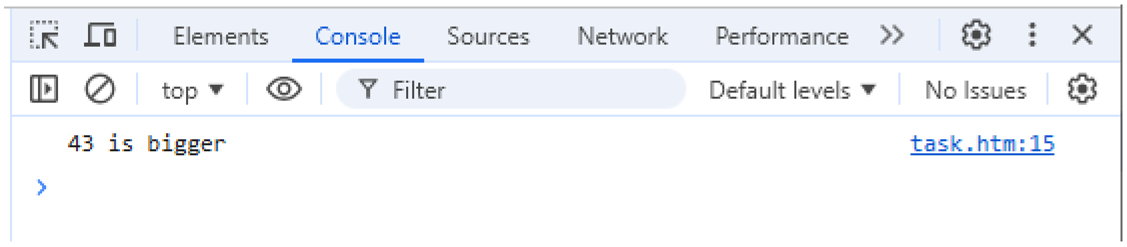


```
false
```

TASK 54

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
</head>
<body>
<script>
let maxValue=(num1,num2)=>{
   if(num1>num2){
      return `${num1} is bigger`;
   }
   else return `${num2} is bigger`;
}
console.log(maxValue(43,6));
</script>
</body>
</html>
```

OUTPUT:



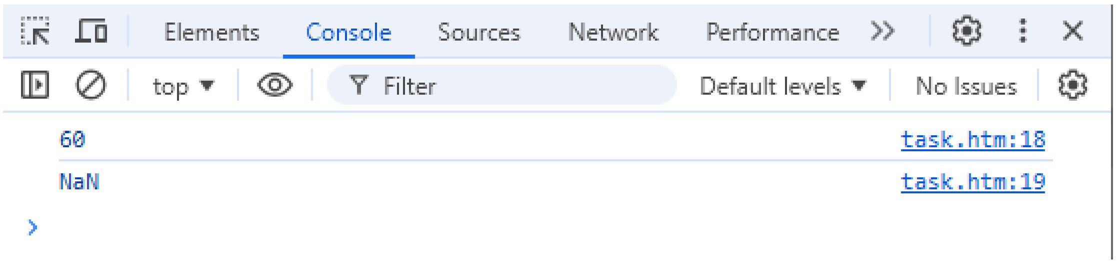
```
43 is bigger
```

TASK 55

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

```
</head>
<body>
<script>
   let myObject={
       value:10,
       multiplyTraditional:function (num){
         return this.value*num;
       },
       multiplyArrow:(num)=>{
          return this.value*num;
       }
   }
console.log(myObject.multiplyTraditional(6));
console.log(myObject.multiplyArrow(10));
</script>
</body>
</html>
```

OUTPUT:



```
60                                                      task.htm:18
NaN                                                     task.htm:19
>
```