UNIT- II

JAVASCRIPT: Introduction to Scripting: Introduction, Simple Program, Modifying Our First Program, Obtaining User Input with prompt Dialogs, Memory Concepts, Arithmetic, Decision Making.

JAVASCRIPT: CONTROL STATEMENTS 1: Introduction, Algorithms, Pseudo Code, Control Structures, if Selection Statement, if…else Selection Statement, while Repetition Statement, Formulating Algorithms: Counter – Controlled Repetition, Sentinel – Controlled Repetition, Nested Control Statements.

JAVASCRIPT: CONTROL STATEMENTS 2: Introduction, Essentials of Counter – Controlled Repetition, for Repetition Statement, Examples Using the for Statement, switch Multiple – Selection Statement, do… while Repetition Statement, break and continue Statements, Labeled break and continue Statements.

JAVASCRIPT

Introduction to scripting:

1. **Introduction:**

JavaScript scripting language, which facilitates a disciplined approach to designing computer programs that enhance the functionality and appearance of web pages. JavaScript serves two purposes- it introduces client-side scripting, which makes web pages more dynamic and interactive, and it provides the programming foundation for the more complex server side developments.

JavaScript contains a standard library of objects, like **Array**, **Date**, and **Math**, and a core set of language elements like **operators**, **control structures**, and **statements**.

- **Client-side:** It supplies objects to control a browser and its Document Object Model (DOM). Like if client-side extensions allow an application to place elements on an HTML form and respond to user events such as **mouse clicks**, **form input**, and **page navigation**.
- **Server-side:** It supplies objects relevant to running JavaScript on a server. Like if the server-side extensions allow an application to communicate with a database, and provide continuity of information from one invocation to another of the application, or perform file manipulations on a server.

2. **Simple Program and Modifying our first program:**

We begin by considering a simple program that displays the text “ Welcome to java script Programming” in the body of an HTML document. Every browser contains java script interpreter, which processes the commands written in java script.

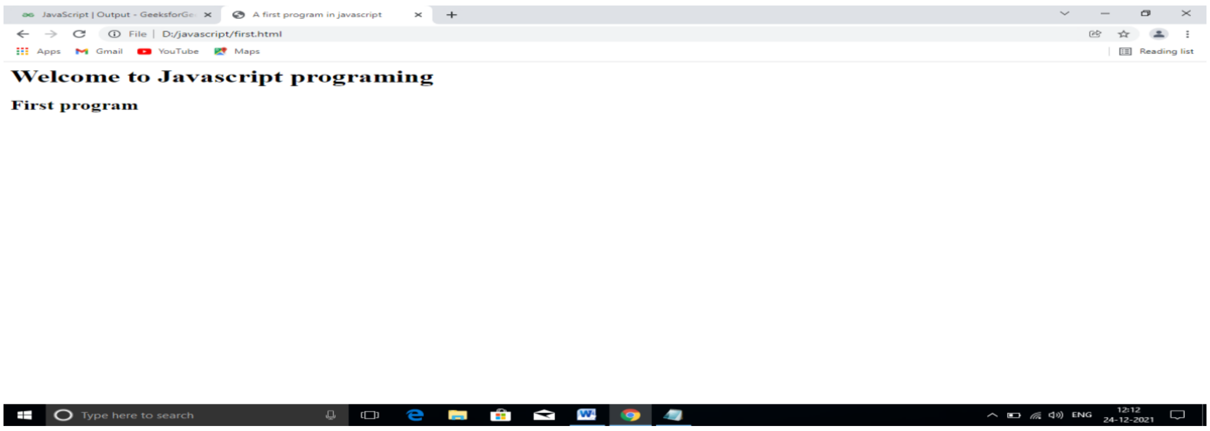
Ex:

```
<html>
<head>
   <title> A first program in javascript</title>
      <script language="javascript">
         document.writeln("<h1> Welcome to Javascript programing</h1>");
      </script>
</head>
<body>
      <h2> First program</h2>
</body>
</html>
```

In above example program, the line 2 indicates the beginning of the <head>section of the HTML document. The line 4 uses the <script> tag to indicate to the browser that the text which follows is part of a script. The language attribute specifies the type of file as well as the scripting languages used in the script. The line 5 instruct the browser's JavaScript interpreter to perform an action, namely, to display in the web page the string of characters contained between the double quotation marks. In the same line 5 use the browser's document object, which represents the HTML document the browser is currently displaying. The document object allows a script programmer to specify text to display in the HTML document.

The</head> tag in line 7 indicates the end of the <head> section. Also in line 8 to 10 , the tags <body> and </body> specify body section in the HTML document. Finally the line 11 indicates the end of the HTML document.

<u>Output:</u>



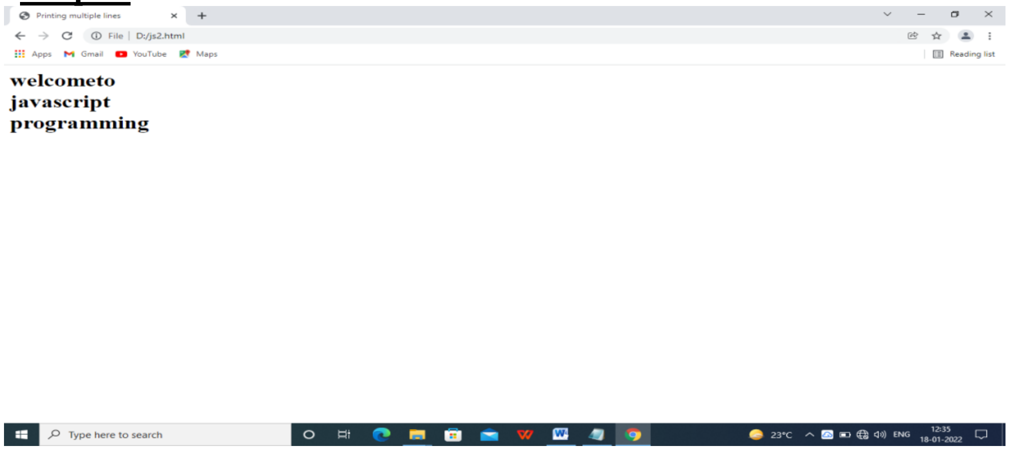
```
Welcome to Javascript programing
First program
```

In the next example, we demonstrate that a single statement that can cause the browser to display multiple lines through by using line-break HTML tags(<br/> throughout the string of HTML text in a write or writeln method call.

<u>Ex:</u>
```
1. <html>
2.<head>
3.<title>Printing multiple lines</title>
4. <script language=" javascript">
5.document.writeln("<h1>welcometo <br/>javascript<br/>programming</h1>");
6.</script>
7. </head>
8.<body></body>
9.</html>
```

In above example, line 5 produce three separate lines of text when the browser renders the HTML document.

Output:



```
welcometo
javascript
programming
```

3. **Obtaining User Input with *prompt* dialogs:**

A script can adapt the content based on input from the user or variables, such as the time of day or the type of browser used by the client. Such web pages are said to be dynamic, as opposed to static, since their content has the ability to change.

❖ **Dynamic welcome page:**

The script uses another predefined dialog box from the window object- a prompt dialog- which allows the user to input a value that the script can use. The program asks the user to input a name , then displays the name in the HTML document.

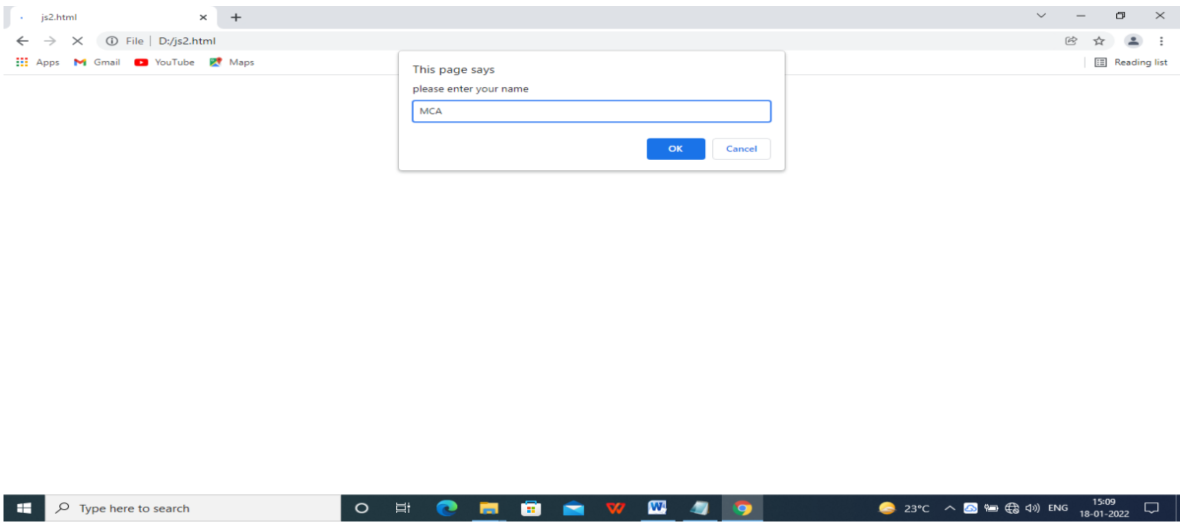
```
1.  <html>
2.  <head>
3.  <script language="javascript">
4.  var name;
5.  name=window.prompt(" please enter your name","MCA");
6.  document.writeln("<h1>Hello " +name+" welcome to java script"</h1>");
7.  </script>
8.  </head>
9.  <body>
10. <p> click refresh to run this script again.</p>
11. </body>
12. </html>
```

In above example, line 4 is a declaration that contains the javascript keyword var. The keyword var at the beginning of the statement that indicates that the word name is a variable. A variable is a location in the computer memory where a value can be stored for use by a program. The name of a variable can be any valid identifier. An identifier is a series of characters consisting of letters, digits, underscores(_) and dollar sign($) that does not begin with a digit and is not a reserved javascript keyword.

The line 6 use document.writeln to display the new welcome message. The expression inside the parentheses uses the operator + to add a string ("<h1>Hello"), the variable name and another string("welcome to javascript programming</h1>"). The result of this operation is a new string.

After the browser interprets the <head> section of the HTML document, it then interprets the <body> of the HTML document and renders the HTML. If you click your browser’s refresh button, the browser will reload the HTML document, so you can execute the script again and change the name.

<u>Output:</u>

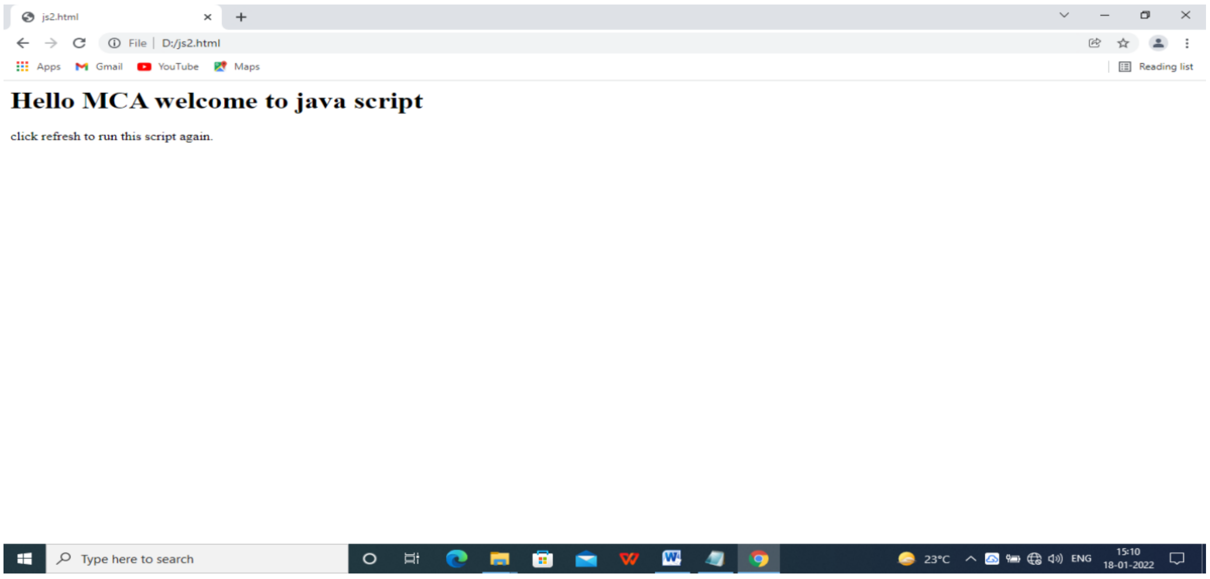


This page says

please enter your name

MCA

OK Cancel



Hello MCA welcome to java script

click refresh to run this script again.

❖ <u>Adding integers:</u>

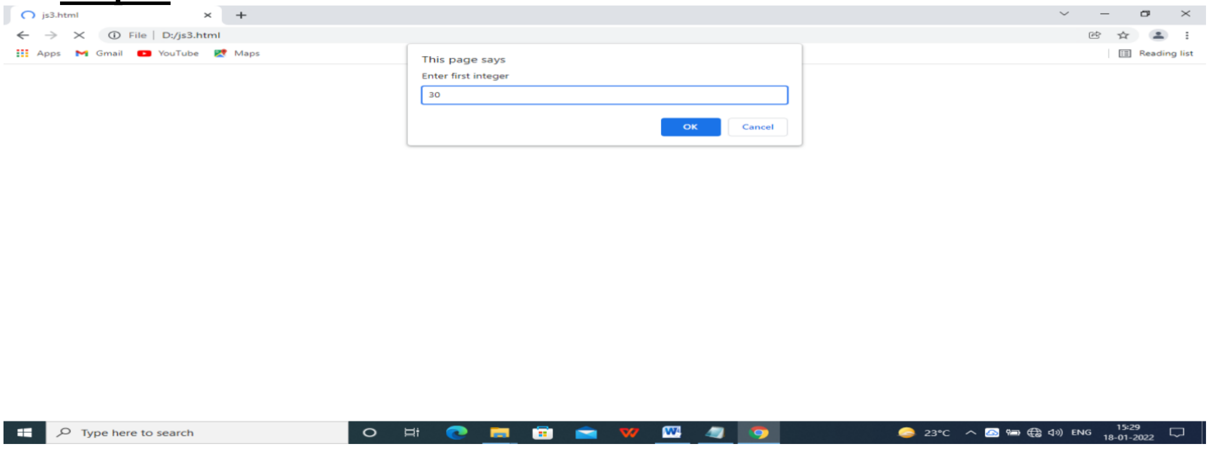
Here illustrates another use of prompt dialogs to obtain input from the user. In below . In below example, inouts two integers typed by a user at the keyboard, computes the sum of the values and displays the result.

<u>Ex:</u>

```
<html>
<head>
<script language=”javascript”>
var firstnumber,secondnumber,number1number2,sum;
firstnumber=window.prompt(“Enter first integer”,”0”);
```

```
secondnumber=window.prompt("Enter second integer","0");
number1=parseInt(firstnumber);
number2=parseint(secondnumber);
sum=number1+number2;
document.writeln("<h1>The sum is:"+sum+"</h1">);
</script>
</head>
<body>
<p>click refesh to run the script again</p>
</body>
</html>
```

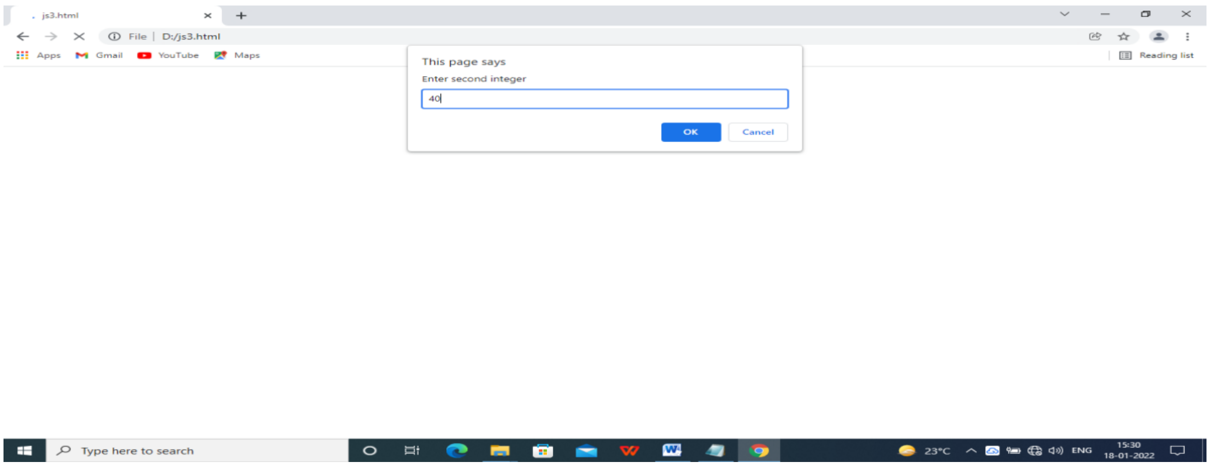
Output:



This page says

Enter first integer

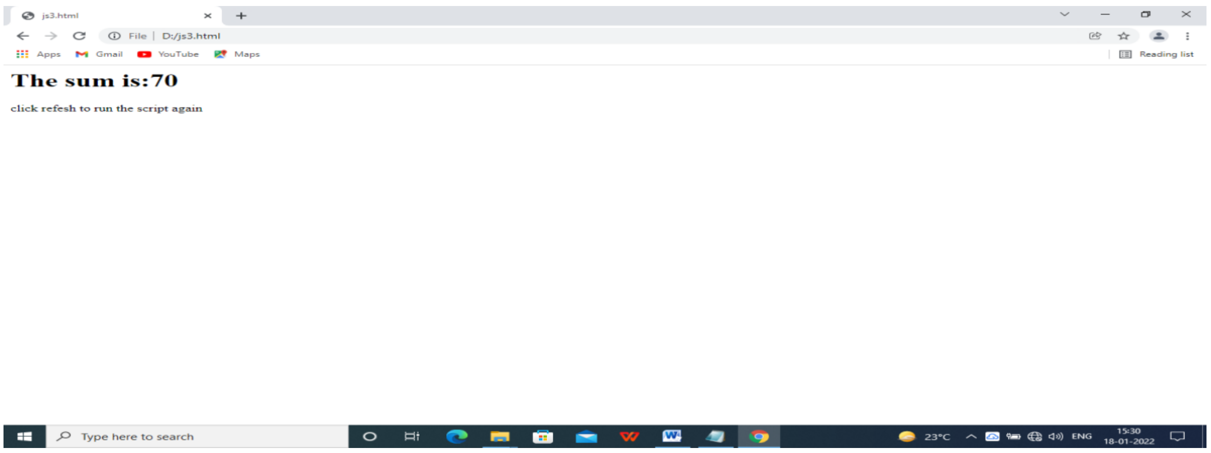
30



This page says

Enter second integer

40



```
The sum is:70
click refesh to run the script again
```

In above example, the line 4 declare the variables firstnumber, secondnumber, number1, number2,sum. Lines 5 and 6, employ a prompt dialog to allow the user to enter a string representing the first of the two integers that will be added. Lines 7 and 8 , converts the two input by the user to integer values that can be used in a calculation. Function parseInt converts its string argument to an integer.

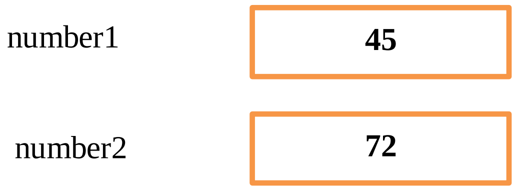
Line 7 and 8, assigns to the variables number1 and number2 , the integer that function pareInt returns. Line 9 calculates the sum of the variables number1 and number2 using the addition operator,=. Line 10 uses document.writeln to display the result of the addition on the webpage. Line11 and 12 close the script and head tags respevtively.Use your browser's refresh button to reload the HTML document and run the script again.

4. <u>Memory Concepts:</u>

In above program variable names such as number1, number2 and sum actually correspond to locations in the computer's memory. Every variable has a name, a type and a value.

In above addition program, when line 7 executes the string firstnumber is converted to an integer and placed into a memory location to which the name number1 has been assigned by the interpreter. Suppose the user entered the string 45 as the value for firstnumber. The program converts firstnumber to an integer, and the computer places the integer value 45 into location number1, as shown in below figure.

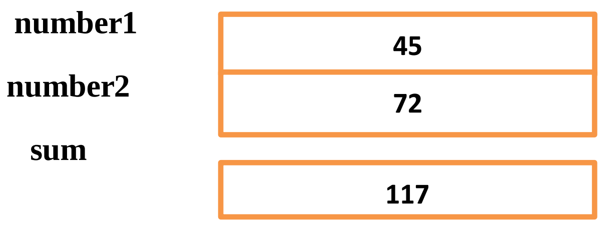
whenever a value is placed in a memory location, the value replaces the previous value in that location. The previous value is lost. When line 8 executes, the program converts secondnumber to an integer, places that integer value 72, into location number2 and the memory appears as shown in below figure.



Once the program has obtained values for number1 and number2, it adds the values and places the sum into variable sum. The statement

sum= number1 + number2

performs the addition and also replaces sum's previous value.Note that the values of number1 and number2 appear exactly as they did before they were used in the calculation of **sum.** These values were used, but not destroyed, when the computer performed the calculation. When a value is read from a memory location , the process is **non-destructive.**



5. Arithmetic:

Many scripts perform arithmetic calculations. The arithmetic operators are binary operators, because each operates on two operands. In javascript arithmetic operators are five types , they are shown below:

Javascript operation	Arithmetic operator	Alzebric expression	Javascript expression
Addition	+	F+7	F + 7
Subtraction	-	p-c	P – c
Multiplication	*	Bm	B * m
Division	/	x/y	X / y
Remainder/Modulo division	%	R mod y	R % y

Arithemtic operators in javascript must be written on straight-line form to facilitate entering programs into the computer. Thus expression such as " a divided by b" must be written as a / b.

Javascript applies the operators in arithmetic expressions in a precise sequence determined by the following **rules of operator precedence**, which are generally the same as those followed in alzebra. Multiplication, division and remainder operations are applied first. If an expression contains several multiplication, division and remainder operations, operators are applied from left to right.Addition and subtraction operations are applied next.

The following is an example of an arithmetic mean of five terms: **Alzebra:** $m = \frac{a+b+c+d+e}{5}$

Javascript: m= (a+b+c+d+e) /5;

Suppose that a, b,c, and x are initialized as follows: a=2,b=3,c=7 and x=5. In below illustrated the order in which the operators are applied in the preceding second degree polynomial.

Ex: **y = (a * x * x)+(b* x) + c;**

Step1: y= 2 * 5 * 5 + 3 * 5 + 7;
2*5 is 10 (left most multiplication)

Step2: y= 10 * 5 + 3 * 5 + 7;
10 * 5 is 50 (left most multiplication)

Step3: y=50 +3 * 5 + 7;
3 * 5 is 15 (Multiplication before addition)

Step4: y=50 + 15 +7;

Operators	**Javascript**	**Sample javascript condition**	**Meaning of javascript condition**
EQUALITY OPERATORS			
= =	= =	x == y	x is equal to y
!=	! =	x != y	x is not equal to y
RELATIONAL OPERATORS			
>	>	x>y	x is greater than y
<	<	x<y	x is less than y
>=	>=	x>=y	x is greater than or equal to y
<=	<=	x<=y	x is less than or equal to y

50 + 15 is 65 (leftmost addition)

Step5: y= 65 + 7;

65 + 7 is 72

Step6: y = 72; (last operation- place 72 into y)

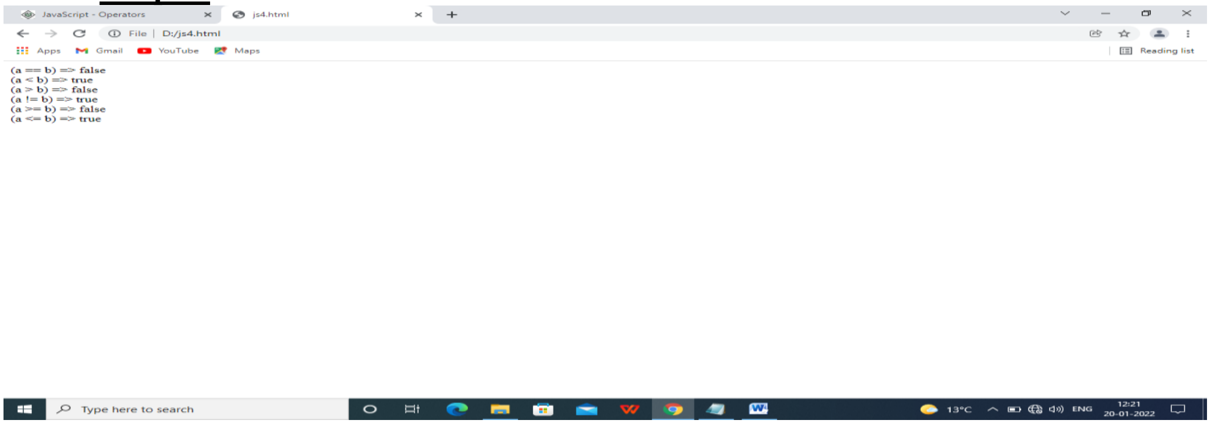
6. <u>Decision making : Equality and Relational operators:</u>

Conditions in *if* statements. Can be formed by using the equality operators and relational operators. The relational operators all have the same level of precedence and associated from left to right. The equality operators both have the same level of precedence., which is lower than the precedence of the relational operators.

Ex:
```
<html>
     <body>
       <script language= "javascript">

           var a = 10;
           var b = 20;
           var linebreak = "<br />";

           document.write("(a == b) => ");
           result = (a == b);
           document.write(result);
           document.write(linebreak);

           document.write("(a < b) => ");
           result = (a < b);
           document.write(result);
           document.write(linebreak);

           document.write("(a > b) => ");
           result = (a > b);
           document.write(result);
           document.write(linebreak);
```

```
                document.write("(a != b) => ");
                result = (a != b);
                document.write(result);
                document.write(linebreak);

                document.write("(a >= b) => ");
                result = (a >= b);
                document.write(result);
                document.write(linebreak);

                document.write("(a <= b) => ");
                result = (a <= b);
                document.write(result);
                document.write(linebreak);
            </script>
        </body>
    </html>
```

Output:



```
(a == b) => false
(a < b) => true
(a > b) => false
(a != b) => true
(a >= b) => false
(a <= b) => true
```

Precedence and associativity of the operators:

Operators	**Associativity**	**Type**
* / %	Left to right	multiplicative
+ -	Left to right	Additive
< <= > >=	Left to right	Relational
== !=	Left to right	equality

JAVASCRIPT: CONTROL STATEMENTS 1

1. Introduction:

Before writing a script to solve a problem, it is essential to have a thorough understanding of the problem and a carefully planned approach to solving the problem. When writing a script, it is equally essential to understand the types of building blocks that are available and to employ proven program construction principles.

2. Algorithms:

Any computing problem can be solved by executing a series of actions in a specific order. A procedure for solving a problem in terms of

- The actions to be executed, and
- The order in which the actions are to be executed.

is called an algorithm. Specifying the order in which statements are to be executed in a computer program is called program control.

For example: a junior executive for getting out of bed and going to work, the steps are: 1. Get out of bed 2.take a shower 3.get dressed 4.eat break fast 5.going to work.. These are the steps are performed in a specific order.

3. Pseudocode:

Pseudo code is an artificial and informal languages that helps programmers develop algorithms. Pseudocode is similar to everyday English; it is convenient and user friendly, although it is not an actual computer programming language.

The style of pseudocode we present consists purely of characters, so that programmers may conveniently type pseudocode in an editor program. Pseudocode normally describes only executable statements- the actions that are performed when the program is converted from pseudocode to javascript and is run. Declarations are not executable statements. For example, the declaration

```
var   value1;
```

instructs the javascript interpreter to reserve space in memory for the variable value1.

4. Control structures:

The research of bohm and jacopini demonstrated that programs could be written without goto statements. The researchers written in terms of only three control structures, namely **the sequence structure, the selection structure, and the repetition structure.**

The sequence structure is built into javascript , Here the computer executes javascript statements one after the other in the order in which they are written(i..e..sequence).

Javascript provides three types of selection structures. They are; if, if…else,switch. The if statement is called a single- selection structure because it selects or ignores a single action. The if…else statement is a double-selection structure because it ignores between two different actions. The switch statement is a multiple-selection structure because it selects among many different actions.

Javascript provides three repetition structure types, namely while, do…while, for. The control structures are attached to one another by connecting the exit point of one control structure to the entry point of the next. This process is similar to the way in which a child stacks building blocks, so we call it control-structure stacking.

❖ If- selection statement:

If the statement is one of the most basic and simplest controls flow statements. We can use the if statement when we want to execute a group of one or more script statements only when a particular condition is met.

Syntax

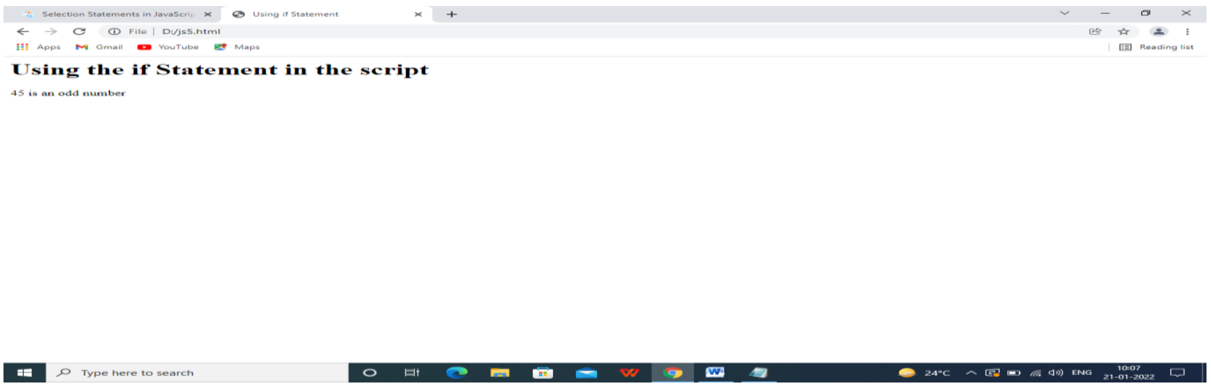
```
if(condition)
{
Statement 1
}
```

In the preceding syntax, if the JavaScript keyword that signifies an if statement and contains a condition, that needs to be evaluated to true, then the script statement, represented by statement 1, enclosed within the curly braces, is executed. If the condition evaluates to false, then the statement enclosed within the curly braces is skipped and the statement immediately after closing curly brace (}) is executed.

Let's understand a simple example of a if statement as in the following:

```
<html>
<head>
    <title>Using if Statement</title>
</head>
<body>
    <h1>
        Using the if Statement in the script
    </h1>
    <script language=”javascript">
      var Number = 45;
        if ((Number % 2) != 0) {
            document.write(Number + " is an odd number");
        }
    </script>
</body>
</html>
```

Output:



```
Using the if Statement in the script
45 is an odd number
```

❖ **The if.....else selection statement:**

As we know, the if statement allows us to execute a set of statements only when a particular condition is true. However, if we want to execute another set of statements when the condition is false, then we can use the if....else statement.

Syntax

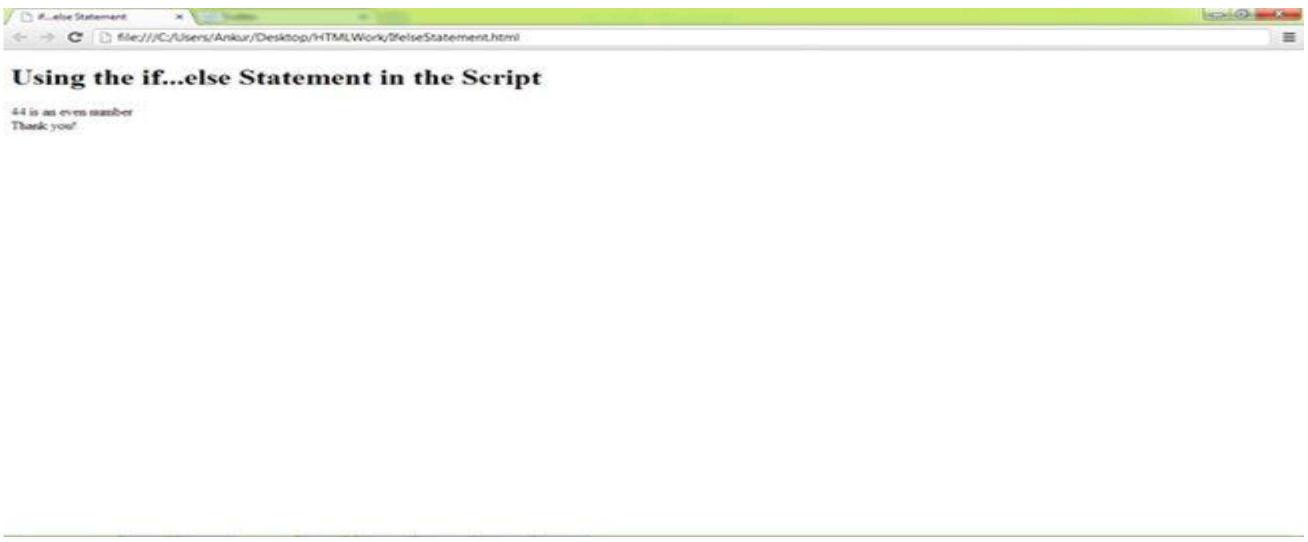
```
if(condition)
{
Statement 1
}
else
{
Statement 2
}
```

In the given syntax, the if and else keywords signify the if...else statement. The condition of the if....else statement is enclosed within parentheses. If the condition is true, then the group of statements represented by statement 1 enclosed within the first curly braces is executed. If the condition is false, then the statement 1 is skipped and the group of statements, represented by statement 2 of the else block is executed.

Let's try a simple if else statement as given below:

```
<html>
<head>
    <title>if...else Statement</title>
</head>
<body>
    <h1>
        Using the if...else Statement in the Script</h1>
    <script language=”javascript">
        var Number = 44;
        if ((Number % 2) != 0) {
            document.writeln(Number + " is an odd number");
        }
        else {
            document.writeln(Number + " is an even  number");
        }
    </script>
</body>
</html>
```

<u>Output:</u>


if...else Statement
file:///C:/Users/Ankur/Desktop/HTMLWork/IfelseStatement.html

Using the if...else Statement in the Script

```
44 is an even number
Thank you!
```

❖ <u>The switch selection statement:</u>

A switch statement selects a particular group of statements to be executed among several other groups of statements. The group of statements that are to be executed is selected on the basis of a numeric or string expression.

<u>Syntax</u>

```
switch(expression)
{
case value1:statement 1
break;
case value2:statement 2
break;
case value3: statement 3
break;
default:statement_default
break;
}
```

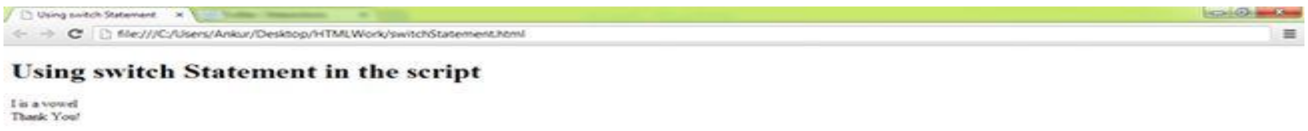
In the given syntax, the switch case, and break are JavaScript keywords. The switch keyword indicates the switch statement. In a switch statement, the expression that is to be evaluated is specified within parentheses. This expression is checked against each of the case values specified in the case statements. If any of the case values match the value of the expression, the group of statements (statement 1, statement 2 or statement 3) specified in the respective case statement is executed.

If none of the case values matches the value of the expression, then the default statement, specified by the default keyword, is executed. The default statement is generally placed at the end of the switch statement; however, we can place it anywhere within the switch statement.

Let's try a simple example of the switch statement as given below:

```
<html>
<head>
    <title>Using switch Statement</title>
</head>
<body>
    <h1>
        Using switch Statement in the script</h1>
    <script language=”javascript">
         var letter = "I";
        switch (letter) {
            default: document.write("consonant");
                break;
            case "A": document.write("A is a vowel");
                break;
            case "E": document.write("E is a vowel");
                break;
            case "I": document.write("I is a vowel");
                break;
            case "O": document.write("O is a vowel");
                break;
            case "U": document.write("U is a vowel");
                break;
        }
    </script>
</body>
</html>
```

Output


Using switch Statement
file:///C:/Users/Ankur/Desktop/HTMLWork/switchStatement.html

```
Using switch Statement in the script

I is a vowel
Thank You!
```

5. Formulating Algorithms: COUNTER-CONTROLLED REPETITION:

Let us use pseudocode to list the actions to execute and specify the order in which the actions should execute. We use counter-controlled repetition to input the grades one at a time. This technique uses a variable called a **counter** to control the number of times a set of statements executes. Counter-controlled repetition often is called **definite repetition,** because the number of repetitions is known before the loop begins executing.

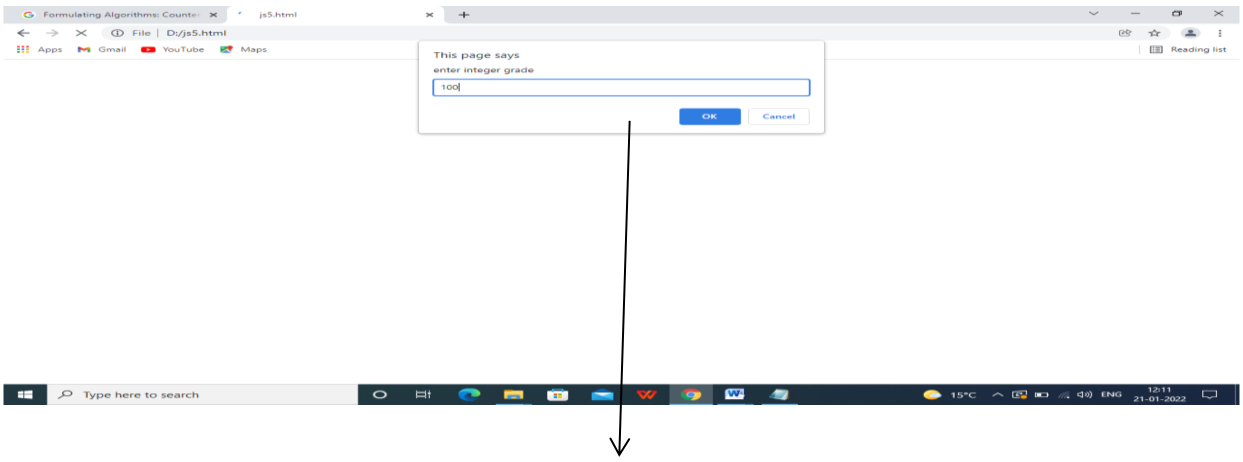
Ex: Pseudocode algorithm that uses counter-controlled repetition to solve the class-average problem:

```
 Set total to zero
 Set grade counter to one

While grade counter is less than or equal to ten
   Input the next grade
   Add the grade into the total
   Add one to the grade counter

Set the class average to the total divided by ten
  Print the class average
```

Ex: program: To calculate a class average:

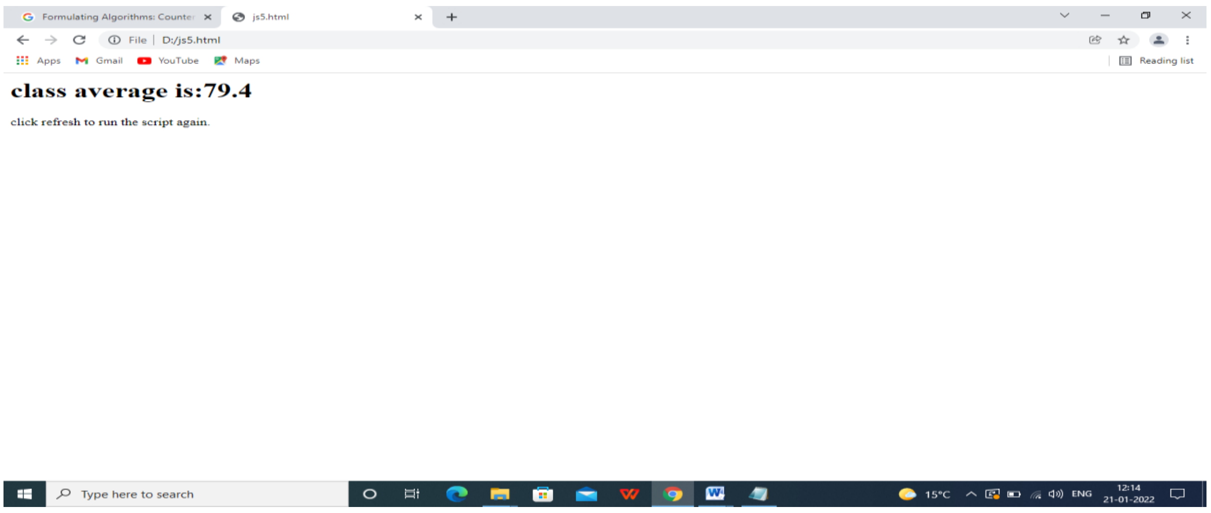
```
<html>
<head>
<script language=”javascript”>
 var total, gradecounter,gradevalue,average,grade;
total=0;
gradecounter=1;
while(gradecounter <= 10)
{
grade=window.prompt(“enter integer grade”.”0”);
gradevalue =parseInt(grade);
total=total + gradevalue;
gradecounter=gradecounter + 1;
}
average=total / 10;
document.writeln(“<h1> class average is:”+average+”</h1>”);
</script>
<body><p> click refresh to run the script again.</body></html>
```

Output:



This page says
enter integer grade
100

This dialog is displayed 10 times user input is 100,88,93,55,68,77,83,95,73,and,62.



```
class average is:79.4
click refresh to run the script again.
```

In above line 4, indicates declare variables total, gradecounter, gradevalue, average and grade. The variable grade will store the string the user types into the prompt dialog. Line 5 and 6 are assignment statements that initialize total to 0 and gradecounter to 1. Line 7 indicates that the while statement continues iterating while ythe value of gradecounter is less than or equal to 10. After the user enters the grade, line 10 converts it from a string to an integer.

Line 11 , adds gradevalue to the previous value of total and assigns the result to total. Line 12 adds 1 to the gradecounter. After this statement executes the program continues by testing the condition in the while statement in line 7 . If the condition is still true, the statements in lines 9 to 12 repeat. Otherwise the program continues execution with the first statement in sequence after the body of the loop. Line 14 assigns the results of the average calculation to variable average. Line 15 displays the string " class average is" followed by the value of the variable average as an <h1> head in the browser…….

6.Nested Control Statements: Formulating Algorithms top down , stepwise refinement:

We will once again formulate the algorithm using pseudocode and top-down , stepwise refinement, and will write a corresponding javascript program.
Consider the following problem statement:

A college offers a course that prepares students for the state licensing for real estate brokers. Last year, several of the students who completed this course took the licensing examination. Naturally, the college wants to know how well its students did on the exam. You have been asked to write a program to summarize the results. You have been given the list of these 10 students. Next to each name is written a 1 if the student passed the exam a 2 if the students failed.

Your program should analyse the results of the exam as follows:

7. Input each test result (i.e., a 1 or a 2). Displays the message "Enter result" on the screen each time the program requests another test result.
8. Count the number of test results of each type.
9. Display a summary of the test results indicating the number of students who passed and the number of students who failed.
10. If more than eight students passed the exam, print the message " Raise tuition".

Pseudocode for above problem statement:

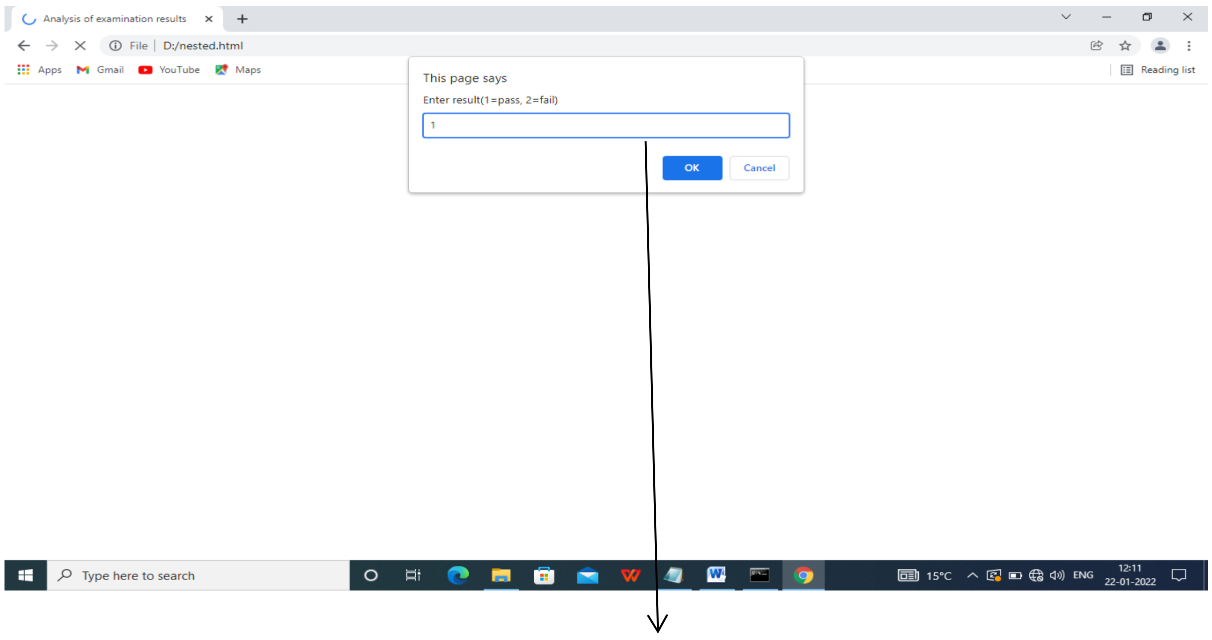
```
Initialize variables
Initialize passes to zero
Initialize failures to zero
Initialize student to one

While student counter is less than or equal to ten
  Input the next exam result
If the student passed
     Add one to passes
Else
      Add one to failures
Add one to student counter

Print the number of passes
Print the number of failures
If more than eight students passed
Print "raise tuition"
```

Ex:

```
1.  <html>
2.  <head>
3.  <title>Analysis of examination results</title>
4.  <script language="javascript">
5.   var passes=0,failures=0,student=1,result;
6.  While(student <= 10) {
7.  result = window.prompt( "Enter result ( 1=pass, 2=fail)","0");
8.  if(result == "1")
9.    passes=passes + 1;
10. Else
11. failures= failures + 1;
12. student=student + 1;
13. }
14. Document.writeln("<h1> Examination Results</h1>");
15. Document.writeln(" passed:" + passes +"<br /> failed: " +failures);
16. If(passes > 8)
17. Document.writeln("<br /> Raie tuiyion");
18. </script>
19. </head>
20. <body>
21. <p>click efresh to run the script again</p>
22. </body>
23. </html>
```

The processing of the exam results occurs in the while statement 6-13 . Note that the if……else statement in lines 8-12 in the loop tests only whether exam result was 1; it assumes that all other exam results are 2.

Output:

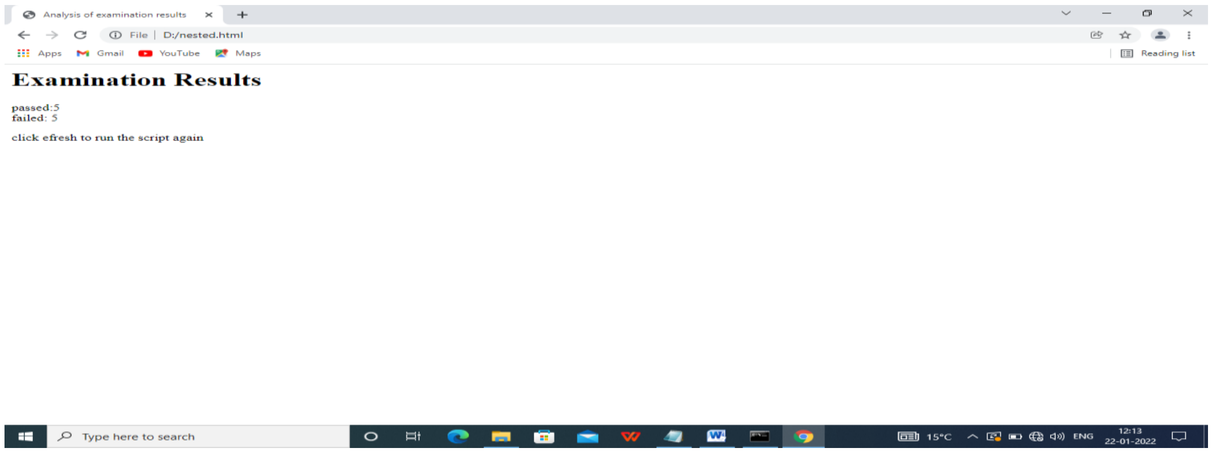


This page says

Enter result(1=pass, 2=fail)

1

This dialog is displayed 10 times. User inputs are 1,2,1,2,2,1,2,2, and 1.



Examination Results

```
passed:5
failed: 5

click efresh to run the script again
```

JAVASCRIPT: CONTROL STATEMENTS 2

❖ Introduction:

Javascript provides three repetition structure types or three counter – controlled repetitions, namely while,do…while, for. The control structures are attached to one another by connecting the exit point of one control structure to the entry point of the next. This process is similar to the way in which a child stacks building blocks, so we call it control-structure stacking.

❖ Essentials of counter-controlled repetition:

Counter-controlled repetitions requires:

- The name of a control variable.
- The initial value of the control variable.
- The increment or decrement by which the control variable is modified each time through the loop.
- The condition that tests for the final value of the control variable to determine whether looping should continue.

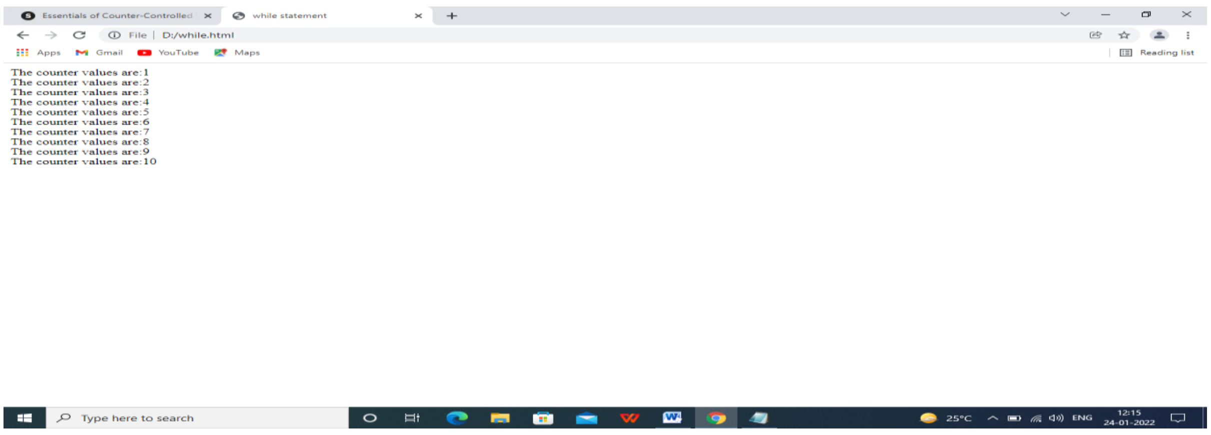
1.While repetition statement:

In the while statement , the loop-continuation test occurs at the beginning of the loop, before the body of the loop executes. The genera format of the while statement is:

Syntax:
```
initialization;
 While ( loop continuation test)
{
    Statement;
    Increment;
}
```

Ex:
```
1. <html>
2. <head>
3.   <title> while statement</title>
4. <script language="javascript">
5. var counter=1;
6.while(counter <=10)
7.{
8.  document.writeln("The counter values are:" +counter+"<br>");
9.counter++;
10. }
11. </script>
12.</head>
13.<body></body>
14.</html>
```

Output:



```
The counter values are:1
The counter values are:2
The counter values are:3
The counter values are:4
The counter values are:5
The counter values are:6
The counter values are:7
The counter values are:8
The counter values are:9
The counter values are:10
```

When the while statement begins executing (line6), the control variable counter is declared and is initialized to 1. Next , the loop continuation condition, counter <=10 is checked . The condition contains the final value (10) of the counter variable. The initial value of the counter is 1. Therefore , the condition is satisfied , so the statement (line 8) is executed. Then the variable counter in the incremented in the expression counter++ and the loop continues execution with the loop continuation test. This process continues until the control variable counter becomes 10 , at which point the loop–continuation test fails and the repetition terminates.

2. for repetition statement :

The for repetition statement handles all the details of counter-controlled repetition. The general format of the for statement is ;

Syntax:

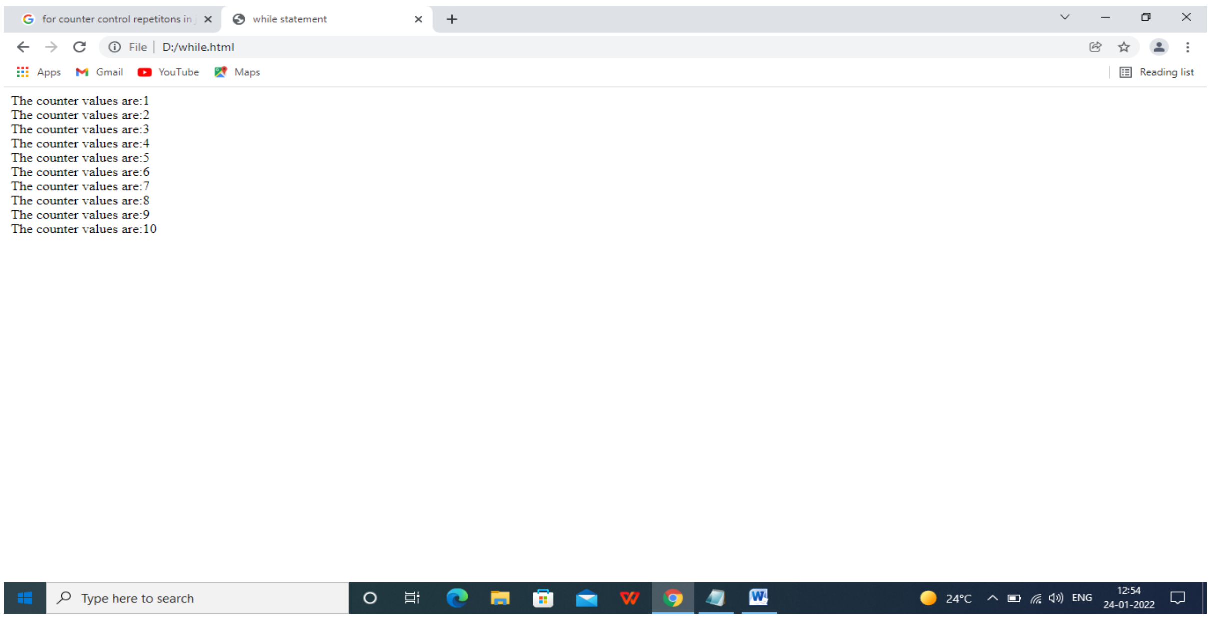
```
for (initialization; loop continuation test; increment/decrement)
  {
     Statement;
  }
```

In above format, the first initialization expression, if the control variable is initialized elsewhere in the program before the loop. Next the loop continuation test expression checks the condition. Finally the increment/ decrement expression , the increments /decrerments are calculated by the statements in the body of the for statement.

Ex:

```
1. <html>
2. <head>
3.   <title> while statement</title>
4. <script language="javascript">
5. for( var counter=1; counter<=10; counter++)
6. {
8.  document.writeln("The counter values are:" +counter+"<br>");
10. }
11. </script>
12.</head>
13.<body></body>
14.</html>
```

Output:



```
The counter values are:1
The counter values are:2
The counter values are:3
The counter values are:4
The counter values are:5
The counter values are:6
The counter values are:7
The counter values are:8
The counter values are:9
The counter values are:10
```

When the for statement begins executing (line 5), the control variable counter is declared and is initialized to 1. Next , the loop continuation condition, counter <=10 is checked . The condition contains the final value (10) of the counter variable. The initial value of the counter is 1. Therefore , the condition is satisfied , so the statement (line 8) is executed. Then the variable counter in the incremented in the expression counter++ and the loop continues execution with the loop continuation test. This process continues until the control variable counter becomes 10 , at which point the loop–continuation test fails and the repetition terminates.

3. **<u>Examples using the for statement:</u>**

The examples in this section show methods of varying the control variable in a for statement. In each case, we write the appropriate for header. Note the change in the relational operator for loops that decrement the control variable.

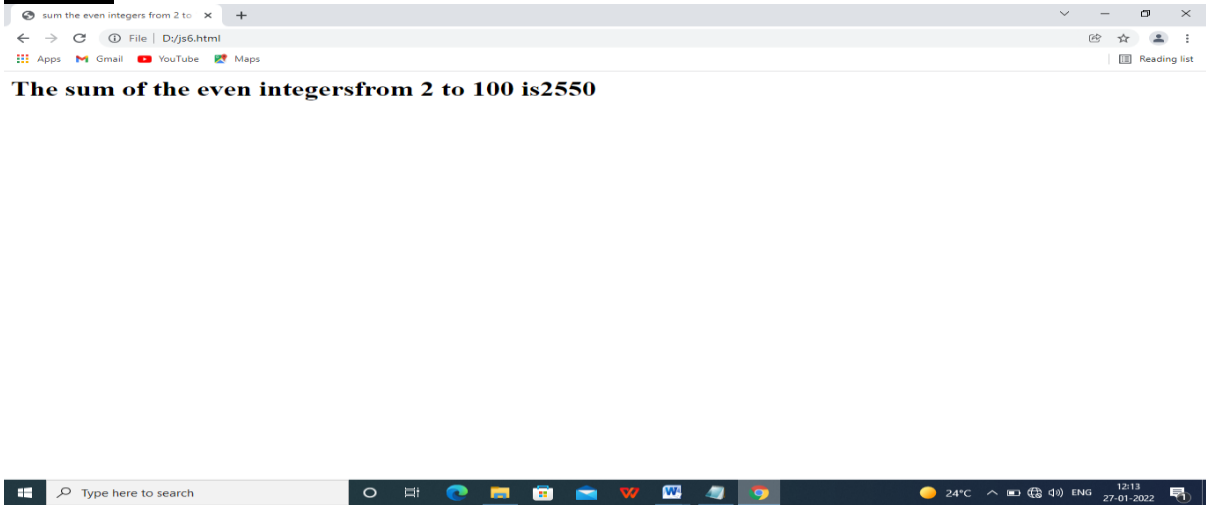
- Vary the control variable from 1 to 100 in increments of 1.
  for (var i= 1; I <= 100; ++i)
- Vary the control variable from 1 to 100 in decrements of 1.
  for (var i=100; i >=1; --i)
- Vary the control variable from 7 to 77 in steps of 7.
  for (var i =7 ; I <= 77; +=7)

  The below example, uses the for statement to sum the even integers from 2 to 100. Note that the increment expression adds 2 to the control variable number after the body executes during each iteration of the loop.

<u>Ex:</u>

```
  <html>
 <head>
     <title> sum the even integers from 2 to 100</title>
      <script language="javascript">
       var sum=0;
        for ( var number =2;number <= 100; number += 2)
        sum += number;
        document.writeln(" <h1>The sum of the even integers" + "from 2 to 100 is"
            +sum+"</h1>");
      </script>
    </head><body></body>
</html>
```

<u>Output:</u>



```
The sum of the even integersfrom 2 to 100 is2550
```

4. The do… while repetition statement:

The do……while repetition statement is similar to the while statement. In the while statement, the loop continuation test occurs at the beginning of the loop, before the body of the loop executes. The do…while statement tests the loop-continuation condition after the loop body executes; therefore , the loop body always executes at least once.

Syntax: initialization;

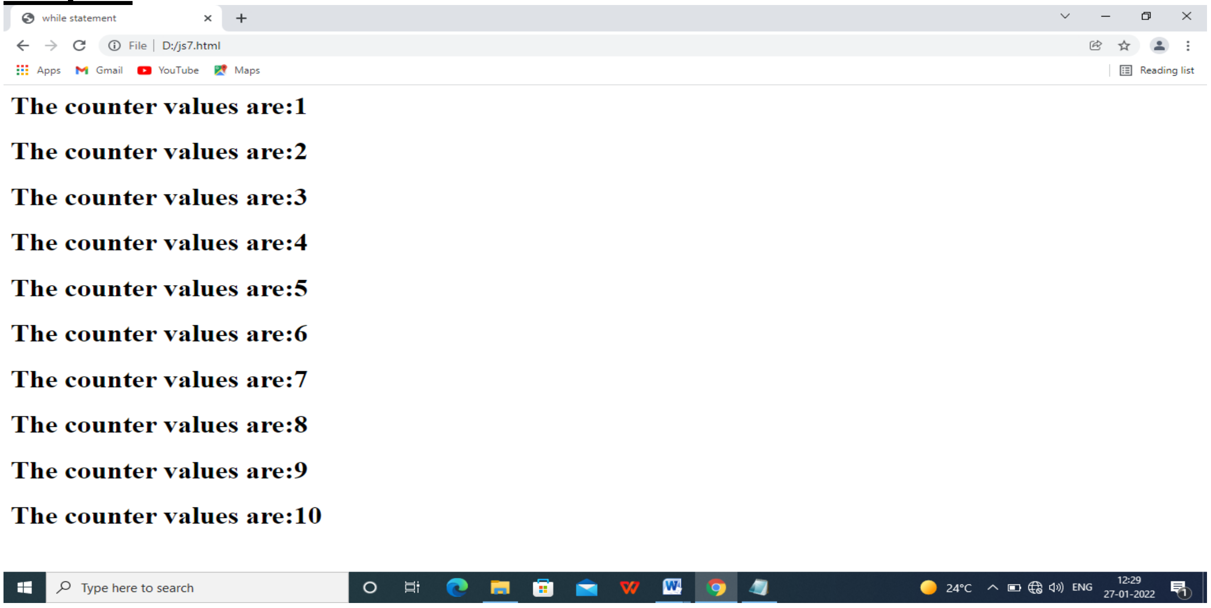
```
do
    {
        Statement;
         Increment;
     } while ( condition);
```

When a do…while terminates, execution continues with the statement after the while clause. Note that it is not necessary to use braces in a do…while statement if there is only one statement in the body.

Ex:

```
<html>
  <head>
     <title> while statement</title>
   <script language="javascript">
   var counter=1;
 do
  {
   document.writeln("The counter values are:" +counter+"<br>");
   counter++;
   } while(counter <=10);
  </script>
 </head>
 <body></body>
 </html>
```

Output:



```
The counter values are:1
The counter values are:2
The counter values are:3
The counter values are:4
The counter values are:5
The counter values are:6
The counter values are:7
The counter values are:8
The counter values are:9
The counter values are:10
```

The do-while loop is a control flow statement that executes the code block at least once and then it executes the code block repeatedly, depending on the condition given at the end of the code block.

The do-while loop is also known as **"post test loop"** as in this loop, condition is checked after the execution of loop once. There are some cases, where we want to execute the code block infinite times, thus creating an **infinite loop**. When such type of loop is created, the break statement is used that allows the termination from the loop.

5. Break and continue statements:

❖ BREAK statement:

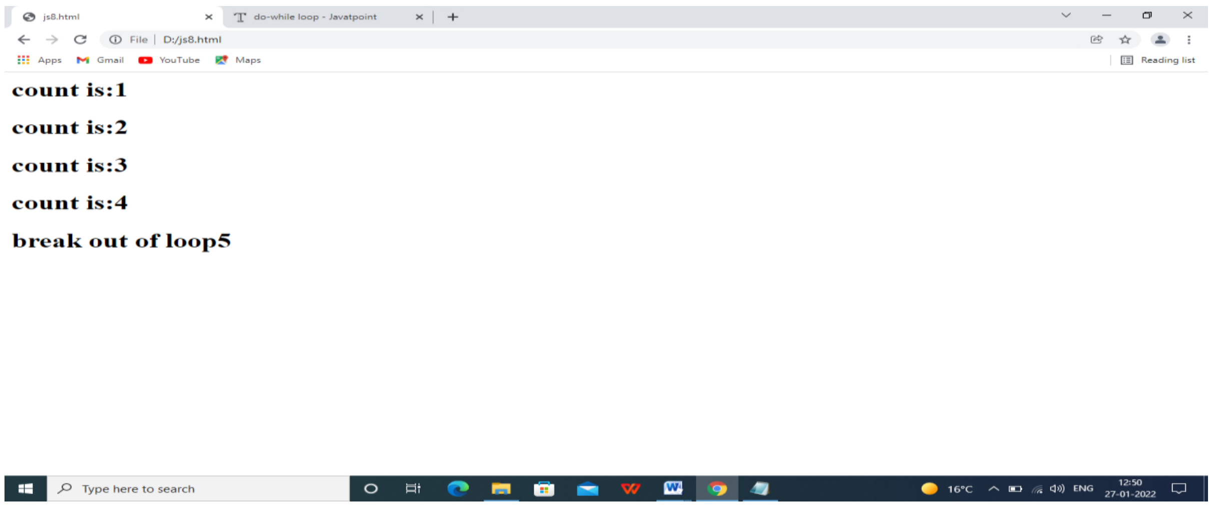
The **break** and **continue** statements alter the flow of control. The break statement, when executed in a while , for, do…while or switch statement, causes immediately exit from the statement. Execution continues with the first statement after the structure. The break statement is commonly used to escape early from a loop or to skip the remainder of a switch statement.

Syntax: **break;**

Ex:

```
<html>
<head>
<script language=”javascript”>
for( var count=1; count <=5; count++)
{
if(count == 5)
break;
document.writeln(“ count is:”+count+”<br>”);
}
document.writeln( “ break out of loop” +count);
</script>
</head><body></body></html>
```

Output:



```
count is:1
count is:2
count is:3
count is:4
break out of loop5
```

In above example, when the if statement in line 6 detects that count is 5, the break in line 7 executes. This statement terminates the for statement, and the program proceeds to line 10, where the script writes the value of count when the loop terminated. The loop executes its body only four times.

❖ <u>CONTINUE statement:</u>

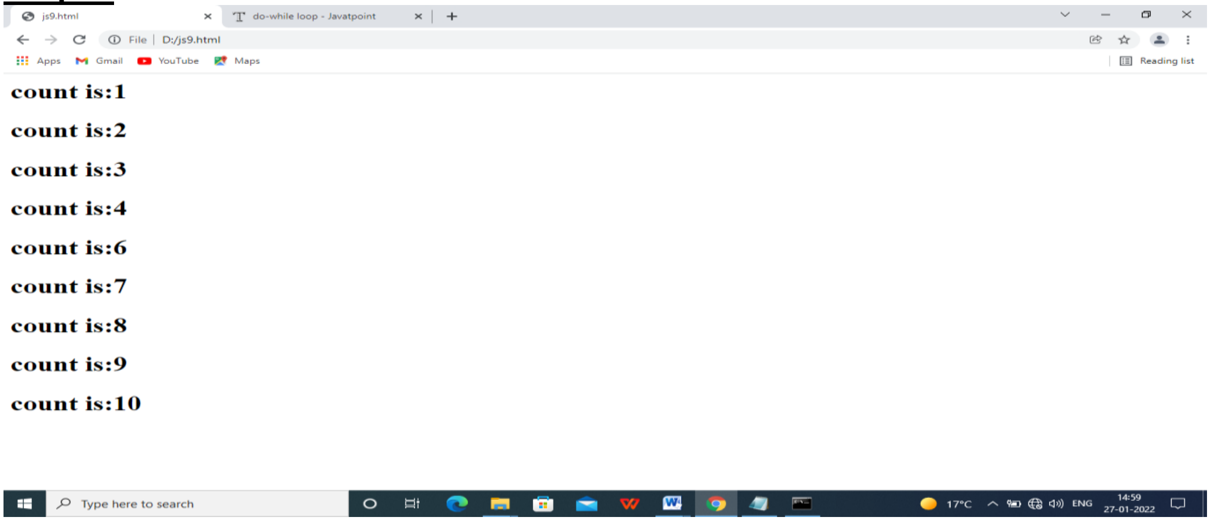
The continue statement , when executed in a while, for or do….while statement, skips the remaining statements in the body of the statement and proceeds with the next iteration of the loop. In while and do…while statements, the loop continuation test evaluates immediately after the continue statements executes.

<u>Syntax:</u> continue;

<u>Ex:</u>

```
1.<html>
2.<head>
3.<script language="javascript">
4.for(var count=1; count<=10;count++)
5.{
6.if (count ==5)
7.continue;
8.document.writeln("count is:" +count+"<br>");
9.</script>
10.</head><body></body></html>
```

<u>Output:</u>



```
count is:1
count is:2
count is:3
count is:4
count is:6
count is:7
count is:8
count is:9
count is:10
```

In above example uses a continue in a for statement to skip the document.writeln statement in line 8 when the if statement in line 6 determines that the value of count is 5. When the continue statement executes, the script skips the remainder of the for statement's body. Program control continues with the increment of the for statement's control variable, followed by the loop-continuation test to determine whether the loop should continue executing.

6. Labeled BREAK and CONTINUE statements:

❖ Labeled BREAK statement:

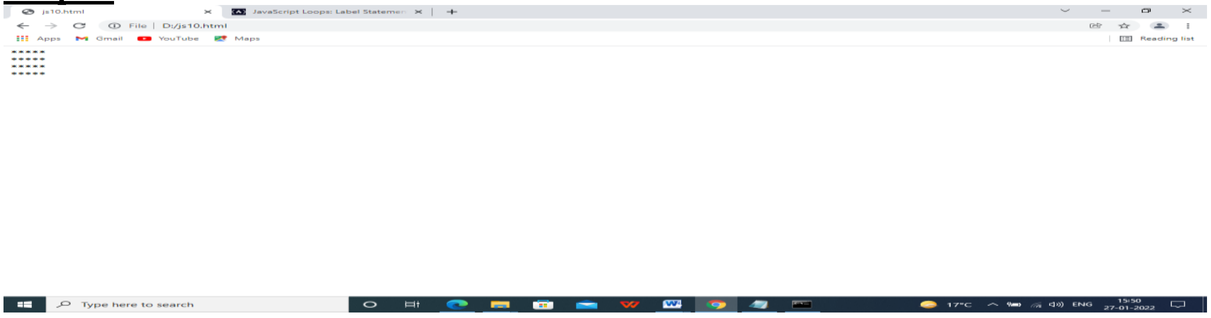
The break statement can break out of an immediately enclosing while, for, do…while or switch statement. To break out of a nested set of structures, you can use the labelled break statement. This statement, when executed in a while,for, do…while or switch statement, causes immediate exit from that statement and any number of enclosing repetition statements; program execution resumes with the first statement afyter the enclosing labeled statement.

Syntax: break label;

Ex:

```
1.<html>
2.<head>
3.<script language='javascript">
4.stop: {
5.for(var row=1; row <= 10; row++){
6. for(var coloumn=1; coloumn <= 5;coloumn++){
7.if(row==5)
8.break stop;
9.document.write(" *");
10.}
11. document.writeln("<br>");
12.}}
13.</script>
14.</head><body></body></html>
```

Output:



```
*****
*****
*****
*****
```

The label block begins(line 4-12)with a label **stop**. And includes both the nested for statement starting in line 5 and the document.writeln statement in line12. When the line 7 if statement detects that row is equal to 5, the statement in line 8 executes. This statement terminates both the for statement in line 6 and its enclosing for statement in line 5, and the program proceeds to the statement in line 12. The inner for statement executes its body only four times. Because it is included in the labelled block and the outer block for statement never completes.

❖ Labeled CONTINUE statement:

The labeled continue statement when executed in a repetition statement (while,for or do..whike), skips the remaining statements in the structure's body and any number of enclosing repetition statements, then proceeds with the next iteration of the enclosing labeled repetition statement.

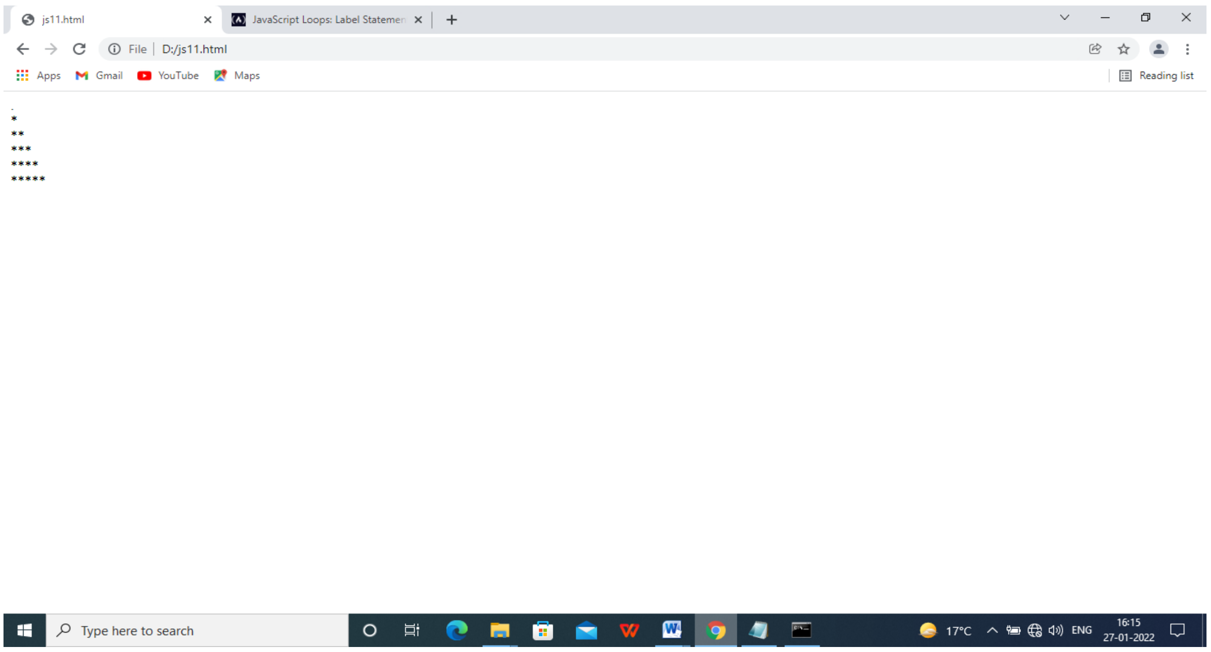
In labeled while and do..while statements, the loop continuation test evaluates immediately after the continue statement executes. In a labeled for statement, the increment expression executes , then the loop continuation tests evaluates.

Syntax: continue label;

Ex:

```
1.<html>
2.<head>
3.<script language"javascript">
4.nextrow:
5.for(var row=1; row <= 5; row++){
6. document.writeln("<br>");
7. for(var coloumn=1; coloumn <= 10;coloumn++){
8.if(coloumn > row)
9.continue nextrow;;
10.document.write(" *");
11.}
12. }
13.</script>
14.</head><body></body></html>
```

Output:



```
*
**
***
****
*****
```

In above example, the labeled for statement (lines 4-12) starts with the nextrow label. When the if statement in line 8 in the inner for statement detects that coloumn is greater than row. Line 9 executes and program control continues with the increment of the control variable of the outer for loop. Even though the inner for statement counts from 1 to 10. The number of * characters output on a row never exceeds the value of row.