3. JavaScript Controls

JS Conditions: If , Else If & Switch, JS Loop: For, For In, While & Do While, JS Break, JS Type Conversion, JS Errors: Try, Catch & Throw

JavaScript If-else

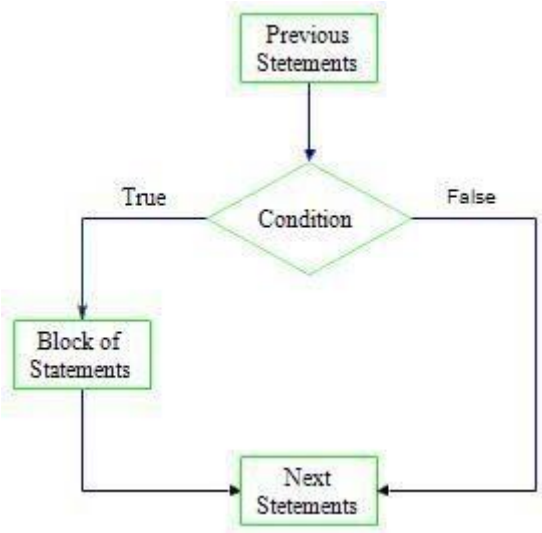
The **JavaScript if-else statement** is used *to execute the code whether condition is true or false*. There are three forms of if statement in JavaScript.

1. If Statement
2. If else statement
3. if else if statement

JavaScript If statement

It evaluates the content only if expression is true. The signature of JavaScript if statement is given below.

```
if(expression){
//content to be evaluated
}
```



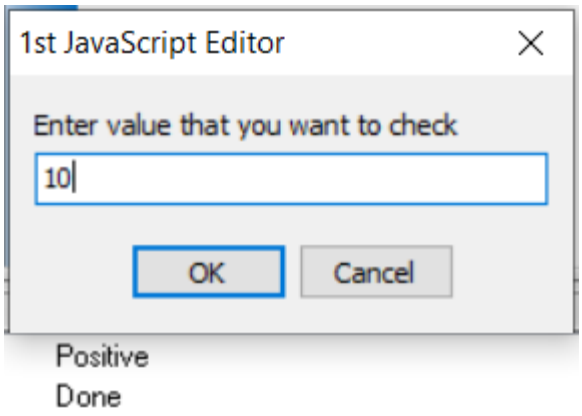
Example:

AIM: Write a javascript code to check number is positive or not.

File: if_demo.js

```
var x = prompt("Enter value that you want to check", "");

if(x>=0)
{
   document.writeln("Positive");
}
document.writeln("Done");
```

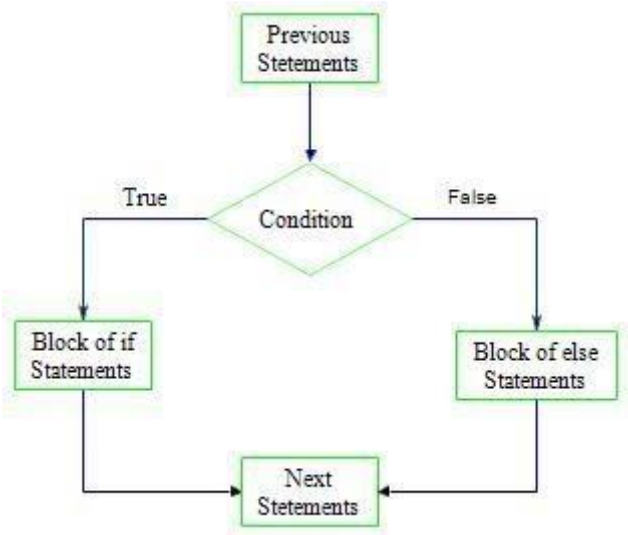
Output:



JavaScript If...else Statement

It evaluates the content whether condition is true of false. The syntax of JavaScript if-else statement is given below.

```
if(expression){
        //content to be evaluated if condition is true
}
else{
        //content to be evaluated if condition is false
}
```



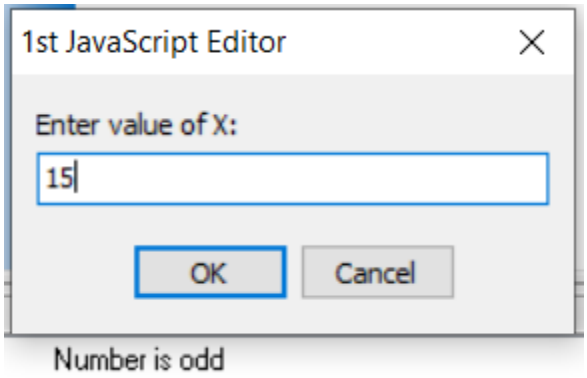
AIM: Write a javascript code to check entered number is odd OR even.

FILE: if_else_demo.js

```
var x = prompt("Enter value of X:", "")

if(x % 2 == 1)
{
   document.write("Number is odd");
}
else
{
```

```
   document.write("Number is even");
}
```

OUTPUT:



JavaScript If...else if statement

It evaluates the content only if expression is true from several expressions. The signature of JavaScript if else if statement is given below.

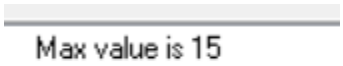
```
if(expression1){
        //content to be evaluated if expression1 is true
}
else if(expression2){
        //content to be evaluated if expression2 is true
}
else if(expression3){
        //content to be evaluated if expression3 is true
}
else{
        //content to be evaluated if no expression is true
}
```

AIM: Write a javascript code to find out maximum from 3 numbers.
FILE: if_elseif_demo.js

```
var x = 10;
var y = 15;
var z = 12;

if(x >=y && x >= z)
{
   document.writeln("Max value is "+x);
}
else if(y>=x && y>=z)
{
   document.writeln("Max value is "+y);
}
```

```
else
{
   document.writeln("Max value is "+z);
}
```

OUTPUT:



JavaScript Switch

The JavaScript switch statement is used to execute one code from multiple expressions. It is just like else if statement. But it is convenient than if..else..if because it can be used with numbers, characters etc.

The signature of JavaScript switch statement is given below.

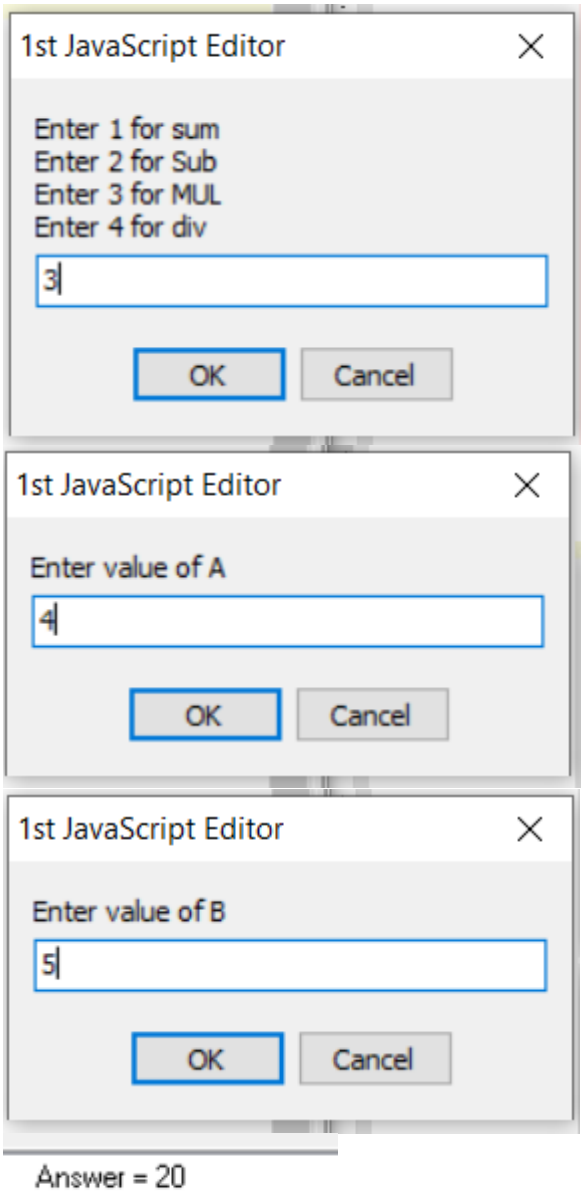
```
switch(expression){
        case value1:
                code to be executed;
                break;
        case value2:
                code to be executed;
                break;
        ......

        default:
                code to be executed if above values are not matched;
}
```

AIM: Write a javascript code to implement calculator.

```
var x = prompt("Enter 1 for sum \nEnter 2 for Sub\nEnter 3 for MUL \nEnter 4 for div", "");
x = parseInt(x);

if(x>=1 && x<=4)
{
   var a = prompt("Enter value of A", "");
   a = parseInt(a);
   var b = prompt("Enter value of B", "");
   b = parseInt(b);
}
var ans;

switch(x)
{
   case 1:
      ans = a + b;
      break;
```

```
    case 2:
        ans = a - b;
        break;
    case 3:
        ans = a * b;
        break;
    case 4:
        ans = a / b;
        break;
    default:
        document.write("You have entered wrong value")
}

document.write("Answer = "+ans);
```

OUTPUT:



JavaScript Loops

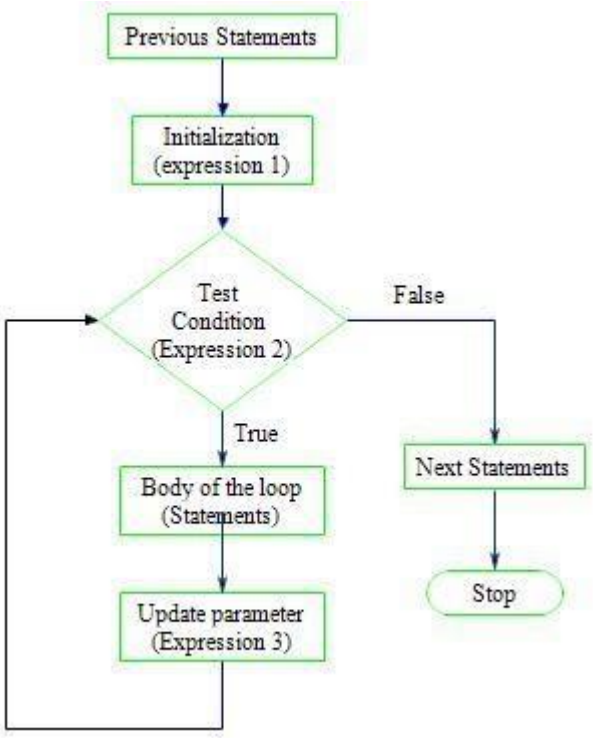
The **JavaScript loops** are used *to iterate the piece of code* using for, while, do while or for-in loops. It makes the code compact. It is mostly used in array.
There are four types of loops in JavaScript.

1. for loop
2. while loop
3. do-while loop
4. for-in loop

1) JavaScript For loop

The JavaScript for loop iterates the elements for the fixed number of times. It should be used if number of iteration is known. The syntax of for loop is given below.

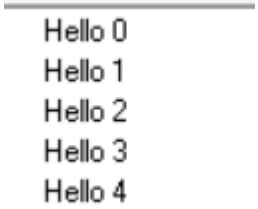
```
for (initialization; condition; increment)
{
   code to be executed
}
```



AIM: Write a JavaScript code to display message 5 times using for loop.

```
var i;
for(i=0;i<5;i++)
{
   document.writeln("Hello "+i);
}
```

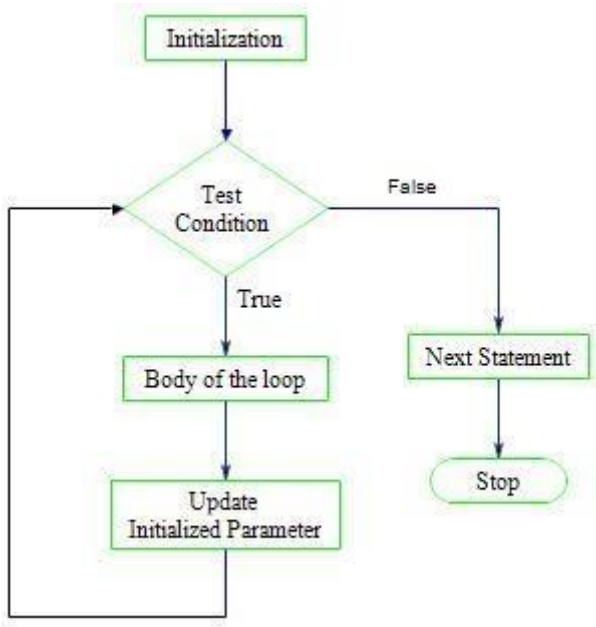
OUTPUT:



2) JavaScript while loop

The JavaScript while loop iterates the elements for the infinite number of times. It should be used if number of iteration is not known. The syntax of while loop is given below:

```
while (condition)
{
   code to be executed
}
```



AIM: Write a JavaScript code to demonstrate the use of while loop.

```
var i=11;
while(i<=15)
{
   document.writeln(i);
   i++;
}
```

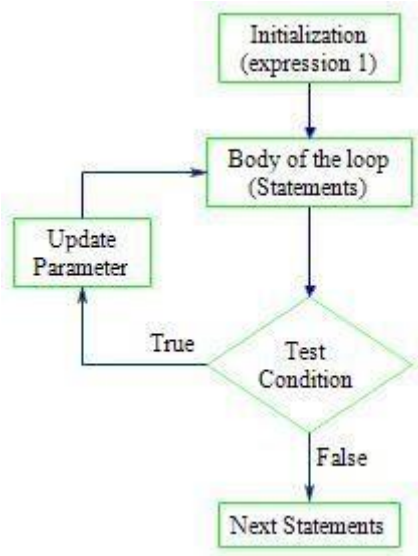
OUTPUT:

```
11
12
13
14
15
```

3) JavaScript do while loop

The JavaScript do while loop iterates the elements for the infinite number of times like while loop. But, code is executed at least once whether condition is true or false. The syntax of do while loop is given below.

```
do{
   code to be executed
}while (condition);
```



AIM: Write a JavaScript code to demonstrate the use of do while loop.

```
var i=11;
do
{
   document.writeln(i);
   i++;
}while(i<=15);
```

OUTPUT:

```
11
12
13
14
15
```

JS Break

The break statement "jumps out" of a loop.

The continue statement "jumps over" one iteration in the loop.

The Break Statement
The break statement can also be used to jump out of a loop. The break statement breaks the loop and continues executing the code after the loop (if any):

Example:

```
for(i=0;i<10;i++)
{
   if(i==3)
   {
```

```
        break;
    }
    document.writeln("Hello "+i);
}
```

OUTPUT:

```
Hello 0
Hello 1
Hello 2
```

The Continue Statement

The continue statement breaks one iteration (in the loop), if a specified condition occurs, and continues with the next iteration in the loop.

Example:

```
for(i=0;i<10;i++)
{
    if(i==3)
    {
        continue;
    }
    document.writeln("The number is "+i);
}
```

OUTPUT:

```
The number is 0
The number is 1
The number is 2
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
```

JS Type Conversion:

Type conversion is the ability to convert from one data type to another. TypeScript provides us with built-in functions to perform type conversion.

JavaScript variables can be converted to a new variable and another data type:

- By the use of a JavaScript function
- Automatically by JavaScript itself

Converting Numbers to Strings

The global method String() can convert numbers to strings. It can be used on any type of numbers, literals, variables, or expressions:

Example

```
String(x)        // returns a string from a number variable x
String(123)      // returns a string from a number literal 123
String(100 + 23)  // returns a string from a number from an expression
```

The Number method **toString()** does the same.

Example

```
x.toString()
(123).toString()
(100 + 23).toString()
```

In the similar way, Boolean and date type of data can also be converted into string.

Converting Strings to Numbers

The global method Number() can convert strings to numbers.

Automatic Type Conversion

When JavaScript tries to operate on a "wrong" data type, it will try to convert the value to a "right" type.

The result is not always what you expect:

```
5 + null        // returns 5          because null is converted to 0
"5" + null      // returns "5null"    because null is converted to "null"
"5" + 2         // returns "52"       because 2 is converted to "2"
"5" - 2         // returns 3          because "5" is converted to 5
"5" * "2"       // returns 10         because "5" and "2" are converted to 5 and 2
```

JS Errors: Try, Catch & Throw

There are three types of errors in programming: (a) Syntax Errors, (b) Runtime Errors, and (c) Logical Errors. No matter how great we are at programming, sometimes our scripts have errors. They may occur because of our mistakes, an unexpected user input, an erroneous server response, and for a thousand other reasons. There's a syntax construct try..catch that allows to "catch" errors and, instead of dying.

The try statement lets you test a block of code for errors.
The catch statement lets you handle the error.
The throw statement lets you create custom errors.
The finally statement lets you execute code, after try and catch, regardless of the result.

```
<html>
  <head>

    <script type = "text/javascript">
        function myFunc() {
          var a = 100;
          var b = 0;

          try {
```

```
            if ( b == 0 ) {
              throw( "Divide by zero error." );
            } else {
              var c = a / b;
            }
          }
          catch ( e ) {
            alert("Error: " + e );
          }
        finally {
            alert("Finally block will always execute!" );
          }
        }
    </script>

  </head>
  <body>
    <p>Click the following to see the result:</p>

    <form>
      <input type = "button" value = "Click Me" onclick = "myFunc();" />
    </form>

  </body>
</html>
```

EXERCISE:

1. **Write an "if" condition to check that age is between 14 and 90 inclusively. ("Inclusively" means that age can reach the edges 14 or 90.)**

2. **Write a program to enter one character from user and display message "Your Character is Vowel", if it's a vowel. (Note: Example of if else Statement).**

3. **Write the code which asks for a login with prompt. If the visitor enters "Admin", then prompt for a password, if the input is an empty line or Esc – show "Canceled.", if it's another string – then show "I don't know you".**

   The password is checked as follows:
   - **If it equals "TheMaster", then show "Welcome!",**
   - **Another string – show "Wrong password",**
   - **For an empty string or cancelled input, show "Canceled."**

Please use nested if blocks. Mind the overall readability of the code.

Hint: passing an empty input to a prompt returns an empty string ''. Pressing ESC during a prompt returns null.

4. **Write a program to take number of unit consumed in a month and calculate the electricity bill by using charges specified as below:**

Unit	0 - 100	101 - 200	201 - 300	301– 450	> 450
Charge	1.5	2.5	3.0	4.5	6

5. What is the output? Why?

```
let i = 3;
  while (i)
  {
        alert( i-- );
  }
  alert(i);
```

6. For every loop iteration, write down which value it outputs. Both loops alert the same values, or not?

```
let i = 0;

while (++i < 5)
        alert( i );
```

```
let i = 0;

while (i++ < 5)
        alert( i );
```

7. Write the code which outputs prime numbers in the interval from 2 to n.

8. Write a code to check entered number is Armstrong or not.