1.What is Javascript?

Ans. It is a High level multi-paradigm language that supports

Dynamic, event-driven, functional, imperative, object oriented and prototype based programming styles.

- It was developed by Brenden Eich at Netscape Communication in year 1995.

- Javascript is also known as Mocha, LiveScript, Jscript, ECMAScript.

USES

- It was primarily used as Client Side Scripting language to develop dynamic web applications.
- Mostly all major Web browser have dedicated javascript engine embedded in them.
- Javascript is also Used for server side scripting ,(i.e) the javascript in also implemented in web server .
- Different major web browser have different javascript engines.

Browser	Javascript Engines
Chrome	V8
Internet Explorer	Chakra
Mozilla Firefox	Spider monkey
Safari	Javascript Core

2. What is ECMA?

Ans. ECMA stands for "European Computer Manufactures Association",

It an organization which releases standard for product.

3. What is ECMAScript?

Ans. ECMAScript is a Standards specification which is defined in ECMA-262 for creating scripting language.

Javascript follows ECMAScript standards.

4. What are Javascript REPL tool?

Ans. REPL- stands for "Read-Evaluate-Print-Loop" it is an environment that allows user to write and execute their code , it is similar to shell and web console.

Javascript REPL tools

1.JSFiddle

2.JsBin

3.Plunker

5. What are Keyword in Javascript ?

Ans. All preserved (or) already predefined word in javascript are keywords.

Ex – var , let , const, function, Object ,in …..

6. What is an Identifier ?

Ans. Any name given to define a particular(object,function,variable,array) in javascript , are called as identifiers.

Identifier cannot be javascript keywords doing so will override the predefined value.

- Identifiers can include

  – Letters (Unicode)

  – Digits [0-9]

  – Underscore '_'

  – Dollar '$'

  Identifiers

  – Can begin only with a letter or an underscore.

  – Cannot be a JavaScript keyword

Identifiers

- Variables / functions names: use camelCase
- Small letters are considered different than the capital letters
- Should have a descriptive name
- It is recommended to use only Latin letters
- Should be neither too long nor too short

Ex- var New = 2; // Here N is capital

var _2Pac; // This identifier begins with _

Incorrect Identifiers

Ex-var new;// new is a keyword

var 2Pac; // Cannot begin with a digit

7. What are variables?

Ans. In Programs the data is to be stored in memory for computing.

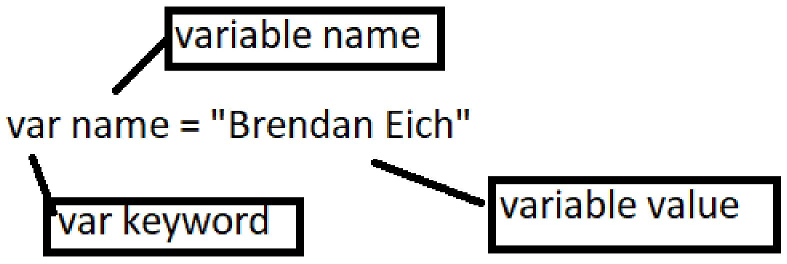
The “variable” are one which hold the data stored in memory .

A variable has a “name”, ”datatype” ,”value”.

In Javascript variable are declared using keyword “var”

And the datatype is implicitly managed by javascript engine.

Example. Variable declaration and assignment in javascript.



8. What are Datatypes in Javascript?

Ans. Datatypes are one which defines the type of data stored in memory.

For example – 256 type is (number)

"Javascript" type is (string)

The data is stored in two ways

- If the data to be stored is complex (ie) it contains data of different types then that complex data is stored in heap memory,and are called non primitive datatype.

Object,Array,functions are stored in heap memory.

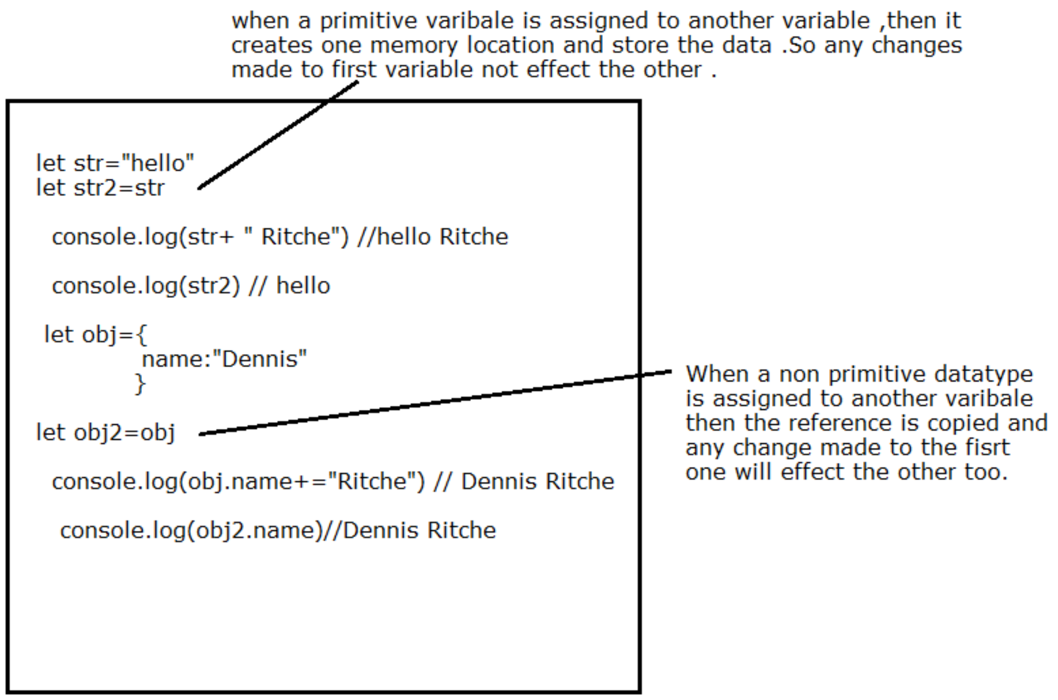
Here variable store reference to data in heap

The complex data is stored in Heap memory .

```
Ex –Employee details{
        Name: "dennis",
        Age: 25,
        DOB: 10/Aug/1992
        Position: "developer"
        }---------------------------- (Object)
```

- All other simple data like containing single values are stored in stack memory rather than heap , and are called primitive datatype.
- The memory location are stored in variable .

  Ex-var str="hello"



when a primitive varibale is assigned to another variable ,then it creates one memory location and store the data .So any changes made to first variable not effect the other .

```
let str="hello"
let str2=str

  console.log(str+ " Ritche") //hello Ritche

  console.log(str2) // hello

 let obj={
            name:"Dennis"
           }

let obj2=obj

  console.log(obj.name+="Ritche") // Dennis Ritche

   console.log(obj2.name)//Dennis Ritche
```

When a non primitive datatype is assigned to another varibale then the reference is copied and any change made to the fisrt one will effect the other too.

In javascript there are 8 datatypes .

1.Number

2.String

3.Boolean

4.Objects

5.function

6.symbol

7.undefined

8.null

9. What is a Number Datatype?

All numbers in javascript are stored internally as double -precision floating-point numbers.

- Numbers can be wrapped as object of type Number.
- Ex-var num=new Number(100)//Number{100}

In javascript the Number datatype represents

- Integer
- Float
- Exponential
- Hexadecimal
- octal

Integers

- represent whole numbers
- Has range of value, depending on size of memory used
- Number can hold integer value from

–9007199254740992 to +9007199254740992

Floating Point types

- Represent real numbers
- Have range of value and precision
- 32 bit OS will have 32 bit number and 64 bit OS will have 64 number.
- Floating point type can hold numbers from

-9007199254740992 to +9007199254740992
Examples

- var PI = Math.PI; // 3.141592653589793
- var minValue = Number.MIN_VALUE; // 5e-324
- var maxValue = Number.MAX_VALUE; // 1.79e+308
- var PI = Math.PI; // 3.141592653589793
- var div0 = PI / 0; // Infinity
- var divMinus0 = -PI / 0; // -Infinity var unknown = div0
- divMinus0; // NaN

Number Conversion

- Convert floating-point to integer number

Ex- var valueDouble=8.75;
 var valueInt=valueDouble | 0 /// 8

- Converting to integer with rounding

Ex- var valueDouble=8.75;
 Var roundedInt = (valueDouble + 0.5) | 0 //9

- Converting String to an integer

Ex- var str = ‘1234’

 Var int = (str) | 0+1 //1234

Boolean Datatype

- Has two possible values
  True and False
- Is useful in logical expressions

Example

- var a = 1; var b = 2;
- var greaterAB = (a > b); console.log(greaterAB); // false
- var equalA1 = (a == 1); console.log(equalA1); // true

10. What is String Datatype ?

- String datatype represents sequence of characters
- String are represented by either with Single Quote(‘ ‘) or double (“ “) quote, or with back ticks (` `)
- Ex - ‘javascript’ , “javascript” , `javascript`.
- String are stored accordingly to Unicode

11. What is Unicode and ASCII code?

Unicode is standards for encoding ,representing most of the world's writing system.

12. What is ASCII Code ?

ASCII CODE- every character in high level language is represented with unique value , The compiler converts this unique value to binary for machine understandability.

- This unique value that represents all character of a language was called ASCII codes
- This Ascii code were bounded only for specific language English
- Mean It can just recognize character from English not of any other language .
- So comes the Unicode that now has codes for characters from all most all languages in universe ,
- it is so called UNICODE-UNIVERSAL CODE

Ex- var ца = 'Тва а ца!';   alert(ца); //here the string is in chinese

13. What is difference between Undefined and Null ?

Undefined

- In JS there is a special value undefined
- When there is variable declared but not initialized then undefined is value assigned to that variable .

```
Ex- var str
            console.log(str)//undefined
            console.log(typeof(str))//type we get is also undefined
```

Null

- intentially representing value as null as nothing (ie) like empty variable.

Ex- let str=null ; console.log(str)// null

- the datatype of null is object which is not present.
- Ex let str = null ; console.log(typeof(str))// object.

14. What are Local and Global variables ?

Local Variables

- Local variables are the variables that are available through out the program .
- Local variables can be referenced from any where in program .
- Local Variables are declared with keyword “var” mostly.

Ex- var a=7//a is local variable
a=”hello” //same variable “a” is referenced and overwritten.

Global Variable

- Variables that are declared without var keyword .
- When you declare a variable without var keyword ,it will be added as a property to the global object “window”

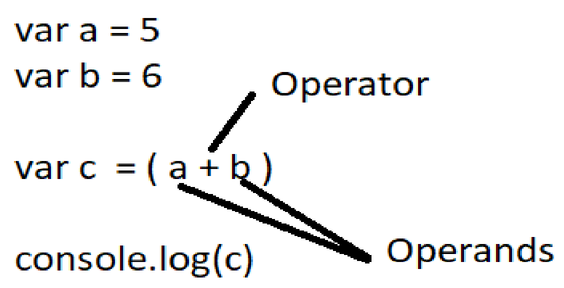
Ex- a=9;console.log(a)//9
Same a can be access as console.log(window.a)//9

- It is bad practice to use global variable

15. What is Operator ?

Operator

- Operator is an operation performed over data at runtime – Takes one or more arguments (operands) – Produces a new value.



```
var a = 5
var b = 6

var c  = ( a + b )

console.log(c)
```

```
console.log(c)//11
```

16 .What are Different Operators in Javascript?

Operators in JavaScript :

- Unary – take one operand
- Binary – take two operands
- Ternary (?:) – takes three operands

Associativity

- Except for the assignment operators, all binary operators are left-associative
- The assignment operators and the conditional operator (?:) are right associative

Types of Operator in JAVASCRIPT

Category	Operators
Arithmetic	+ - * / % ++ -- Logical && || ^ !
Binary	& | ^ ~ << >>
Comparison	== != < > <= >=
Assignment	= += -= *= /= %= &= |= ^= <<= > >>=
String concatenation	+

Other operator . [] () ?: new

17. What is Operator precedence?

Ans. **Precedence**

- Operators have precedence – Precedence defines which will be evaluated first
- Expressions are sequences of operators and operands that are evaluated to a single value

Highest precedence operator ()--- (parenthesis)

++ -- (postfix) new, typeof

++ -- (prefix) + - (unary) ! ~

* / %

+ -

<< >>

< > <= >= is as

== != &

|

&&

||

?:

Lower precedence operator ^ = *= /= %= += -= <<= >>= &= ^= |=

Parenthesis operator always has highest precedence

Note: prefer using parentheses, even when it seems stupid to do so

18. What are Arithmetic Operators?

- Arithmetic operators includes------- +, -, *, /

Add operator ----- +

- Used to ad integers
- If used to add string ,then concates two strings
- If used to add integer and string then converts integer to string and then add two strings.

Multiplication --- --- *

- The Asterik is used as multiplication operator to multiply numbers.
- When operated on string will give NAN.

Division operator --- /

- if used on integers returns integer (without rounding) or exception
- if used on real numbers returns real number or Infinity or NaN

Remainder operator -- %

- returns the remainder from division of integers

The special addition operator - ++

- increments a variable
- Called as increment and Decrement operator

Arithmetic operator Examples

var a = 5; var b = 4;

```
console.log( a + b ); // 9
console.log( a + b++ ); // 9
console.log( a + b ); // 10
console.log( a + (++b) ); // 11
console.log( a + b ); //11
console.log(12 / 3); // 4
console.log(11 / 3); // 3.6666666666666665
console.log(11.0 / 3); // 3.666666667
console.log(11 / 3.0); // 3.666666667
console.log(11 % 3);   // 2
console.log(11 % -3);  // 2
console.log(-11 % 3);  // -2
console.log(1.5 / 0.0);  // Infinity
console.log(-1.5 / 0.0); // -Infinity
console.log(0.0 / 0.0);  // NaN
var x = 0; console.log(5 / x);//Infinity
```

19. What is postfix and Prefix?

Post Fix

- Ex var a=5
  console.log(a++)/4 –post fix will effect after executing current line.
- console.log(a)//4
- here in above code the ++ is placed after "a" variable ,here the operation is postfix
- As it name refers post fix ,the value will be incremented after execution of current line

Prefix

- Ex var a = 6
- Console.log(++a)//7 here the variable is incremented before execution.

20 . What are Logical Operator ?

Logical operators take boolean operands and return boolean result • Operator ! turns true to false and false to true • Behavior of the operators &&, || and ^ (1 == true, 0 == false) :

Operation	||	||	||	||	&&	&&	&&	&&	^	^	^	^
Operand1	0	0	1	1	0	0	1	1	0	0	1	1
Operand2	0	1	0	1	0	1	0	1	0	1	0	1
Result	0	1	1	1	0	0	0	1	0	1	1	0

21. What are Bitwise Operators ?

Bitwise operators are used on integer numbers,it first converts them to binary then apply operator and then returns the result in form of integer.

Bitwise operators are applied bit by bit

Examples:

```
var a = 3;                    // 00000011
var b = 5;                    //  00000101
console.log( a | b);          // 00000111
console.log( a & b);          //  00000001
console.log( a ^ b);          // 00000110
console.log(~a & b);          //00000100
console.log( true << 1);      // 00000010
console.log( true >> 1);       // 00000000
```

23. What are Comparator Operators ?

Comparison operators are used to compare variables

Operators – ==, <, >, >=, <=, !=, ===, !==

Comparison operators example:

var a = 5; var b = 4;

console.log(a >= b); // True

console.log(a != b); // True console.log(a == b); // False

console.log(0 == ""); // True console.log(0 === ""); //False

24. What are Assignment Operators ?

Assignment operators are used to assign a value to a variable

– = , +=, -= , |=, ...

Assignment operators example:

var x = 6; var y = 4;

console.log(y *= 2); // 8

var z = y = 3; // y=3 and z=3

console.log(z); // 3

console.log(x |= 1); // 7 -------(x|=1) is in short of(x=x|1)

console.log(x += 3); // 10

console.log(x /= 2); // 5

25. What is **+** operator in string?

The + operator in javascript will also be used for string concatination

String concatenation operator + is used to concatenate strings

If the second operand is not a string, it is converted to string automatically .

- The (.) dot operator used to access members of object

  [] are used to access array elements through indexes

  Ex let array=[1,2,3]

  Console.log(array[0]) // 1

- Parentheses ( ) are used to override the default operator precedence

  var first = "First";

var second = "Second";

  console.log(first + second); // FirstSecond

  var output = "The number is : ";

  var number = 5;

  console.log(output + number); // The number is : 5

26. What is Conditional operator **?:**

- It behaves same as “if_else”
- Ex – let a=5
  Let result=(a==5)?(true):(false)
  Here ‘ ? ’ represents the condition
  ‟ : ‟ represents the two result statements
  The first statement after ‟ ? ” is returned when condition evaluates to “true” .
  The Second statement after ‟ ? ‟ is returned when condition is false.

27. What is ‟ **new** ‟ operator in javascript ?

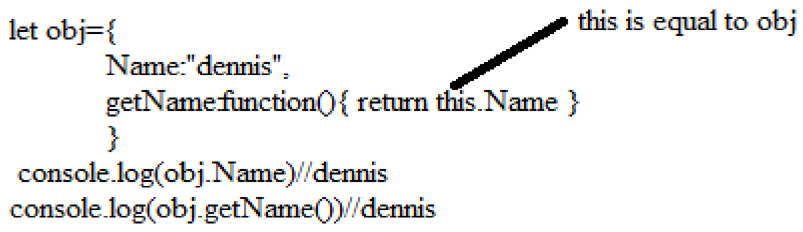
- The new operator is used to create new objects .

28. What is “ **typeof** “ Operator ?

The “ typeof “ operator returns the type of the variable .

29. What is “ **this** “ operator in javascript?

- **“ this”** operator will be referencing to the current context.
- “this” is nothing but the name of current object ,in whose context it is used.



```
let obj={
        Name:"dennis",
        getName:function(){ return this.Name }
        }
 console.log(obj.Name)//dennis
console.log(obj.getName())//dennis
```

Ex

30. What is Expression in javascript?

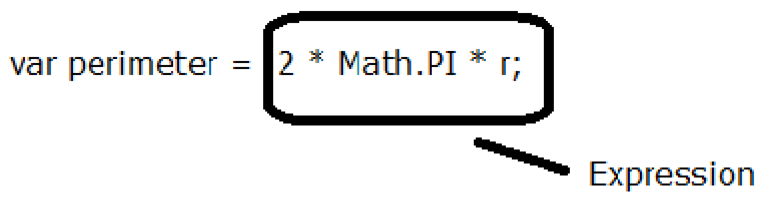
Expressions are sequences of operators, literals and variables that are evaluated to some value .

Examples:

var r = (150-20) / 2 + 5; // r=70 //

Expression for calculation of circle area
var surface = Math.PI * r * r;

Expression for calculation of circle perimeter



```
var perimeter = 2 * Math.PI * r;
```

-----------Branching Statements /Conditional Statement -------------

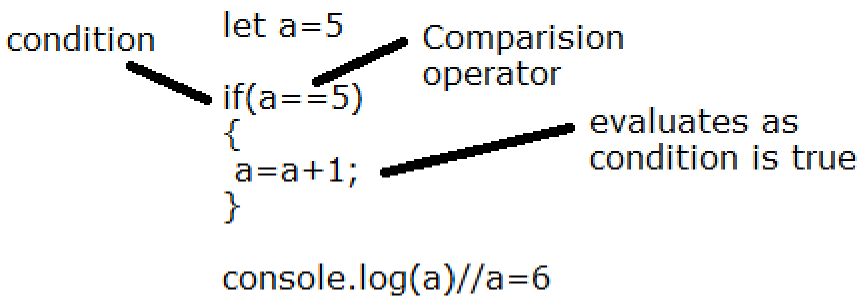
31. What are Branching Statements /Conditional Statement in Javascript?

Ans .The statement that allows to execute statement upon condition are called contional statement.

-
- If else
- Else if
- Switch case

The **if** statement

- The ‵ If ‵ is used when you have some block of code that is to be evaluated only when some condition is satisfied .
- The block of code in IF will be executed only when the condition is true.

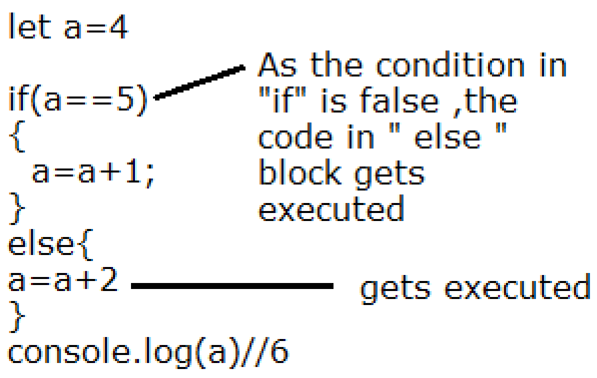


```
let a=5

if(a==5)
{
 a=a+1;
}

console.log(a)//a=6
```

Ex-

The " **else** " statement

- The Statements within " else " block are executed only when condition in " If " condition is false.



```
let a=4

if(a==5)
{
  a=a+1;
}
else{
a=a+2
}
console.log(a)//6
```

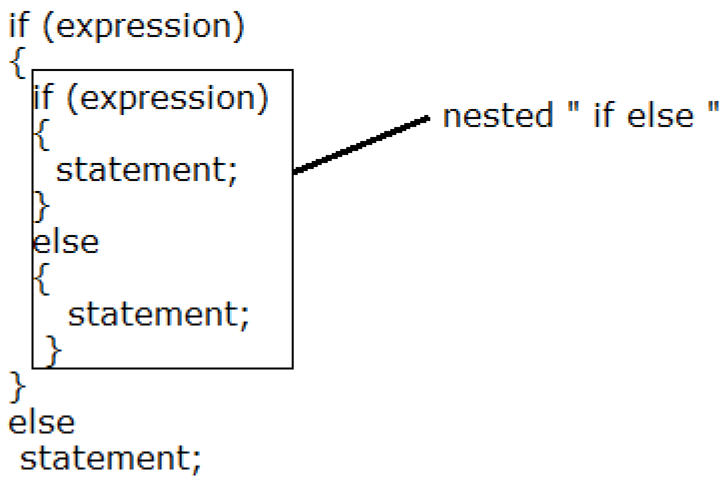
As the condition in "if" is false ,the code in " else " block gets executed

Ex-

- The code in if and else can be either a single line code ,or
- It can be a block of statements separated with semi-colon ;

Nested “ **if-else** “

- Using if-else inside another if-else block .
- This can be used when you have to check a condition upon a condition.



```
if (expression)
{
if (expression)
{
  statement;
}
else
{
   statement;
 }
}
else
 statement;
```

Ex -

32. What is the "**break**" keyword used for?

Ans:- It is used to stop iteration, and break out of the loop.

33. What is “**continue**” used for?

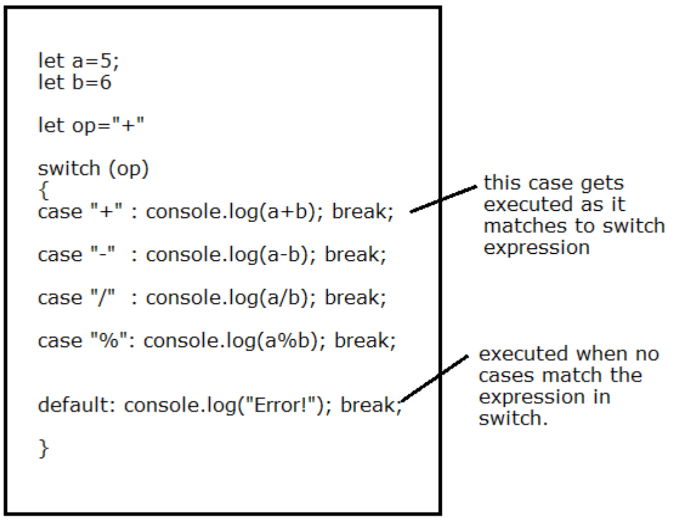
Ans:- It is used to skip to the next iteration in a loop.

34. What is “ **switch Case** “ ?

- Executes the statement depending on the expression in switch case.
- There are three keywords in switch case statement
  1.switch
  2.case
  3.default

- switch() is were we specify expression.

- case – Here different statement are written in form of cases where a particular case gets executed when it matches to expression in switch.
- default – the statement in default is executed when no cases matches the expression in switch



```
let a=5;
let b=6

let op="+"

switch (op)
{
case "+" : console.log(a+b); break;

case "-"  : console.log(a-b); break;

case "/"  : console.log(a/b); break;

case "%": console.log(a%b); break;


default: console.log("Error!"); break;

}
```

this case gets executed as it matches to switch expression
executed when no cases match the expression in switch.

---------- Looping Statement in Javascript---------------------

35 .What are Looping statement?

Ans - A looping/control statement are one that allows repeated execution of a block of code.

- Loops that never end are called an infinite loops.

36. What are different types of loops in javascript?

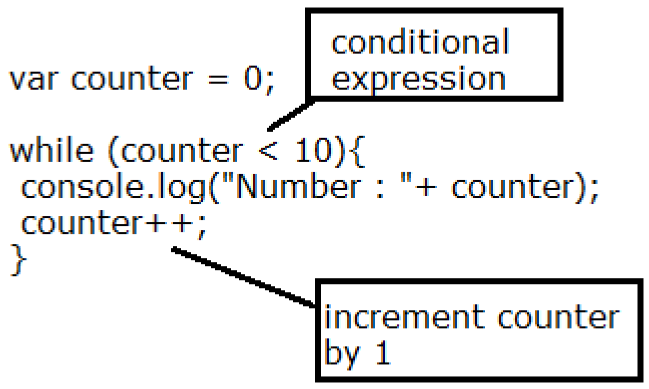
Ans. There are 3 types of loop

- **for loop**
- **while loop**
- **do while**
- **for in**

37. What is **While** loop?

Ans:- It repeatedly execute the statements in while block till the condition in " while " is true.

- It stops iterating ,when the condition gets turned to false



```
var counter = 0;

while (counter < 10){
 console.log("Number : "+ counter);
 counter++;
}
```

Ex-

38. What is "**do while**" ?

Ans The loop is executed at least once ,(i.e) the statement gets executed for once before checking the condition ,and then if the condition is true it loops again or if the condition is false it stops .

Ex – let a=6, b=7

do{

console.log(a+b)

++a

}while(a>10)// Though condition evaluates to false the loop get executed for once .//Output – 16

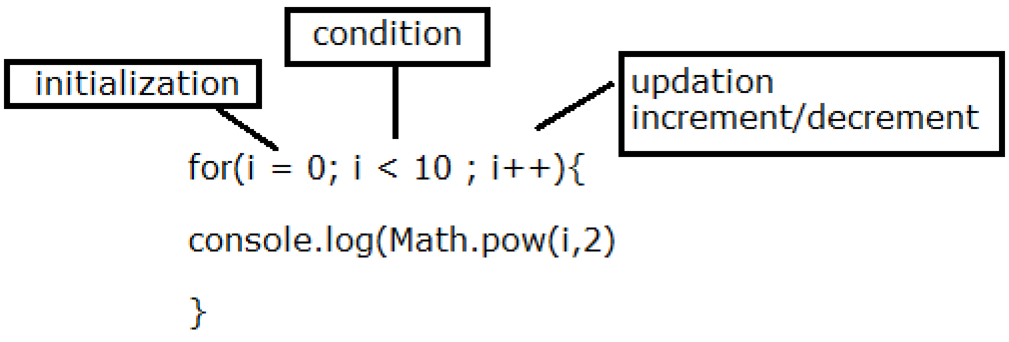
39. What is **forLoop** ?

Ans .The "for loop " is also used for repeatedly executing block of code.

- It is mainly used for looping through some collection of elements like arrays.

Syntax

```
for(initialization ; condition ; increment/decrement){
}
```



```
for(i = 0; i < 10 ; i++){
console.log(Math.pow(i,2)
}
```

Ex-

Steps of working of " for loop "

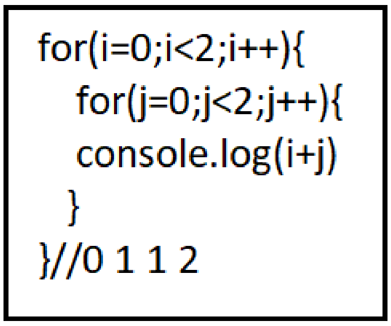
- First the initialization of variable is done
- Second condition is evaluated if true then moves for next iterate or breaks out of loop.
- The statement within the block are executed
- Then updation is done (i.e) increment or decrement .
- Repeates the process till condition is false.

Nested " for loops "

- o Implementing (for loop) within another for –loop is called nested for loop.

Working of (for loop)

First initializes and checks condition of outer (for loop) ,and if the condition is true then moves to inner (for loop) and initialize variable and checks for condition in inner (for loop) if true then iterates through the inner (for loop) and once all the iterations of inner (for loop) are done it moves to the next iteration of outer (for loop).



```
for(i=0;i<2;i++){
   for(j=0;j<2;j++){
   console.log(i+j)
  }
}//0 1 1 2
```

40. What is (**For in**) loop?

- o for-in loop iterates over the properties of an object
- o When the object is array, nodeList or liveNodeList for-in iterates over their elements.
- o When the object is not an array, for-in iterates over its properties.

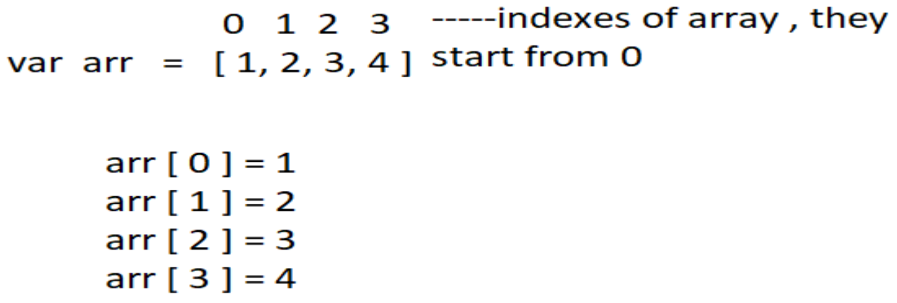
-----------------Array in javascript ------------

41. What is Array?

Ans. It`s is a collection of data items stored in continuous memory location.

- Array can store any value string, function ,number ..

Ex-



```
                 0   1  2   3   -----indexes of array , they
var  arr   =   [ 1, 2, 3, 4 ]  start from 0


        arr [ 0 ] = 1
        arr [ 1 ] = 2
        arr [ 2 ] = 3
        arr [ 3 ] = 4
```

Features of Array in Javascript

- Every element in array will have a index value
- The index value starts from 0 (zero) .
- The array elements are represented using the indexes
- The array elements are accessed using the indexes.
- The element in zero index ins first element of array
- The size of array is number of elements in array.

- The Length(size) of array is the number of elements in array

- In javascript length of array can be calculated using predefined method “length”

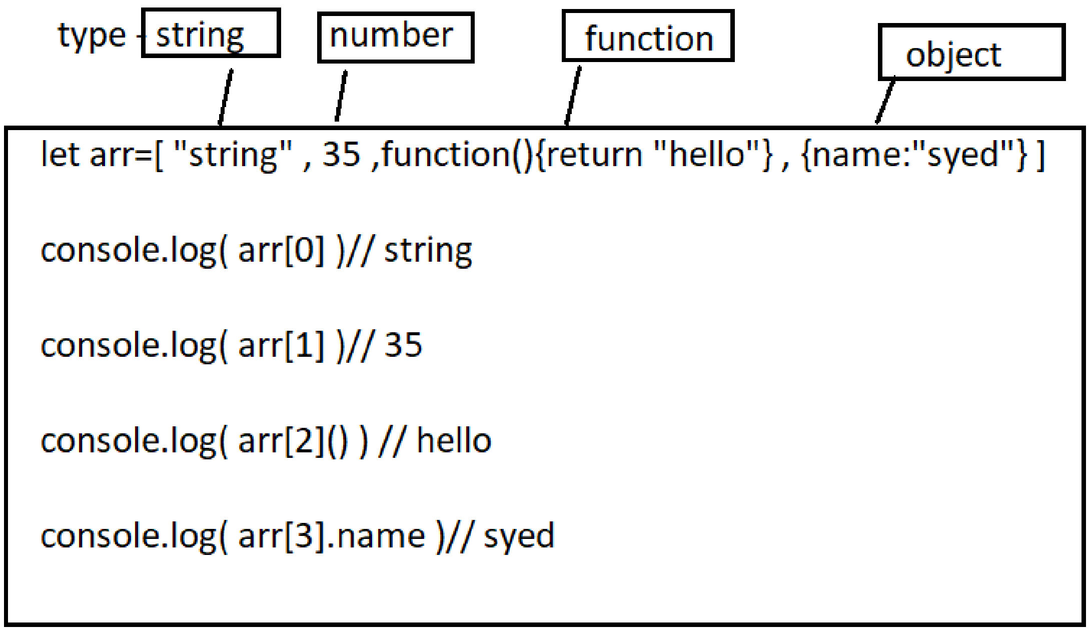
- (array_name).length will give you the length of array.

Ex var course=[“c “,” javascript “, “java”]

console.log(course.length)//3

course[0]= c , course[1] =javascript , course [2] =java

Ex



```
let arr=[ "string" , 35 ,function(){return "hello"} , {name:"syed"} ]

console.log( arr[0] )// string

console.log( arr[1] )// 35

console.log( arr[2]() ) // hello

console.log( arr[3].name )// syed
```

42. What are Different ways of declaring Array in javascript?

Ans. Initializing an array in JavaScript can be done in three ways: –

- **Using new Array(elements):**

  var arr = new Array(1,2,3,4,5);

- **Using new Array(initialLength):**

  var arr = new Array(10);

- **Using array literals:**

  var arr = [1,2,3,4,5];

Accessing of array elements using

- For loop
- For in

Accessing array elements using **for loop**

Ex –

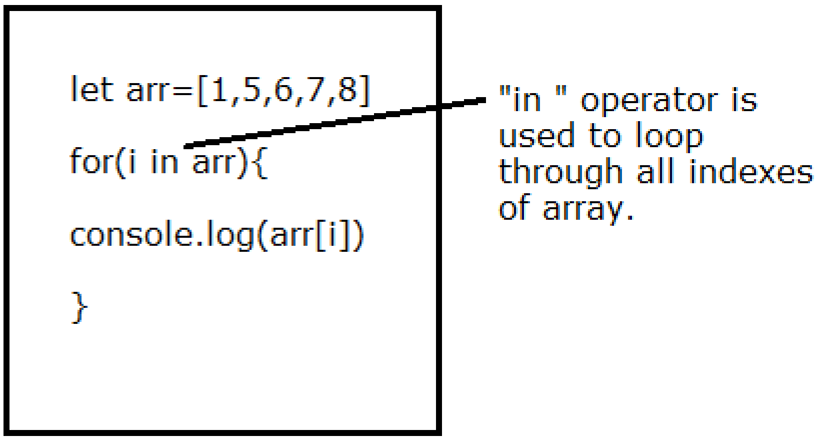
```
var  arr=[ 11 , 12 , 34 , 56]
for ( i= 0 ; i< arr.length ; i++){
console.log(arr[i])
}
```

The array length is calculated fo revery iteration .
a bad practice to cal array length in loop.

```
var  arr=[ 11 , 12 , 34 , 56]
var length = arr.length
 for(i =0 ; i< length ; i ++ ){
 console.log(arr[i])
 }
```

calculating array length
--good practice to do it before using it in forloop.

Ex – **for in** In array



```
let arr=[1,5,6,7,8]

for(i in arr){

console.log(arr[i])

}
```

"in " operator is used to loop through all indexes of array.

---------------Functions in Javascript-----------------------

43. What is **Function** in javascript?

Ans: -A Function is where you can write a "block of code" and that code can be invoked any where in your program by calling that function.

Suppose there is block of code that you are writing many times ,then you can avoid duplicate code by writing that block code in a function and calling it where ever you need

- Function can take parameters and return a value

Function have three things

1.function declaration

2.function definition

3.function call

44. Why to **Use Functions**?

- More manageable programming
- Split large problems into small pieces
- Better organization of the program
- Improve code readability and understandability
- Enhance abstraction
- Avoiding repeating code
- Improve code maintainability
- Code reusability – Using existing functions several times

45. What is Declaration of function and different ways of declaring function?

- Each function has a name
- It is used to call the function
- Describes its purpose
- Functions in JavaScript does not have return type

Functions can be defined in three ways:

– Using the constructor of the Function object

– By function declaration

– By function expression

Declaring function using constructor

var print = new Function("console.log('Hello')");

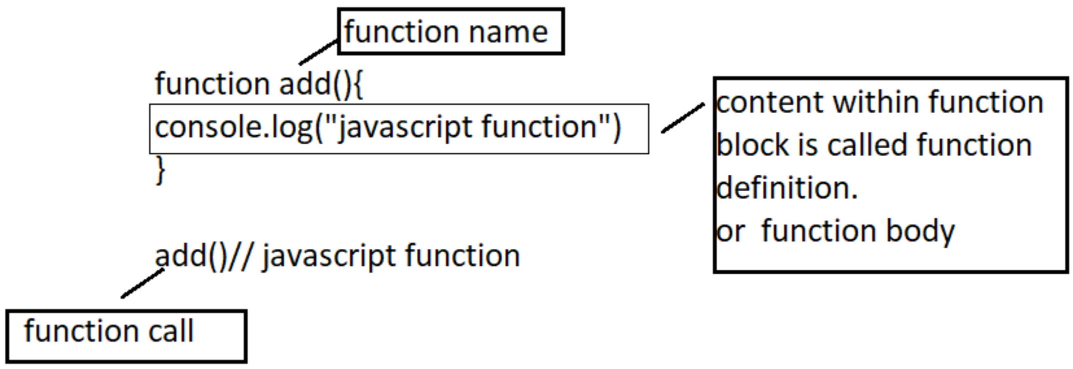
By function declaration

function print() {console.log('Hello')};

By function expression

var print = function() {console.log('Hello')};

Ex Function declaration ,call, definition



function name

```
function add(){
console.log("javascript function")
}

add()// javascript function
```

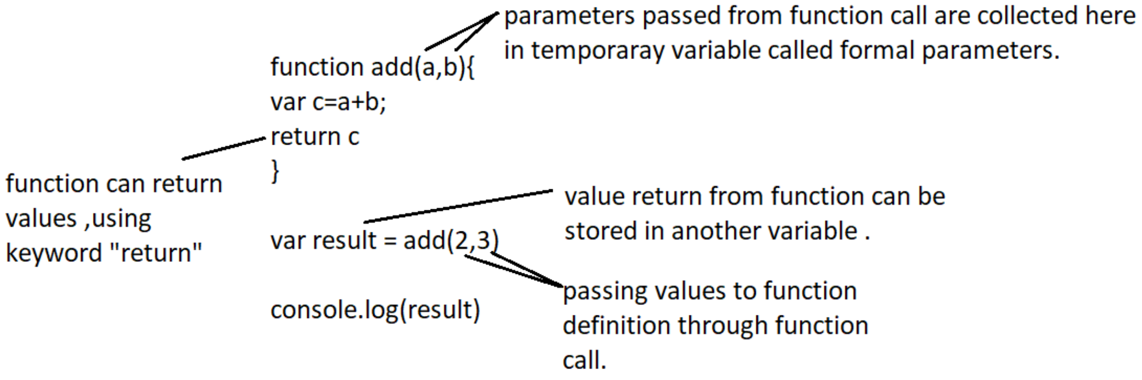
content within function block is called function definition.
or function body

function call

Parameters in function

We can pass parameters to a function through function call.

- the values or result can be returned from a function using keyword



parameters passed from function call are collected here in temporaray variable called formal parameters.

```
function add(a,b){
var c=a+b;
return c
}

var result = add(2,3)

console.log(result)
```

function can return values ,using keyword "return"

value return from function can be stored in another variable .

passing values to function definition through function call.

- Any value can be passed to a function ,a string ,number , object,array , function .

Pass by Value and **Pass by reference**

Pass by Value

- When a primitive datatype variable is passed to function ,then its value is copied to formal parameters ,and any change to that formal parameters will not effect the actual parameters

Ex -
```
var a=5
        function add(b){
        console.log(++b)//6
        }
        add(a)
        console.log(a)// 5
```

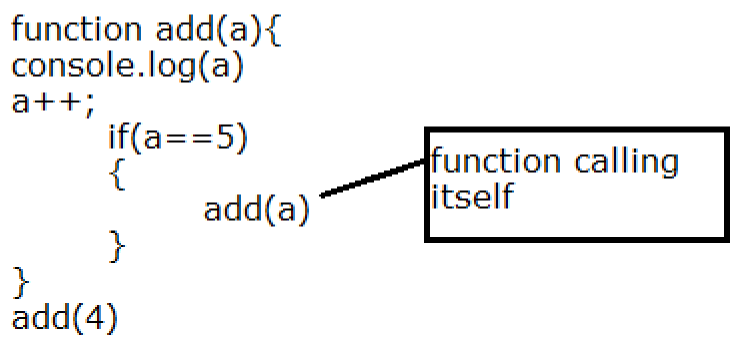
Pass by Reference

- When a non primitive datatype is passed to a function, then the reference is passed to formal parameters and any change to formal parameters will effect the actual variable.

Ex-
```
var obj={
a:5
}
function add(obj2){
console.log(++obj2.a)//6
}
add(obj)
console.log(obj.a)// 6
```

46. What is **Recursive** function ?

- A function calling itself is called recursive function .



```
function add(a){
console.log(a)
a++;
        if(a==5)
        {
                add(a)
        }
}
add(4)
```

----------------------------Scope Of variable------------------------------------

47. What is a Functional scope?

Ans:-Variable declared inside the function, can be accessed only within that function.

"var" keyword has functional scope

48. What is a block Block Scope?

Ans:- Variable declared inside a block can be available within that block, and cannot be accessed outside the block.

This block scope can be achieved by declaring a variable with "Let" keyword.

49. What is a constant?

Ans:- Constant is keyword used to declare a variable whose value once assigned cannot be overwritten anywhere in program.

50. What is "use-strict" in javascript?

Ans: In Javascript the Strict mode changes some previously-accepted mistakes into errors.

Strict mode treats these mistakes as errors so that they're discovered and promptly fixed.

----------**String and Array methods**--------------------------

String Methods

- **Length**
  This gives the length of string.
  Syntax – “stringName”.length
  Ex- var str = “javascript”
  Console.log(str.length)//10

- **charCodeAt()**
  This gives the unicode code of a character at specific index.
  Syntax – var str=”Android”
  var result= str.charcodeAt(str[0]) //65

- **concat()**
  - this will combine two string and return the string.
  - Ex var str=”Android’
    Var str1= ”pie”
    Var res = str.concat(str1)
    Console.log(res)// Androidpie

- **charAt()**
  - this will return the character at particular index.
  - Ex var str=”Android”
    Var res=str.charAt(0)
    Console.log(res)//A

- **Split()**
  - This will convet a string to array by splitting the character of array at the specific character specified in parantheses of split.
    Ex var str=”And”
    Var res=str.split(``)
    Console.log(res)// [‘A’, ’n’, ’d’]

indexOf()

- This will return the postion of first occurrence of word in string.
- Ex var str = "code and code";

```
var res = str.indexOf("code");
console.log(res)// 0
```

lastIndexOf()

This will return index of last occurrence of word in string.

- var str = "hello how r u”
- var res= str.lastIndexOf("locate");
- console.log(res)

search()

- This will give you the index of the word in string
- The difference between indexOf and search method is search cannot accept second parameter.

Ex

```
var str = "think and code";
        var res = str.search("code");
        console.log(res)//10
```

slice()

- This will return a extracted part of string.

Ex-

```
var str = "Android java ,reactNative";
        var res = str.slice(7, 13);//java
        var res = str.slice(-7, 0); //Android
        console.log(res)
```

substring()

- Substring is similar to slice()
- But here it cannot accept negative values

Ex

```
var str = "Android java reatNative";
var res = str.substring(7, 13);

console.log(res)//java
```

substr()

- substr also returns the extracted part but here the second argument is length of the extracted part.

replace()

- this is used to replace particular element with another.
- var str = "code beat";
- var res = str.replace("beat", "chef");
- console.log(res)// code chef

toUpperCase()

- this will convert the string character to uppercase
- Ex var str=”code”
- Console.log((str.toUpperCase())//CODE

toLowerCase()

- This will convert Uppercase character to lower case.
- Ex – var str=”CODE”
- Console.log(str.toLowerCase())//code

trim()

- This will remove white spaces from either side of string
- Ex str=”   javascript “
- Console.log(str.trim())//javascipt

Array Methods

- Length()
    - This returns length of array.
    - Ex var arr= [1,2,3,4]
    - console.log(var.length())//4

- push()
    - this is used to add element to end of the array.
    - Ex var arr=[1,2]
    - arr.push(3)
    - Console.log(arr)//[1,2,3]
- pop()
    - This is used to remove element from end of the array.
    - Ex var arr=[1,2,3,4]
    - arr.pop()
    - console.log(arr)// [1,2,3]

- map()
    - This is used to operate through all elements of array and store the result in new array.
    - Ex var arr=[1,2,3,4]
    - var res= arr.map(function(i){return i*i})
    - console.log(res) // [1,4,9,16]

- reduce()
    - this operates through all elements and returns single value
    - Ex var arr=[1,2,3,4]
    - var res = arr.reduce(function(a,b){return a+b})
    - console.log(res) // 10

- forEach()

    - this will operate through all the elements but do not return anything.
    - It returns undefined ,if you try to return something.
    - Ex var arr=[1,2,3,4]
    - arr.forEach(function(arrayElement){

      return(arrayElement)*(2)

      })
    - console.log(arr) // [2,4,6,8]

- shift()
    - this will remove element from beginning of array
    - Ex var arr=[1,2,3,4]
    - arr.shift()
    - Console.log(arr)//[2,3,4]

- Unshift()
    - Will add element to the beginning of an array.
    - Ex var arr=[1,2,3,4]
    - arr.unshift(2)
    - console.log(arr)// [2,2,3,4]

- join()
    - Is used to convert array to string ,it join each element of array with the special character provided in parantheses.
    - Ex var arr=[1,2,3,4]
    - var res = arr.join(‘ *’)
    - console.log(res) // 1*2*3*4
    - var res=arr.join(` `) // 1 2 3 4
    - var res= arr.join()
    - console.log(res) /// 1,2,3,4
    - if you do not mention anything in join parantheses ,by default it join the character with (,) commas.

- every()
  - This operates through all array elements on a condition and if all the elements satisfy the condition then it returns 'true' or returns "false"
  - Ex var arr=[1,2,3,4]
  - Var res=arr.every(function(element)

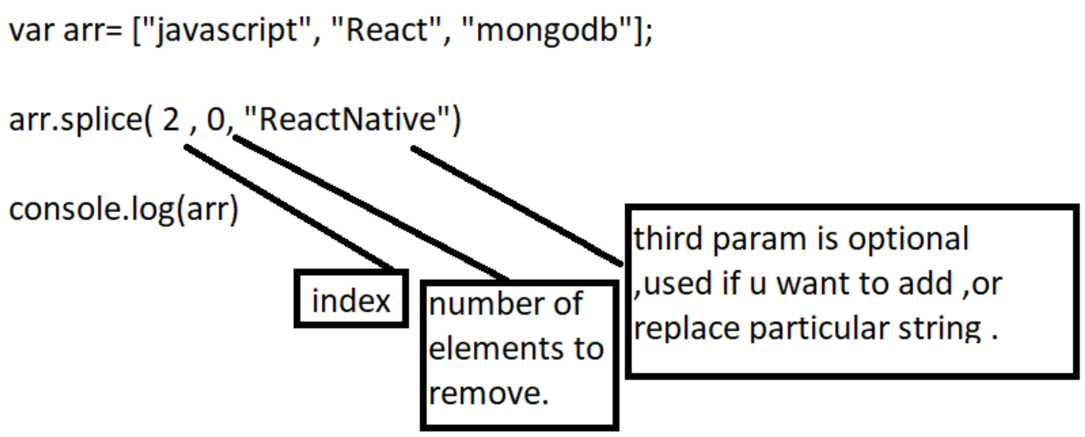
```
                        {
                        return (element>0)
                        })
console.log(res) // true
var res=arr.every(function(element){return
                              (element<0)
                                    })

     console.log(res) // false
```

- some()
  - This operates through all array elements .
  - It checks all the elements upon a condition and if atleast one element satisfy the condition then "true " is returned or else false is returned
  - Ex var arr=[1,2,9,4]
  - var res=arr.some(function(element){return

```
                                             (element>5)
})
```

  - Console.log(res) // true
  - As there is one element 9 greater then 5 it returns true

- **splice** is used to remove elements at specified position in array ,and can also be used to add element add specified position.



```
var arr= ["javascript", "React", "mongodb"];

arr.splice( 2 , 0, "ReactNative")

console.log(arr)
```

index
number of elements to remove.
third param is optional ,used if u want to add ,or replace particular string .

Output of above code – javascript ,React ,ReactNative ,Mongodb

- Entries()
  - Create a iterator Object that iterate through index and values of array.
  - Ex var arr= ["javascript", "React", "mongodb"];

```
var res = arr.entries();
console.log(res)
for (i of res) {
console.log(i)
}

Output - Array Iterator {}
[0, "javascript"]
[1, "React"]
[2, "mongodb"]
```

Math Object Methods

- Math is in-built Object in javascript that contains methods to perform Mathematic operation such as

- Math.random()
    - This will generate random numbers between 0 to 1

- Math.sqrt(number)
    - This will return square root of the given number.

- Math.round(number)
    - Returns the round up of number .

- Math.pow(a,b)
    - It returns the powers of numbers
    - First argument in Math.pow(a,b) defines the number .
    - Second argument in Math.pow(a,b) defines the power to which the first argument value must be calculated.

```
Ex var num=2
        Var res= Math.pow(num,3)
console.log(res) // 8  (i.e) (2*2*2)
```

- Math.floor()
    - It round the value to its nearest small number.
    - Ex num=2.6
    - var res= Math.floor(num)
    - console.log(num)// 2

- Math.ceil()
    - It round up the value to its nearest greater number .
    - Ex num =2.7
    - var res = Math.ceil(num)
    - console.log(res) //2

- Math.min()
    - This will return minimum value in given array of number.

- Math.max()

- This will return Maximum value from given number array.

The above were some of the most used javascript Math object methods they are few more methods to perform trigonometric calculations too.