JavaScript Functions/ operators/ events and their descriptions

- **Date()** - shows Date and Time
- **window.alert(ERROR!)** - displayes a small alert window
- **document.write()** - after an HTML document is fully loaded, will delete all existing HTML. **document.write()** method should be used only for testing.
- **document.getElementById(id)** - to access an HTML element
- **innerHTML** - property which defines HTML content
- **console.log()** - to display data. Press f12 and choose 'console' from the menu.
- **typeof** - an operator to find type of a javascript variable.
- **onchange -** An HTML element has been changed
- **onclick** - The user clicks an HTML element
- **onmouseover** - The user moves the mouse over an HTML element
- **onmouseout** - The user moves the mouse away from an HTML element
- **onkeydown** - The user pushes a keyboard key
- **onload** - The browser has finished loading the page
- **str.length** - returns length of the string object 'str'
- **indexOf()** - returns the index of the **first** occurrence of the specified string(as argument) of the string object. returns -1 if not found.
- **lastIndexOf()** - returns the index of the **last** occurrence of a specified text in a string. returns -1 if not found.
- **search()** - searches a string for a specified value and returns the position of the match
- **slice(*start, end*)** - extracts a part of a string and returns the extracted part in a new string
- **substring(*start, end*)** - same as slice(), but cannot accept negative indexes.
- **substr(*start, length*)** - same as slice(), but arg2 takes length.
- **replace(*/yeh replace hoga/g, "isse replace hoga"*)** - like ctrl + h. /.../g globally replace karega. warna first item hi replace hoga,
- **toUpperCase() -** Eg. obj2 = obj1.toUpperCase();
- **toLowerCase() -** Eg. obj2 = obj1.toLowerCase();
- **concat() -** Eg.
```
var obj1="Rahul", obj2="Nalawade", obj3;
obj3 = obj1.(" ", obj2);
```
- **charAt(*position*)** - returns character at 'position' index
- **charCodeAt(*position*)** - returns unicode of the character at 'position' index
- **split(*str*)** - converts a string into array. Eg.
```
var str = "a,b,c,d,e";
var arr = str.split(",");
```
- **x.toString()** - returns a number as a String
- **x.toExponential(*d*) -** returns a number as a String, with *d* places after decimal and base = 10. Eg. var x=9.6667; x.toExponential(3) returns 9.667e+0
- **x.toFixed(*d*)** - returns a number as a String, with *d* places after decimal
- **x.toPrecision(*p*)** - returns a string with *p* significant digits.
- **x.valueOf()** - returns number as a number
- **x.toString(16)** or **x.toString(8)** or **x.toString(2)** - returns value of number x in hexa/ octal/ binary.
- **isNaN()** - is Not a Number? Note **typeof Nan** and **typeof Infinity** is a **number**.
- **Number() -**
```
x = true;
Number(x);          // returns 1
x = false;
Number(x);          // returns 0
x = new Date();
Number(x);          // returns 1404568027739
```

```
x = "10"
Number(x);          // returns 10
x = "10 20"
Number(x);          // returns NaN
```

- **parseInt()** -
```
parseInt("10");                 // returns 10
parseInt("10.33");              // returns 10
parseInt("10 20 30");           // returns 10
parseInt("10 years");           // returns 10
parseInt("years 10");           // returns NaN
```

- **parseFloat()** -
```
parseFloat("10");               // returns 10
parseFloat("10.33");            // returns 10.33
parseFloat("10 20 30");         // returns 10
parseFloat("10 years");         // returns 10
parseFloat("years 10");         // returns NaN
```

- **Number Properties** -Eg. var x = **Number**.*property;* //**Number** is wrapper object here

  MAX_VALUE Returns the largest number possible in JavaScript

  MIN_VALUE Returns the smallest number possible in JavaScript

  NEGATIVE_INFINITY Represents negative infinity (returned on overflow)

  NaN Represents a "Not-a-Number" value

  POSITIVE_INFINITY Represents infinity (returned on overflow)

- **Math.random()** - returns a random number between 0 and 1.
- **Math.min()** - returns minimum out of list of arguments
- **Math.max()** - returns maximum out of list of arguments. list CANNOT be Array
- **Math.round(*num*)** - rounds a number to its nearest integer
- **Math.ceil(*num*) , Match.floor(*num*)** - as name suggests
- **Math Constants** -
  Math.E = Euler's number **Math.PI** = Pi value **Math.SQRT2** = sqrt(2)
  Math.SQRT1_2 = sqrt(0.5) **Math.LN2** = ln(2) **Math.LN10** = ln(10)
  Math.LOG2E = log2(E) **Math.LOG10E** = log10(E)
- **Math Object methods** -
  abs(x), sin(x), ..., asin(x), ..., exp(x), pow(base,power), sqrt(x).
- There are many methods to represent Date and/or Time in any required format. visit w3school.

Arrays:

var *arrayname* = [*item1*, *item2*, *item3*]; //one can define nested objects and/or arrays
arrays use numbers (indices) and object uses labelnames to acces it's member elements
var fibo = [0,1,1,2,3,5,8,13,21];
var person = {firstname: "Rahul", lastName: "Nalawade", age: 21};
fibo[8] = person.age; //arrays are special kind of objects with numbered indices

a.length(); //length of array
a.sort(); //alphabetical in case of strings
a.push("sedan"); // adding at last a.pop() //popping from last
Array.isArray(a); //Array is wrapper and isArray() checks if it is an Array or not
toString(a) method returns an array as a comma separated string
a.shift(), a.unshift("XUV"),

a.splice(2,0,"Lemon","Kiwi") arg1 – where new element is to be added;

arg2 – how many elements should be removed; arg3 onwards – Elements to be added

a.sort() //sorts an array considering elements as Strings

a.reverse() //reverses the order of array elements

a.sort(function(a,b){return a-b}) //Numeric sort

c = b.concat(a1, a2); //concatenate as c = [b + a1 + a2]

b = a.slice(d1, d2); //slices a from (including) a[d1] to a[d2] (id d2 omitted, till ends).

a.valueOf() is same as a.toString()

Boolean(NaN) is false; Boolean(false) is false; Boolean(null) is false;

Boolean(0) is false; Boolean(empty string) is false; Boolean(undefined) is false;

JS Forms

Constraints Validation HTML Input Attributes:

disabled	Specifies that the input element should be disabled
max	Specifies the max value of an input Element
min	Specifies the min value of an input Element
pattern	Specifies the value pattern of an input element
required	Specifies that the input field requires an element
type	Specifies the type of the input element

Constraints Validation CSS pseudo selectors:

:disabled	Selects input elements with the 'disabled' attribute specified
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no 'required' attribute specified
:required	Selects input elements with the 'required' attribute specified
:valid	Selects input elements with valid values