1: Course Introduction

2: JavaScript Intro

3: Hello World Program

```
console.log("hello World");    // see in console
document.write("hello world");    // see in output


let jss = "hello world"
console.log("my name is ... & say", jss, 10+54)
//my name is ... & say hello world 64

console.log("my name is ... & say "+ ""+jss + 10+54);
// my name is ... & say hello world1054

console.log(`my name is ... & say ${jss} 10+54`);
// my name is ... & say hello world 10 + 54
```

4: Statement and comments in js

```
// single line comment---> broweser does not show

/*multiline
 line comment---> broweser does
                        not show */
```

5: Javascript Valiable - let, var and const

Variable : -JavaScript Variables can be declared in 4 ways:

- Automatically.
- Using var.
- Using let.
- Using const.

```
x = 5;
y = 10;
z = x + y;
console.log(z);  //output: 15
```

```
let x = 5;
let y = 6;
let z = x + y;
console.log(z); // 11
```

```
var x = 5;
var y = 6;
var z = x + y;
console.log(z);  // 11
```

```
const x = 5;
const y = 6;
const z = x + y;
console.log(z);  // 11
```

Variable Declaration:-

1. Names can contain letters, digits, underscores, and dollar signs.
2. Names must begin with a letter.
3. Names can also begin with $ and _ (but we will not use).
4. Names are case sensitive (y and Y are different variables).
5. Reserved words (like JavaScript keywords) cannot be used as names.

When to Use var, let, or const?

1. Always declare variables
2. Always use const if the value should not be changed
3. Always use const if the type should not be changed (Array and Object)
4. Only use let if you can't use const
5. Only use var if you MUST support old browsers.

JavaScript Scope

JavaScript has 3 types of scope:

Block scope:

```
{
let x = 2;
}
// x can NOT be used here
```

```
{
    var x = 2;
}
// x CAN be used here
```

Function or Local scope:

- Variables declared within a JavaScriptfunction, become LOCAL to the function.
- Variables declared with var, let and const are Quite similar when declared inside a function

```
// code here can NOT use carName
function myFunction() {
    let carName = "Volvo";
    // code here CAN use carName
}
// code here can NOT use carName
```

Global scope: Variables declared with var, let and const are quite similar when declared outside a block.

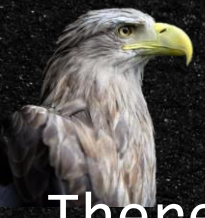
```
let carName = "Volvo";
// code here can use carName
function myFunction() {
    // code here can also use carName
}
```

Lexical scope is the ability for a function scope to access variables from the parent scope.

```
function outerFunction() {
    const outerVariable = "I'm from outer function";

    function innerFunction() {
        const innerVariable = "I'm from inner function";
        console.log(outerVariable); // Access outerVariable from the outer scope
        console.log(innerVariable); // Access innerVariable from the current scope
    }

    innerFunction();
}
outerFunction();
```



6: Operators in Javascript

There are different types of JavaScript operators:

- Arithmetic Operators + - * / % ** ++ --
- Assignment Operators = += -= *= /= %= **/
- Comparison Operators== === != !== > < >= <=
- String Operators
- Logical Operators

AND Operator (&&) – if(a && b) [if true execute else don't]

OR Operator (||) – if(a || b) [if one of them is true to execute else don't]

NOT Operator (!) – !(a<b) [returns false if a is smaller than b]

- Bitwise Operators **&** **|** **~** **^** **<<** **>>** **>>>**

- Ternary Operators condition ? value if true : alue if false

```
const age = 20;
const message = age >= 18 ? "You are an adult" : "You are a minor";

console.log(message); // Output: "You are an adult"
```

Type Operators typeof instanceof

Example

```
//  operator
var x = 7;
var y = ++x + x++ + --x + x--;
var z = ++y + y++ + --y + y--;
console.log(y);  // 32
console.log(z);  // 132
```

7: Data Type in js

JavaScript has 8 Datatypes

1. String
2. Number
3. Bigint
4. Boolean
5. Undefined
6. Null
7. Symbol
8. Object

```
// Numbers:
let length = 16;
let weight = 7.5;
```

```
// Booleans
let x = true;
let y = false;
```

```
// Strings:
let color = "Yellow";
let lastName = “manas";
```

```
// Object:
const person = { firstName: "John", lastName: "Doe" };
```

```
// Array object:
const cars = ["Saab", "674", "BMW"];
```

The Object Datatype

The object data type can contain:

1. An object
2. An array
3. A date

```
// Date object:
const date = new Date("2022-03-25");
```

```
let x = 16 + "Volvo";
//treated as
let x = "16" + "Volvo";   // "16Volvo"


let x = 16 + 4 + "Volvo";   //  "20Volvo"
let x = "Volvo" + 16 + 4;   //  "Volvo164"



// Single quote inside double quotes:
let answer1 = "It's alright";

// Single quotes inside double quotes:
let answer2 = "He is called 'Johnny'";

// Double quotes inside single quotes:
let answer3 = 'He is called "Johnny"';

console.log(answer1);    //"It's alright"
console.log(answer2);    // "He is called 'Johnny'"
console.log(answer3);    // "He is called \"Johnny\""
```

```
let x;        // Now x is undefined
x = "John";   // Now x is a String
x = 5;        // Now x is a Number
console.log(x);


// With decimals:
let x1 = 34.00;
// Without decimals:
let x2 = 34;


// let y = 123e5;    // 12300000
let z = 123e-5;   // 0.00123


let x = BigInt(12345678904343434445);
document.write(x);// 12345678904343434445

let y = 123456789043434435545
document.write(y);  1.2345e….
```

```
let x = 5;
let y = 5;
let z = 6;
console.log(x == y);       // Returns true
console.log(x == z);       // Returns false



const cars = ["Saab", "Volvo", "BMW"];
const prize = [232, 54, 464,]
console.log(cars);       // Returns   ["Saab", "Volvo", "BMW"]
console.log(prize);       // Returns  [232, 54, 464]


const person = {
    firstName: "John",                                  null == undefined // true
    lastName: "Doe",                                    null === undefined // false
    age: 50,
    eyeColor: "blue"
};
console.log(person);


let car;    // Value is undefined, type is undefined
```

8: Comparison and Logical operator

```
//  == equal value
5 == 5; -> true
6 == 5; -> false
5 == '5'; -> true

//  === equal value and type
5 === 5; -> true
6 === 5; -> false
5 === '5'; -> false

//   !=  not equal value
8 != 5; --> true
5 != 5; --> false

//  !== not equal value or not equal type
5 !== 5    -- -> false
5 !== '5'    -- -> true
5 !== 8    -- -> true
```

```
5 < 10 && 6 > 1          true
5 > 10 && 6 > 1          false

5 == 10 || 6 == 6        true
5 > 10 || 6 > 7          false

!(6 == 5)-- >             true
```

9: If Else condition

10: switch statement

11: Loop

Different Kinds of Loops

JavaScript supports different kinds of loops:

for - loops through a block of code a number of times

```
// ----------------------------for loop
// program to display the sum of natural numbers
let sum = 0;
const n = 100

for (let i = 1; i <= n; i++) {
    sum = sum + i
}
console.log('sum:', sum);
```

for/in - loops through the properties of an object.

for/of - loops through the values of an iterable object & array.

```
const person = [
    {
        fname: "John",
        lname: "Doe",
        age: 86
    },
    {
        fname: "mansee",
        lname: "maurya",
        age: 67
    },
    {
        fname: "manas",
        lname: "puggul",
        age: 69
    }
];
```

```
// Using a for loop to iterate over the array and print each person object
for (let i = 0; i < person.length; i++) {
    console.log(person[i]);
}

// Using a for...of loop to achieve the same result
for (let i of person) {
    console.log(i);
}

// Using a for...in loop to iterate over the properties of the objects in the array
for (let i in person) {
    console.log(i); // This will print the indexes of the array (0, 1, 2)
}

// You can also access specific properties of the objects using the for...in loop
for (let i in person) {
    console.log(person[i].fname); // This will print the 'fname' property of each obje
}
```

while - loops through a block of code while a specified condition is true

do/while - also loops through a block of code while a specified condition is true

12: Break, Continue and Nested loop

13: alert, confirm and prompt

Function to intract with the user: alert prompt &confirm

alert():- shows a message.

```
alert("welcome to A Doctor's Diary");
```

prompt():- shows a message, input text. It returns the text on ok or, if cancel button or Esc is clicked, null.

```
prompt(message, defaultInput);

let age = prompt('enter your age', 20);
      if (age != null)
        alert('this website contain 18+ content’);
      else
        alert('this website contain 18+ content');
```

confirm():- show a message , confirm with “ok” or “cancle”. It returns true for ok false for Vancle/Esc

```
let response = confirm('Are you sure want to visit this website!’);
    if(response)
        alert('welcome’);
    else
        alert('good luck visit next time');
```

14: Type conversion

- Converting Strings
- Converting Numbers
- Converting Booleans

JavaScript Type Conversion:-

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

By the use of a JavaScript function. `explicit conversion`

Automatically by JavaScript itself. `Implicit conversion`

Converting Strings:-

```
console.log(String(123));   // explicit   //"123"
console.log(123 + "");       // implicit   //"123 "

console.log(String(-123));        //"-123"
console.log(String(null));        //"null"
console.log(String(undefined));   //"undefined"
console.log(String(true));        //"true"
console.log(String(false));       //"false"
```

Converting Boolean:-

```
Boolean(2);  // explicit
  if (2) {....}    // implicit
 !!2               // implicit
  2 || "hello"     // implicit
```

```
console.log(Boolean(""));   // false
console.log(Boolean(0));// false
console.log(Boolean(-0));// false
console.log(Boolean(NaN));// false
console.log(Boolean(null));// false
console.log(Boolean(undefined));// false
console.log(Boolean(false));// false
```

Note:-

falsy value--> "" ,
0 ,
NaN ,
null ,
undefined ,
false

ko boolean me convert karne pr false hi milega

Converting Numbers:-

```
console.log(Number(null));      //0
console.log(Number(undefined));      //NaN
console.log(Number(true));      //1
console.log(Number(false));      //0
console.log(Number(" 12 "));      //12
console.log(Number("-12.34"));      //-12.34
console.log(Number("\n"));      //0
console.log(Number("12s"));      //NaN
console.log(Number(123));      //123
```

parseFloat() :- Parses a string and returns a floating point number

parseInt() :- Parses a string and returns an integer

```
console.log(parseFloat(3.14)); // Output: 3.14
```

```
console.log(parseInt('19Mansee'))// 19
console.log(parseInt('19Mansee06'))// 19
console.log(parseInt(42.645)); // Output: 42
                               // number, base
console.log(parseInt(1010, 8);); // 520
```

Note:- `&&` returns the **first falsy** value or the **last value** if all are truthy.

`||` returns the **first truthy** value.

```
console.log("" || 2);              // 2
console.log("hi" || "bye");        // "hi"
console.log("hi" && "bye");        // "bye"

console.log(null && NaN);          // null
console.log(null || -1);           //-1

console.log(null || 0 || -1);      //-1
console.log("hey" && 0 || -1 || null);  //-1

console.log(2 || "hello");         //2
```

15: String Manipulation

templet leteral:-

```
let str = "medicine and who treats people who are ill"
let y = `a person who has been trained in ${str}`;
console.log(y); //a person who has been trained in medicine and who treats people who are ill
```

for new line

```
let str = "medicine and who \n treats people who are ill";
console.log(str);  /* "medicine and who
                      treats people who are ill"  */
```

4 space ek sath

```
let str = "M\tS";
console.log(str) // "M   S"
```

String length with space count

```
let str = "medicine and who treats people who are
                                              ill";
console.log(str.length);  // 42
```

Double inverted comma

```
let str = "Mansee \"Puggul\" Manas";
console.log(str)  //"Mansee \"Puggul\" Manas"
```

Single inverted comma

```
let str = "Mansee \'Puggul\' Manas";
console.log(str)  // "Mansee 'Puggul' Manas"
```

```
//access any character
let str = "medicine and who treats people who are ill";
console.log(str[35]);  // "a"


// two and more string concat
let str1 = "a person who has been trained in "
let str2 = "medicine and who treats people who are ill";
console.log(str1.concat(str2));  // "a person who has been trained in medicine and who treats
people who are ill"


let str1 = "a person who has been trained in "
let str2 = "medicine and who treats ";
let str3 = "people who are ill"
console.log(str1.concat(str2, str3)); //"a person who has been trained in medicine and who
treats people who are ill"
console.log(str1 + str2 + str3);


// access substring via index value
let str = "a person who has been trained in medicine and who treats people who are ill";
console.log(str.substring(0, 12)); //"a person who"
```

find position from start

```
let str = "My name is M.S.Prince";
console.log(str.indexOf('i'));        // 8
console.log(str.indexOf('i', 9));     // 17
console.log(str.indexOf('z'));        // -1   if not present
```

find posituon from last

```
let str = "My name is M.S.Prince";
// last index means last wale character ko starting se find krega
console.log(str.lastIndexOf("i")); //17
console.log(str.indexOf("i"));//8
```

remove first and last white space

```
let str = " My name is M.S.Prince ";

console.log(str); //    " My name is M.S.Prince "
console.log(str.trim()); // "My name is M.S.Prince"
console.log(str.trimStart());  // "My name is M.S.Prince "
console.log(str.trimEnd());  // "My name is M.S.Prince "
```

toUpperCase(), toLowerCase()

```
const original = "Hello, World!";
const upper = original.toUpperCase();
const lower = original.toLowerCase();

console.log(upper); // "HELLO, WORLD!"
console.log(lower); // "hello, world!"
```

replace()

```
const original = "I love JavaScript!";
const replacedString = original.replace("JavaScript", "Python");
console.log(replacedString); // "I love Python!"
```

includes()

```
const mainString = "This is a sample string.";
const searchString = "sample";
const includes = mainString.includes(searchString);
console.log(includes); // true
```

16: Array in JavaScript

Syntax

```
const arr = [36, "prince", 8, "mansee", 87, "manas"];
console.log(typeof arr);     // object
```

access Element

```
console.log(arr[0]);      //36
console.log(arr[1]);      //"prince"
console.log(arr[2]);      //8
```

length

```
console.log("length is " + arr.length);     // 6
```

Middle Element

```
const arr = [36, "prince", 8, "mansee", 87, "manas"];
let middle_value = parseInt(arr.length / 2);
console.log(middle_value); //3
console.log(arr[middle_value]);  //"mansee"
```

Multi dimensional array

```
let books = ["maths", "physics", "english"];
let bookWithPages = [
                    ["math", 300], //0
                    ["physics", 500],//1
                    ["english", 1000]//2
                   ];
console.log(bookWithPages);
console.log(bookWithPages[1][0]);//"physics"
console.log(bookWithPages[2][1]);//1000
```

```
let book = [
            ["math", [300, "mansee"]], //0
            ["physics", 500],//1
            ["english", 1000]//2
          ];
console.log(book[0][1][1]);//"mansee"
```

array element access with for loop

```
let books = ["maths", "physics", "english"];
for (let i = 0; i < books.length; i++) {
    console.log("element " + i + " is " +
books[i]);
}
"element 0 is maths"
"element 1 is physics"
"element 2 is english"
```

array element access with for each function

```
let books = ["maths", "physics", "english"];
books.forEach(myFunction);

function myFunction(value) {
    console.log(value);
}
```

Array method

.length .toString() .pop() .push() .shift() .unshift() .delete()

.concat() .flat() .splice() .slice()

toString()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
console.log(fruits);      //["Banana", "Orange", "Apple", "Mango"]
console.log(fruits.toString());//"Banana,Orange,Apple,Mango"
```

pop(): remove last item

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
console.log(fruits.pop());//"Mango"
console.log(fruits);//["Banana", "Orange", "Apple"]
```

push(): add item at last

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.push("Kiwi");
console.log(fruits);//["Banana", "Orange", "Apple", "Mango", "Kiwi"]
```

.shift()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.shift();
console.log(fruits);//["Orange", "Apple", "Mango"]
```

.unshift()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
fruits.unshift("Lemon");
console.log(fruits);//["Lemon", "Banana", "Orange", "Apple", "Mango"]
```

.delete ()

```
const fruits = ["Banana", "Orange", "Apple", "Mango"];
delete fruits[0];
console.log(fruits);//[undefined, "Orange", "Apple", "Mango"]

delete fruits[2];
console.log(fruits);//[undefined, "Orange", undefined, "Mango"]
```

Merging(Concatenating) Arrays: -

.concat()

```
const myGirls = ["Cecilie", "Lone"];
const myBoys = ["Emil", "Tobias", "Linus"];

const myChildren = myGirls.concat(myBoys);
console.log(myChildren);//["Cecilie", "Lone", "Emil", "Tobias", "Linus"]

const yourChildren = myBoys.concat(myGirls);
console.log(yourChildren);//["Emil", "Tobias", "Linus", "Cecilie", "Lone"]

const arr1 = ["Cecilie", "Lone"];
const arr2 = ["Emil", "Tobias", "Linus"];
const arr3 = ["Robin", "Morgan"];
const myChildren = arr1.concat(arr2, arr3);
console.log(myChildren);//["Cecilie", "Lone", "Emil", "Tobias", "Linus","Robin",
                                                              "Morgan"]
```

Splicing and Slicing Arrays

The splice() method adds new items to an array.
The slice() method slices out a piece of an array.

.splice()

```
const fruits = ["Banana", "Orange", "Apple", "Mango", "Banan", "Orang", "Appl", "Mang"];
fruits.splice(2, 4, "Lemon", "Kiwi");
console.log(fruits);//["Banana", "Orange", "Lemon", "Kiwi", "Appl", "Mang"]
```

The first parameter(2) defines the position where new elements should be added(spliced
The second parameter(4) defines how many elements should be removed.
The rest of the parameters("Lemon", "Kiwi") define the new elements to be added.
The splice() method returns an array with the deleted items:

slice()

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(3, 5);  //no change in original array just copy
//  starting-value-index, last-value-index+1
console.log(citrus);   //["Apple", "Mango"]
console.log(fruits);//["Banana", "Orange", "Lemon", "Apple", "Mango"]   no changes
```

string text convert in array

```
let text = "this is a rendom text"
console.log(text.split(' '))//["this", "is", "a", "rendom", "text"]
console.log(text.split('s')); // ["thi", " i", " a rendom text"]
// jaha se arrat banana hoga usi ki input dege but input remove ho jayega
```

array convert in string array

```
const text = ["this", "is", "a rendom text"];
console.log(text.join(' ')) //"this is a rendom text"
console.log(text.join('*')) //"this*is*a rendom text"
```

join array

```
const fruits = ['Banana', 'Orange', 'Apple', 'Mango'];
let myArray = [1, 'hello', true, 3.14, { name: 'John' }];
let books = [
    { title: "The Great Gatsby", author: "F. Scott Fitzgerald", year: 1925 },
    { title: "To Kill a Mockingbird", author: "Harper Lee", year: 1960 },
    { title: "1984", author: "George Orwell", year: 1949 },
    { title: "The Catcher in the Rye", author: "J.D. Salinger", year: 1951 },
];
console.log(fruits.concat(myArray, books));
```

Sorting Arrays: -

sort()

```
 const fruits = ["Banana", "Orange", "Apple", "Mango", "Banan", "Orang", "Appl", "Mang"];
 fruits.sort();
console.log(fruits);//["Appl", "Apple", "Banan", "Banana", "Mang", "Mango", "Orang", "Orange"]
```

reverse()

```
const fruits = ["Banana", "Orange", "Apple", "Mango", "Banan", "Orang", "Appl"];
fruits.reverse();
console.log(fruits);//["Appl", "Orang", "Banan", "Mango", "Apple", "Orange", "Banana"]
```

Numeric Sort

```
//accending order
const points = [40, 100, 1, 5, 25, 10];
points.sort(function (a, b) {
    return a - b
});   //accending order
console.log(points);//[1, 5, 10, 25, 40, 100]
```

decending order

```
const points = [40, 100, 1, 5, 25, 10];
points.sort(function (a, b) { return b - a });   //decending order
console.log(points);//[100, 40, 25, 10, 5, 1];
```

Find the Lowest(or Highest) Array Value: -

```
const points = [40, 100, 1, 5, 25, 10];
points.sort(function (a, b) { return a - b });
// now points[0] contains the lowest value
// and points[points.length-1] contains the highest value
console.log("lowest value is " + points[0]);//"lowest value is 1"
console.log("maximum value is " + points[points.length - 1]);//"maximum value is 100"


console.log(Math.min(40, 100, 1, 5, 25, 10));//1
console.log(Math.max(40, 100, 1, 5, 25, 10));//100
```

Sorting Object Arrays: -

```
const myArr = [
    { name: " aa", price: 100 },
    { name: " cc", price: 100 },
    { name: " fdgre", price: 110 },
    { name: " tuy", price: 110 },
    { name: " myj", price: 110 },
    { name: " wrew", price: 100 },
    { name: " efe", price: 100 },
    { name: " wrwe", price: 110 }
];

myArr.sort(function (a, b) {
    // Compare names alphabetically
    return a.name.localeCompare(b.name);
});

console.log(myArr);
```

```
const myArr = [
        { name: "X00", price: 100 },
        { name: "X01", price: 100 },
        { name: "X04", price: 110 },
        { name: "X05", price: 110 },
        { name: "X06", price: 110 },
        { name: "X02", price: 100 },
        { name: "X03", price: 100 },
        { name: "X07", price: 110 }
    ];

myArr.sort(function (a, b) {
        return a.price - b.price;
    });
    console.log(myArr)
```

17: Operation on Array

18: Function in javascript

```
function table() {
    for (i = 1; i <= 10; i++) {
        document.write("2 * " + i + " = " + 2 * i);
        document.write("<br>")
    }
}

table();
document.write("print table");
document.write("<br>")
table();            // resuable function
```

js arrow function **let varr = () => { }**

```
function add(a, b) {
    return a + b
}
let y = add(5, 7);
console.log(y);
```

19: Parameters and Arguments

```
function table(num) {   // num is parameter
    for (i = 1; i <= 10; i++) {
        document.write(num + " * " + i + " = " + num * i);
        document.write("<br>")
    }
}

table(5);    // 5 is argument
document.write("print table");
document.write("<br>")
table(6);           // 6 is argument



function addTwoNumber(x, y) {
    document.write("sum is " + (x + y));
    document.write("<br>");
    document.write(`sum is ${x + y}`);
}
addTwoNumber(9, 56);
```

20: The arguments object

jb bhi function call hota hai to argument nam ka array bhi bnta hai usi me value store hoti hai iska kam ye hai ki kitte bhi argument pas ho kam krega ye function

```
function add(num1, num2) {
    console.log(arguments.length)
    console.log(arguments);
}

add(40, 60);
add(40, 60, 50);
add(40, 60, 50, 64);
```

```
function add(num1, num2) {
    console.log(arguments[2])
}
add(40, 60, 50, 64);//   50
```

```
function add() {
    if (arguments.length == 0) {
        console.log("No argument passed !")
    } else {
        let sum = 0;
        for (i = 0; i < arguments.length; i++) {
            sum = sum + arguments[i];
        }
        console.log(sum);
    }
}
add();
```

21: Return a value in function

```
function add(a, b) {
    return a + b;     // return ke bad execute hona band ho jayega
}
let x = add(3, 5);
console.log(x)
```

```
function compare(a, b) {
    if (a > b)
        return 1;
    else if (b > a)
        return -1;
    else
        return -0;
}
console.log(compare(6, 9));
```

22: Global variable vs local variable

```
let car = "audi"; // global variable, accessible anywhere

function myFunction() {
    let result = 67; // local variable, only accessible within the function
    console.log(result); // You can access 'result' within the function
}

function mySecondFunction() {
    let result = 69; // local variable, only accessible within the function
    console.log(result); // You can access 'result' within the function
}

console.log(car); // You can access 'car' globally
myFunction(); // You can call the function to access 'result' within it
mySecondFunction();
```

23: Anonymous function in javascript

jb bhi function call hota hai to function save hota hai isi se bachne ke liye function ko variable me save kra lete hai isse function execute hone ke bad memory free ho jati hai. Or **isa function jiska nam na ho**

```
function show() {
    console.log("M.S.Prince");
}
setTimeout(show, 2000);  //  () ni lagana hai
```

or

```
let showw = function () {
    console.log("M.S.Prince");
}
setTimeout(showw, 2000);// (variable_name , time_in_milisecond);
```

or

```
setTimeout(function () {
    console.log("M.S.Prince");
}, 2000);
```

24: Immediately Invoked function

Syntax `(function.....)();` //execute hone ke bad memory free ho jayega save ni hoga automatic delete ho jayega

Example:-

```
(function(){
    console.log("hello world");
})();
```

Note:- agar hm kisi third party ka code use karte hai to chances hai ki hamara aur uska code variable name same ho sakta hai isa hone pr anonymous function bna dete hai ek ko aur immediately call kar lete hai bina save hue turant execute ho jayega.

25: objects in javascript

Object: properties;
Action or Function

Example: in human value
properties: Name, Height, Mobile-No, Adress, Weight
action/function: eat, walk, talk, run, study

car
properties: car, model, varient, color,
action/function: AesDerivedKeyParams, Park, Servicing

Object In Programming:

- Student , Teacher, Courses in learning management System.
- Account , Account holder, cashier , bank manager etc in banking.
- Book department etc in liberary management System.

26: object properties

```
const person = {
    Name: "M.S.Prince",
    class: 12,
    age: 54,
};
```

access object properity via dot notation

```
    console.log(person.Name);
```

access object properity via array [] notation

```
      console.log(person['age']);
      console.log(person["class"]);
```

access properity

```
      console.log(person);
```

modify properity

```
person.class = 13;
console.log(person.class)//13
```

add more properties

```
person.gf = "mansee"
console.log(person)
```

delete properity

```
delete person.class;
console.log(person);
```

search properity use in operator

```
console.log('age' in person);   // true
console.log('MobileNo' in person); // false
```

iterate object properity

```
for (let key in person) {
    console.log(key + ":" + person[key]);
}
```

27: Methods in object

object ke function ko method bolege
4 tarike se method ko bna sakte hai

```
const person = {
    Name: "M.S.Prince",
    class: 12,
    age: 54,
};

// 1st method
person.sayHello = function(){
  console.log("hello M.S.Prince !");
}

person.sayHello(); // "hello M.S.Prince"
```

```
// 2nd method
const person = {
    Name: "M.S.Prince",
    class: 12,
    age: 54,
};

function greet() {
    console.log("hello M.S.Prince !");
}
person.sayHello = greet;

person.sayHello();  // "hello M.S.Prince !"
```

```
3rd method
const person = {
    Name: "M.S.Prince",
    class: 12,
    age: 54,

    sayHello: function () {
        console.log("hello M.S.Prince!");
    }
};

person.sayHello();   //"hello M.S.Prince!"
```

```
// 4th method          es6  me aaya tha
const person = {
    Name: "M.S.Prince",
    class: 12,
    age: 54,

    sayHello() {
        console.log("hello M.S.Prince!");
    }
};

person.sayHello();   //"hello M.S.Prince!"
```

28: This Keyword

“This” keyword refers to an object that is executing the current piece of code.

```
const car = {
    brand: "tata",
    model: 8,
};

const person = {
    Name: "M.S.Prince",
    class: 12,
    age: 54,

    sayHello() {
        console.log("hello! I am " + this.Name + " and I have a " + car.brand + " car");
    },
};

person.sayHello(); // "hello! I am M.S.Prince and I have a tata car"
```

29: Math Object

Math Properties(Constants)

Syntax Math.property

The syntax for any Math property is: Math.property.

JavaScript provides 8 mathematical constants that can be accessed as Math properties:

Example

```
console.log(Math.E)         // returns Euler's number
console.log(Math.PI )       // returns PI
console.log(Math.SQRT2)     // returns the square root of 2
console.log(Math.SQRT1_2)   // returns the square root of 1/2
console.log(Math.LN2 )      // returns the natural logarithm of 2
console.log(Math.LN10 )     // returns the natural logarithm of 10
console.log(Math.LOG2E)     // returns base 2 logarithm of E
console.log(Math.LOG10E)    // returns base 10 logarithm of E
```

Math Methods

The syntax for Math any methods is : **Math.method (number);**

Math.round(x)	Returns x rounded to its nearest integer
Math.ceil(x)	Returns x rounded up to its nearest integer
Math.floor(x)	Returns x rounded down to its nearest integer
Math.trunc(x)	Returns the integer part of x

```
console.log(Math.round(4.4));    // 4
console.log(Math.round(4.5));    //5

console.log(Math.ceil(4.3));     //5
console.log(Math.ceil(4.7));    //5

console.log(Math.floor(4.33));  // 4
console.log(Math.floor(4.7));    // 4

console.log(Math.trunc(4.1));   // 4
console.log(Math.trunc(4.9));  // 4
```

abs(x)	Returns the absolute value of x
acos(x)	Returns the arccosine of x, in radians
acosh(x)	Returns the hyperbolic arccosine of x
asin(x)	Returns the arcsine of x, in radians
asinh(x)	Returns the hyperbolic arcsine of x
atan(x)	Returns the arctangent of x as a numeric value between - PI / 2 and PI / 2 radians
atan2(y, x)	Returns the arctangent of the quotient of its arguments
atanh(x)	Returns the hyperbolic arctangent of x
cbrt(x)	Returns the cubic root of x
ceil(x)	Returns x, rounded upwards to the nearest integer
cos(x)	Returns the cosine of x(x is in radians)
cosh(x)	Returns the hyperbolic cosine of x
exp(x)	Returns the value of Ex

floor(x)	Returns x, rounded downwards to the nearest integer
log(x)	Returns the natural logarithm(base E) of x
max(x, y, z, ..., n)	Returns the number with the highest value
min(x, y, z, ..., n)	Returns the number with the lowest value
pow(x, y)	Returns the value of x to the power of y
random()	Returns a random number between 0 and 1
round(x)	Rounds x to the nearest integer
sign(x)	Returns if x is negative, null or positive(-1, 0, 1)
sin(x)	Returns the sine of x(x is in radians)
sinh(x)	Returns the hyperbolic sine of x
sqrt(x)	Returns the square root of x
tan(x)	Returns the tangent of an angle
tanh(x)	Returns the hyperbolic tangent of a number

30: Generate Random Number

```
console.log(Math.random());   always returns a number lower than 1.
```

Returns a random integer from 0 to 9:

```
console.log(Math.random() * 10);
console.log(parseInt(Math.random() * 10));
console.log(Math.floor(Math.random() * 10));
```

Returns a random integer from 0 to 10:

```
Math.floor(Math.random() * 11);
```

Returns a random integer from 0 to 99:

```
Math.floor(Math.random() * 100);
```

Returns a random integer from 1 to 100:

```
console.log(Math.floor(Math.random() * 100) + 1);
```

```
// random number between min (included) and max (excluded):
function getRndInteger(min, max) {
    return Math.floor(Math.random() * (max - min)) + min;
}



// This JavaScript function always returns a random number between min
and max (both included):
function getRndInteger(min, max) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}
```

31: Date Object

Creating Date Objects

Date objects are created with the new Date() constructor. There are 9 way to create a new date object:

new Date()
new Date(date string) `new Date("October 13, 2014 11:13:00");`
`new Date("2022-03-25");`

0 – 11 **0 - 6** **0 – 24**
1 - 31

new Date(year , month , day , hours , minutes , seconds , ms)

```
new Date(year,month,day,hours,minutes,seconds)
new Date(year,month,day,hours,minutes)
new Date(year,month,day,hours)
new Date(year,month,day)
new Date(year,month)
new Date(milliseconds)
```

```
const x = new Date();
console.log(x)  // Tue Nov 14 2023 GMT +05:30(India Standard Time)

const y = new Date("October 13, 2014 16:13:00");
console.log(y) //Mon Oct 13 2014 GMT +05:30(India Standard Time)
const z = new Date("2022-03-25");
console.log(z) // Fri mar 25 2022 GMT +05:30(India Standard Time)


const d = new Date(2018, 11, 24, 14, 33, 30, 0);
console.log(d) // Mon Dec 24 2018 GMT +05:30(India Standard Time)
```

Date Get Methods

Method	Description
getFullYear()	Get year as a four digit number (yyyy)
getMonth()	Get month as a number (0-11)
getDate()	Get day as a number (1-31)
getHours()	Get hour (0-23)
getMinutes()	Get minute (0-59)
getSeconds()	Get second (0-59)
getMilliseconds()	Get millisecond (0-999)
getDay()	Get weekday as a number (0-6)
getTime()	Get time (milliseconds since January 1, 1970)

Set Date Methods

Method	Description
setFullYear()	Set the year (optionally month and day)
setMonth()	Set the month (0-11)
setDate()	Set the day as a number (1-31)
setHours()	Set the hour (0-23)
setMinutes()	Set the minutes (0-59)
setSeconds()	Set the seconds (0-59)
setMilliseconds()	Set the milliseconds (0-999)
setTime()	Set the time (milliseconds since January 1, 1970)

```
const d = new Date(2018, 11, 24, 14, 33, 30, 0);
console.log(d.getFullYear());   //  2018
console.log(d.getMonth());    //11
console.log(d.getDate());    //  24
console.log(d.getDay());    //   1   monday
console.log(d.getHours());    // 14
console.log(d.getMinutes());    // 33
console.log(d.getSeconds());    //30
console.log(d.getMilliseconds());    //0
console.log(d.getTime());    //15456422210000
```

```
const d = new Date();
d.setFullYear(2018)          //  2018
d.setMonth(11)   //11
d.setDate(20)   //  24
d.setHours(14)    // 14
d.setMinutes(33)   // 33
d.setSeconds(30)   //30
d.setMilliseconds(0)    //0

console.log(d); // Thu Dec 20 2018 GMT+053Ø
                                (India Standard Time)
console.log(d.getDay()); // 4
```

50 din pahle kon sa din tha (hamesha new vRIABLE ME HI STORe krege)

```
const d = new Date();
d.setDate(d.getDate() - 30);
console.log(d); //
```

Compare Dates:

Dates can easily be compared.

```
let x = new Date("October 13, 2023 11:12:33");
let y = new Date();


if(y>x)
  console.log("x is past date")
else if(x > y )
    console.log("x is future date date")
else
    console.log("same date")
```

```
let text = "";
const today = new Date();
const someday = new Date();
someday.setFullYear(2100, 0, 14);


if (someday > today)
 text = "Today is before January 14, 2100.";
 else
  text = "Today is after January 14, 2100.";
```

32: New Keyword

```
const person = {
    name:"M.S.Prince",
    age:27
}

console.log(person.name);



// by the help of new
const person = new Object();
person.name = "M.S.Prince"

console.log(person.name);
```

33: Getter and Setter

Jb data ko manipulate (delete , add, replace .....) karke access karna ho to **get** ka use karte hai

```
const manPower = {
    name: 'M.S.Prince',
    class: 12,
    mobilenumber:7897173138,

    // getName:function(){
    //    return this.name.toUpperCase()
    // }
                // sane as
    get getName(){
        return this.name.toUpperCase();
    }
}

console.log(manPower.getName);
// jb get ka use krege to function ko call
karte time ()ka use nahi krege jb normal
method ko call krege to () iska use karte hai
```

Jb value ko enter karte hue change karna ho to **set** ka use karte hai

```
const manPower = {
    name: 'M.S.Prince',
    class: 12,
    mobilenumber: 7897173138,

    set setName(n) {
        this.name = n.toUpperCase();
    }
};

manPower.setName = "mansee maurya";
console.log(manPower);
```

34: Object constructor

- It is considered good practice to name constructor functions with an upper-case first letter
- jb kai sare object bnane ho aur unki properity same ho bs value alag alag ho tb constructor ka use karte hai
- constructor me kuch bhi add karne ke liye constructor me hi add karna hoga

```
// const student1 = {
//     firstName: "M.S.",
//     lastName: "Prince",
//     age: 67,
//     class: 23,
// };
```

```
function Student(fName, lName, age, cls) {
    this.firstName = fName;
    this.lastName = lName;
    this.age = age;
    this.class = cls;
    this.name = function () {
        return `${this.firstName} ${this.lastName}`;
    };
}
```

```
const student1 = new Student("M.S.", "Prince", 67, 23);
const student2 = new Student("John", "Doe", 45, 98);

console.log(student1); // Student { firstName: 'M.S.', lastName: 'Prince', age: 67, class: 23, name: [Function] }
console.log(student2); // Student { firstName: 'John', lastName: 'Doe', age: 45, class: 98, name: [Function] }

student1.gender = "male";
student2.gender = "female";

console.log(student1);
console.log(student2);

delete student1.age;
console.log(student1); // Student { firstName: 'M.S.', lastName: 'Prince', class: 23, name: [Function], gender: 'male

student1.sayHello = function () {
   return "Welcome to the M S 17 future";
}

console.log(student1.sayHello()); // Welcome to the M S 17 future
console.log(student1);
```

35: Object prototype

```
//  constructor me properity add karni ho ho bina constructor ko edit kiye prototypeka use karte hai

function Student(fName, lName, age, cls) {
    this.firstName = fName;
    this.lastName = lName;
    this.age = age;
    this.class = cls;

}
```

```
const student1 = new Student("M.S.", "Prince", 67, 23
const student2 = new Student("John", "Doe", 45, 98);


Student.prototype.gf = "mansee"
console.log(student1)
console.log(student2)

Student.prototype.name = function () {
    return `${this.firstName} ${this.lastName}`;
};

console.log(student1); console.log(student2)
```

36: nested object

```
const myObj = {
                name: "John",
                age: 30,
                cars: [
                        { name: "Ford", models: ["Fiesta", "Focus", "Mustang"]},
                        { name: "BMW", models: ["320", "X3", "X5"] },
                        { name: "Fiat", models: {animal: "Panda"} }
                      ]
              }
// access
console.log(myObj);
console.log(myObj.cars);
console.log(myObj.cars[2]);


// how to access animal
console.log(myObj.cars[2].models.animal);
```

37: Hoisting

JavaScript Hoisting:

Hoisting is JavaScript's default behavior of moving declarations to the top

```
console.log(x);    // undefined
var x;    //declaration
x = 7;  // assignment
console.log(x);     // 7

console.log(x);    // undefined
var x = 7;    //declaration  + assignment

console.log(x);    // Cannot access 'x' before initialization
let x;//declaration
x = 7;  // assignment
console.log(x);
```

```
hello();        // correct
function hello() {
    console.log("mansee")
}
hello();      //correct
```

```
hello();                // "mansee undefined"
function hello() {
    console.log("mansee" + " " + x)
}
hello();                //"mansee undefined"
var x = 7;
```

```
var x = 7;
hello();        // "mansee 7"
function hello() {
    console.log("mansee" + " " + x)
}
hello();    //"mansee 7"
```

38: DOM - Document Object Model

Select Element

```
document.getElementById("");
document.getElementsByClassName("");
document.getElementsByTagName("");

document.querySelectorAll("");
document.querySelector("").firstElementChild;
                          .lastElementChild;
                          .children;
                          .previousElementSibling;
                          .nextElementSibling;
                          .parentElement;
```

39: Select Element by ID

```
<body>
        <header id="main1" class="main2">
            <h1 class="main3">Simple HTML Example</h1>
            <h5>Lorem ipsum dolor sit amet consectetur.</h5>
        </header>
</body>

<script>
                        // select elment
    let selectedElement = document.getElementById("main1")
                        //  change
    selectedElement.innerHTML = "<h1>M.S.Prince</h1>"

                    // or  selection and change combined
    document.getElementById("main1").innerHTML = "<h1>M.S.Prince</h1>"
     Object          method                       Property


</script>
```

40: Select Element by Class

```
    <section>
        <h2 class="main3">Welcome to my website!</h2>
        <p class="para1">This is a simple HTML example with some basic styling.</p>
        <p class="para1">Feel free to modify and expand upon it!</p>
        <h3 id="main7">Lorem ipsum dolor sit amet.</h3>
    </section>

        <footer>
            <p class="para1">&copy; 2023 Your Name</p>
        </footer>
<script>
    // select elment via class name
    let selectedElementt = document.getElementsByClassName("para1")

    //  change via loop   bcoz    array bn jata hai na so
    for(let i = 0; selectedElementt.length;i++){
    selectedElementt[i].innerHTML = "<h1>M.S.Prince</h1>"
    }
</script>
```

41: Select Elements by Tag Name

```
<body>

    <p>This is paragraph 1.</p>
    <p>This is paragraph 2.</p>
    <p>This is paragraph 3.</p>

    <script>
        // Get all the <p> elements
        var paragraphs = document.getElementsByTagName("p");

        // Change the content of the first <p> element
        paragraphs[0].innerHTML = "Updated paragraph!";
    </script>

</body>
```

42: Query Selector in javascript

Most Powerful Selector

document.querySelector(“ ”)

```
<header>
    <h1 class="main-heading">Welcome to My Website</h1>
    <h1 class="main-heading">Welcome to My Website</h1>
    <h1>Welcome to My Website</h1>
    <h1 class="">Welcome to My Website</h1>
    <h1 class="main-heading">Welcome to My Website</h1>
</header>
<script>
    // Use querySelector to select the element with class "main-heading" in h1 tag
    let mainHeading = document.querySelector("h1.main-heading");
    console.log(mainHeading)
    // Check if the element is found
    if (mainHeading)
        // Manipulate the content of the selected element
        mainHeading.innerHTML = 'Mansee Maurya';
     else
        console.error('Element not found.');
</script>
```

document.querySelectorAll(“ ”)

```html
<header id="main1" class="main2">
    <h1 class="main3">Simple HTML</h1>
    <h5>Lorem ipsum dolor sit amet </h5>
    <p class="main3">Lorem ipsum dolor</p>
</header>

<section>
    <h2 class="main3">Welcome to my</h2>
    <p class="para1">This is a </p>
    <p class="para1">Feel free to </p>
    <h3 id="main7">Lorem ipsum dolo</h3>
</section>

<footer>
    <p class="para1">&copy; 2023 Your Name</p>
</footer>
```

```html
<script>
    // Use querySelectorAll to select all elements with class "main3" and "para1"
    let elements = document.querySelectorAll('p.main3, p.para1');

    // Loop through the selected elements and log their content
        for (let i = 0; i < elements.length; i++) {
            elements[i].textContent = "masee maurya";
        }
</script>
```

43: Traversing element in js

For indirect selection (parent , child, sbling)

```
.firstElementChild;
.lastElementChild;
.children;
.previousElementSibling;
.nextElementSibling;
.parentElement;
```

```
<section>
    <li>list-1</li>
    <li>list-2</li>
    <li id="main">list-3</li>
    <li>list-4</li>
    <li>list-5</li>
</section>
 <script>
     let firstChild = document.querySelector('section').firstElementChild;
     let lastChild = document.querySelector('section’).lastElementChild;
     let allChild = document.querySelector("section").children;
     let prevSibling = document.querySelector("#main").previousElementSibling;
     let nextSibling = document.querySelector("#main").nextElementSibling;
     let parenttt = document.querySelector("#main").parentElement;
```

44: Change HTML with javascript

45: Create and Append Element

जोड़ना,

```
let A = document.createElement("jo bhi tag banana ho");
A.attribute = value;

let C =  document.querySelector("jisme  add karna ho")
C.appendChild(A);
```

```
document.querySelector("jisme  add karna ho").appendChild(A);
```

```html
<body>

    <div id="intro">
        <p id="para">Hello this is a p tag</p>
    </div>

    <script>
        let elm = document.getElementById("para");

        // Create a new paragraph element
        var newParagraph = document.createElement("p");

        // Append the paragraph to the div with the id "intro"
        elm.appendChild(newParagraph);

        // Set the content of the paragraph
        newParagraph.textContent = "This is a dynamically created paragraph.";

        // If you want to append it to the body instead, uncomment the following line:
        document.body.appendChild(newParagraph);
    </script>
</body>
```

```
<body>
    <!-- An empty div where we will append elements -->
    <div id="container">

    </div>
    <script>
        //Select the container element by its ID
        var container = document.getElementById("container");

        // Create a new paragraph element
        var paragraph = document.createElement("p");

        // Set the text content of the paragraph
        paragraph.textContent = "bmhvbhjvjhggj";

        // Append the paragraph element to the container
        container.appendChild(paragraph);

        // You can also create and append other elements in a similar way
        var link = document.createElement("a");
        link.href = "https://www.google.com";
        link.textContent = "Visit our Website";

        container.appendChild(link);
    </script>
```

```
// Select the container div
let childContainer = document.querySelector(".childd");

// Create an image element
let imageElement = document.createElement("img");
imageElement.src = "https://i.pinimg.com/474x/a9/7b/a2/a97ba27b10a64cb534ef75d5e9c7e4ff.jpg";
imageElement.alt = "Description of the image";

// Create a video element
let videoElement = document.createElement("video");
videoElement.src = "https://www.youtube.com/embed/OkJdnxuvYjA?si=cOdsV0-KPNw25660";
videoElement.width = 320; // Set the width of the video
videoElement.height = 240; // Set the height of the video
videoElement.controls = true; // Add video controls

// Create an anchor (link) element
let anchorElement = document.createElement("a");
anchorElement.href = "https://www.google.com"; // Set the URL of the anchor
anchorElement.textContent = "Click me"; // Set the text content of the anchor
anchorElement.style.fontSize = "50px";

// Append the elements to the container
childContainer.appendChild(imageElement);
childContainer.appendChild(videoElement);
childContainer.appendChild(anchorElement);
```

```
<body>
    <div class="childd"> </div>
</body>
```

46: Insert Before an Element

```
<body>
    <ul class="container">
        <li>M.S.Prince</li>
        <li>M.S.Prince</li>
        <li>M.S.Prince</li>
        <li>M.S.Prince</li>
    </ul>
</body>
<script>
    let createHeading = document.createElement("h1");
    createHeading.textContent = "Mansee Maurya";

    const container = document.querySelector('.container');

    let abc = container.firstElementChild.nextElementSibling.nextElementSibling
    container.insertBefore(createHeading, abc)

</script>
```

47: Remove Child Element

```
<body>
    <ul id="container">
        <li>M.S.Prince</li>
        <li>M.S.Priya</li>
        <li>Mansee Maurya</li>
        <li>Twinkle</li>
    </ul>
</body>
<script>
    let parent = document.getElementById("container");
    let deletedElement = parent.lastElementChild;
    parent.removeChild(deletedElement);
</script>
```

48: Clone or Copy Element

```
<body>
    <ul id="container">
        <li>M.S.Prince</li>
        <li>M.S.Priya</li>
        <li>Mansee Maurya</li>
        <li>Twinkle</li>
    </ul>
</body>
<script>
   let parent = document.getElementById("container");
   let clone = parent.cloneNode(true); //agar false or blank krege to bs parent
                                       select hoga true se sara element

  document.body.appendChild(clone); //jaha bhi add karna ho  waha append krege
</script>
```

49: Replace element in js

```
let parent = document.getElementById("parentElement");
let newChild = document.createElement("div");
let oldChild = document.getElementById("oldElement");
parent.replaceChild(newChild, oldChild);
```

OR

```
let oldElement = document.getElementById("oldElement");
let newElement = document.createElement("div");
oldElement.replaceWith(newElement);
```

50: Insert Adjacent HTML

```
<!-- create element karne ki jarurat bhi nahi padti isme 4 position milti hai most powerful method hai ye-->
```

```
<!--beforebegin"-->
      <h2 id="main"> <!--selected-->
<!--afterbegin"-->
          Main Heading
          <p>Lorem ipsum dolor sit amet.</p>
          <img src="pic.jpg" alt="">
<!--beforeend-->
      </h2>
<!--aftereend-->
```

```
<body>
    <div id="example">
        <p>This is the target element.</p>
    </div>
</body>
 <script>
    let targetElement = document.getElementById("example");

    // Insert HTML content before the target element
    targetElement.insertAdjacentHTML("beforebegin", "<div>M.s.priya</div>");

    // Insert HTML content at the beginning of the target element
    targetElement.insertAdjacentHTML("afterbegin", "<div>M.S.Prince</div>");

    // Insert HTML content at the end of the target element
    targetElement.insertAdjacentHTML("beforeend", "<div>Mansee Maurya</div>");

    // Insert HTML content after the target element
    targetElement.insertAdjacentHTML("afterend", "<div>Puggul Maurya</div>");
 </script>
```

51: Change Attribute

```
body>
    <img src="https://images.pexels.com/photos/2505110/pexels-photo-
2505110.jpeg?auto=compress&cs=tinysrgb&w=1260&h=750&dpr=2"    alt="">
</body>
<script>
    let selectImg = document.querySelector("img"); // select element

    // for set attribute
    selectImg.setAttribute("alt", "cute baby image");
    selectImg.setAttribute("width", "600px");
    selectImg.setAttribute("height", "500px");

    // for get attribute
    console.log(selectImg.getAttribute("width"));

    // for remove attribute
    selectImg.removeAttribute("alt");

    // to check attribute availability
    console.log(selectImg.hasAttribute("alt"));
</script>
```

52: Change inline style

53: Get computed css

54: Change CSS Classes

55: Get Width and Heigh of Element

56: Dom Events in Javascript

57: Remove Event Listner in Javascript

58: Page Load Events in Javascript

59: Mouse Events

61: Scroll Events

62: Events on Form

63: Events Bubbling and Event Capuring

64: Prevent Default in js

65: BOM - Browser Object Model

66: Window Object in Javascript

67: Time out and Time interval in javascript

68: Location object in javascript

69: Navigator object in js

70: screen object in js

71: Project - 1(Temperature Converter)

72: Project - 2(Word Counter)

73: Project - 3(Background Color Switcher)

74: Project - 4(Sticky Menu Bar)

75: Project - 5(Form Validation)

76: Project - 6(Image Slider)

77: Project - 7(Stopwatch)