alert ("Hello world"); [To display an alert].

console.log ("Hello world"); [To print in console].

⇒ Computations can be done in console. →

> 34 + 35
<- 39

> 10 * 10
<- 100.

— x — x —

> document.querySelector (".class"); [To select class and
> document.querySelector ("#id"); id respectively

• document.querySelector (".class").click();

↓

To click the selected class.

NOTE - id can be selected directly by just entering id name.

```
id.innerText = "It is a text";
```

But if id name has symbols (Ex- bd-versions) we cannot access this id directly, we have to type -

```
document.getElementById("bd-versions");
```

to access this.

— x — x —

- Javascript can be used in front-end as well as back-end.
- It is a high-level dynamic interpreted programming language.
- Javascript and Java are different programming languages.

> document.getElementsByTagName("h1")[0].
style.fontSize = "29px";

In TagName, ClassName,

[0] [1] ... are used. as it can be more than one.

- Script is written in body (basically at the end)

```
<body> ----
<script>
</script>
</body>.
```

- > console.log("Hello World");
  > alert("Alert");
  > Console.warn("This is a warning");
  > console.error("Error displayed");
  > document.write("Message is directly written in HTML page");
- # > console.clear();

> console.assert(4===6);

⊗ ▸ Assertion failed: console.assert
↳ Displayed output.

> console.assert(4===(6-2));
<- undefined

NOTE –

console.log("Yash","Ayush");

It will print output as

<- Yash Ayush

(Comma is automatically removed).

- We can write the javascript in a different file, which has an extension of Js (Ex- yash.js)
- <script src="yash.js"></script>.

Javascript Variables -

- Denoted by
  1- var
  2- let.
- Constant is denoted by const

Ex-

```
var a = 10;

var b = 20;

console.log(a+b);
   <- 30
```

- NOTE- · In console, text written in "xyz" (" ") are considered as string.

  Ex- console.log("Hello world");

- And text written without " " are considered as variable.

  Ex- console.log(a+b);

  Here a and b are variables.

- Strings —

```
var a = "This is a string";

> Console.log(a);

<- This is a string.
```

- Objects -

```
var marks = {

ravi : 34,

Shubham : 30,

yash : 99

};

> consok.log(marks);

<- { ravi : 34, shubham : 30, yash : 99 }
```

- Booleans (true or false) ->

```
var a = true;

var b = false;
```

- Undefined -

```
var a;
```

- It is undefined as yet till now we have not defined its value thus on

```
console.log(a);
```

It will give us output as undefined.

- Null -

```
var a = null;
```

It means we alloted the value of a as null.

At a very high level, there are two types of data types in Javascript

1. Primitive : undefined, null, number, string, boolean, symbol.
2. Reference data types : Arrays and objects.

Array -

```
var a = [1, 2, 3, 4, 5];

> console.log(a);
  <- (5) [1,2,3,4,5]

> console.log(a[0]);
  <- 1

> console.log(a[1]);
  <- 2

> console.log(a[2]);
  <- 3
```

```
var a=34;

var b=56;

> console.log( "The sum of a & b is", a+b);

<- The sum of a & b is 100
```

} output.

An array could comprise of constants as well as strings written as "Yash".

// Arithmetic Operators

+, -, *, / are arithmetic operators.

```
a+=5;  ⇒
  a=a+5;

c*=10; ⇒
   c= c*10;
```

// Comparison operators

```
var x=34;

var y=56;

Console.log(x==y);
   <- false
Console.log(x>=y);
   <- false
Console.log(x<=y);
   <- True
```

NOTE -

Comparison operators in console.log and console.assert are different.

console.log( ) → True or false.
console.assert() → Assertion failed or undefined

// logical operators

console.log(true && true); True

console.log(true && false); False

console.log(false && true); False

console.log(false && false); False

console.log(true || true); True

console.log(true || false); True

console.log(false || true); True

console.log(false || false); False.

Incorrect method of defining function in Javascript:

```
function a(b,c);

a(b,c) {

    d=b+c;

    return d;

}
```

- Logical not.

Console.log(! false); True

- Console.log(! true); False.

- function -

① function a(b,c) {

d = b+c;

return d;

}

e = a(10,2);

console.log(e);

⇒ output →

12.

②
```
a = (b,c) => {
    d = b+c ;
    return d ;
    }
```

Conditionals in Javascript –

if, else if, else — same as in C.

Loops —

while, do-while, for — same as in C.

★★★ An unique way to print all the elements of an array –

we use forEach

```
-> a. forEach (function (b) {
       console.log (b);
   });
```

Array name.

b → Anything.

Checked by me

Array –

let a= ["fan", "camera", 1,2,3];

- for length

M-1-

b= a.length;

console.log(b);

> output will be — 5

M-2-

console.log(a.length);

> output will be — 5

It means either first relate it with something and then put that in console or the second method is directly put that in console.

~~a.pop~~

Pop —

To remove one element of array from the end.

```
let a = ["fan", "camera", 1,2,3];
a.pop();

Console.log(a);
```

> output →

(4) ["fan", "camera", 1, 2]

***

NOTE –

1.
```
m = a.pop();
Console.log(m);
```

> output → 3

The element that is removed.

2.
```
Console.log(a.pop());
```

> output → 3

Same thing different method

- Push –

To add an element from end.

a.push ("harry"); – To add a string
a.push (10); – To add a constant.

Console.log(a);

> output →

(6) ["fan", "camera", 1, 2, 3, 10]

NOTE –

1. m = a.push (10);
   Console.log (m);

> output → 6

New length of array after element is added.

Same thing different methods.

2. Console.log (a.push (10));

> output → 6

- Shift –

To remove first element of an array.

```
a.shift ();

Console.log(a);
```

> output →
(4) ["camera", 1, 2, 3]

NOTE –

1.
```
m= a.shift ();

console.log (m);
```

> output → "fan"
The element that is removed.

2.
```
console.log (a.shift());
```

> output – "fan"

Same thing different methods.

- Unshift —

To add an element from beginning.

```
a.unshift(10);

Console.log(a);
```

> output →

(6) [10, "fan", "Camera", 1, 2, 3]

NOTE —

1.
```
m = a.unshift(10);
Console.log(m);
```

output → 6

length of array after element is added

2.
```
console.log(a.unshift(10));
```

output → 6

Same thing different methods.

- toString —

To convert an array into a string.

a.toString()

<- "fan, camera, 1,2,3"

- sort —

a.sort();

NOTE <- first it will write numbers, or element which comes first in dictionary

X X

String — let a = "Harry is a good boy";

- length —

m = a.length;
console.log(m); ⇒ Output = 19

OR

console.log(a.length);

- Index of — It gives the position of a word (first one).

1- m= a.indexOf ("Harry");

Console.log (m);

2- Console.log (a.indexOf ("Harry"));

Output = 0

⇒ index of good= 11.

NOTE- Index is similar to array and starts from 0.

- Last index of - Position of last word of that kind (if repeated).

1- m= a.lastIndexOf ("good")
Console.log(m);

2. Console.log (a.lastIndexOf ("good"));

Wrong method to use Replace.

```
a.replace("Harry", "Ayush");
Console.log(a);
```

→ It will print a which is the unchanged string.

→ In replace, a new string is formed.
So to print that either relate that with a variable or type –

```
Console.log(a.replace("Harry","Ayush"));
```

- **Slice** — In this we provide index from where to where we want to cut-off

1. `m = a.slice(0,4);`
   `console.log(m);`
2. `console.log(a.slice(0,4));`

Output ⇒ • Harry

- **Replace** — To replace an element of string with any other element.

1. `m = a.replace("Harry", "Ayush");`
   (Target = Jisko, Source = Jisse)
   `console.log(m);`
2. `console.log(a.replace("Harry", "Ayush"));`

Same thing different methods

⇓

Output — "Ayush is a good boy"

Date, time, hours ....

```
let a = new Date();
```

- `Console.log(a);` → It will print complete data, (date, time, day).
- `Console.log(a.getTime());`
- `Console.log(a.getHours());`
- `Console.log(a.getDate());`
- `Console.log(a.getDay());` [0-6]
- `Console.log(a.getMinutes());`
- `Console.log(a.getFullYear());`

Document

> document.location

> document.images

- document.getElementById("click").click();
  Id Name

- document.getElementById("click").style.border =
  "2px solid blue";

document.getElementsByClassName("class");

(★) NOTE - class name, and tag name can give more than one values so [0], [1].[2] are to be used.

a= document.getElementsByClassName("class");

(1) a[0].style.background = "yellow";
↓
Targeting 1st element with class name "class".

```
<style>

.bg-primary {

    background-color: violet;
}
```

2. document.getElementsByClassName("class")[0].style.background = "yellow";

1 & 2 will produce same output.

- To add class

M-1 -

a[0].classList.add("bg-primary");

M-2 -

document.getElementsByClassName(".class")[0].classList.add(".bg-primary");

M1 and M2 will add a class of name bg-primary that is made on left-page.

NOTE - A class could be added without even defining it.

- To remove class :

```
a[0].classList.remove("bg-primary");
```

→ It will remove a class of name "bg-primary".

- Inner HTML / Inner Text.

```
a[0].innerHTML;
```

Output → HTML of a[0] will be printed.

```
a[0].innerText;
```

Output - Text of a[0] will be output.

```
console.log(a[0].innerHTML);
console.log(a[0].innerText);
```

```
a = document.getElementsByTagName("div");
console.log(a);
```

- Create Element — To create an element (< > </ >).

```
b = document.createElement("p");
b.innerText = "Yash";
```
} Adding text to the created element.

- Append child —
  To add an element inside an element.

```
a[0].appendChild(b);
```

- Replace child —
  To replace an element with another

⇒ (new child, old child)

```
Let c = document.createElement("b");
c.innerText = "This is a created bold";
```

a[0].replaceChild(c,b);

- Remove child —

First target the parent element, let us assume with a[0]
Also target the child element, that is to be removed.
And if there are multiple child it will be considered as [0], [1]..... (for ex - b[0]).

a[0].removeChild(b[0]);

Parent element child.

→ document.location

document.title

document.URL

document.Scripts

document.links

document.images

document.domain.

- document.querySelector(".container");
  ↓
  className with "container".

- document.querySelectorAll(".class");

NOTE- It works on the principle of CSS. (., #).

Events —

Example -

① `<button id="click" onclick="clicked()">click Me`

`</button>`

Function (it could be anything &a()..).

`<script>`

```
function clicked () {
    console.log("Click hua"); }
```

②
```
window.onload = function () {
    console.log("The document was loaded")
}
```

↓

It is written in script.

id tha isiliye seedha likh diya

Date: _____ Page: 34

```
id.addEventListener("click", function(){
    console.log("click hua");
});
```

If it was className or tagName, we would have used

```
b = document.getElementBy____("");
```

Events are as follows —

- click
- mouseover
- mouseout
- mouseup
- mousedown.

Set Timeout and SetInterval.

- ~~Set~~ SetTimeout (function , 2000);
  - SetTimeout (for setting time after which function call to be done).
  - function — any function
  - 2000 — Time after which function call should be done. (It is in ms.)

Ex-

```
function = () => {
    console.log("Vash");
}
```

By 2nd method of writing function.

- SetInterval — Determines how many times a function should be called.

```
SetInterval (function, 2000);
```

- function — any function
- 2000 — Number of times a function should be called.

- Clear Interval —

To clear interval, first target Set Interval with a variable And then type

```
clearInterval(a);
```

where

```
a = setInterval(function, 2000);
```

>

- Clear Timeout —

To clear timeout, first target Set Timeout with a variable and then type

```
clearTimeout(a);
```

where

```
a = setTimeout(function, 2000);
```

Javascript local storage -

- To save data as a string.

→ > localStorage.setItem ("name", "harry");

Output <- Storage { name: "harry" };

To clear local storage -

→ localStorage.clear ();

To get Item -

localStorage.getItem ("name");

< "harry"

To remove a particular item from local storage -

localStorage.removeItem ("name");

JSON — Javascript object Notation.

- By converting object into string and string into object, data can be transferred easily.

```
let obj = {
    name: "harry",
    length: 1
};

j = JSON.stringify(obj);
```

object is converted into string.

NOTE- To know whether anything is string, var, or object —

```
Console.log(typeof j);
```

< typeof obj

```
b= JSON.parse(" ");
```

↓ an string

```
console.log(b);
```

<- object will be its output.

IMP -

```
let a=34;
console.log(" this is my ${a}");
```

⇓ way to print a variable within a string.

<- this is my 34.