script
<string>

```
var x = "25";
document.write(x);
document.write(typeof x);</lscript>
```

Result

25

string

Note: 25 is showing string no number because it is written " "

```
<script>
var x = 25
document.write(x);
document.write(typeof x); </lscript>
```

Result:

25

number

```
<script>
var x = true;
document.write(x);
document.write(typeof x); </lscript>
```

Result:

true

boolean

```
<script>
var x = ["apple", "mango", banana"];
document.write(x);
document.write (typeof x); </script>
```

Result:

apple, mango, banana
object

Note: It is showing you that it is object but in actual it is array

```
var x = {
    firstname: "Istakar",
    secondname: "Husain" };
    document.write(x);
    document.write(typeof x); </script>
```

Result:

[object Object]
object

```
<script>
    var x = null;
    document.write(x);
    document.write(typeof x); </script>
```

Result:

null
object

```
<script>
var x;
document.write (x);
document.write(typeof x); </script>
```

Result:

undefined
undefined

Note: We can also write var x = undefined, instead of var x;

→ Operators

Operator is a symbol that is used to perform operations according to user requirements. We can perform this in variables or values

There are 5 type of operators

- Arithmetic Operators
  +, -, /, *, ++, --, etc
- Assignment Operators
  =, +=, -=, /=, etc
- Comparison Operators
  <, >, <=, ==, etc
- Logical Operators
  &&, ||, !

Arithemetic Operators

```
<script>
document.write(5+3); </script>
```

Result: 8

```
<script>
document.write(5-3); </script>
```

Result : 2

```
<script>
  var a=5 ; varb =3;
  document.write (a+b); </script>
```

Result: 8

```
<script>
  var a=5 ; varb =3;
  var c=a +b ;
  document.write(c); </script>
```

Result : 8

Note: You can do -, *, / the same way

```
<script>
var a=6; var b=3;
a++;
document.write(a); </script>
```

Result: 7

```
<script>
  var a = 6 ;  var b = 3;
  a--;
  document. write(a); </script>
```

Result: 5

Note: -- means subtract 1

Assignment Operators

- simple: = ; a=b
- Compound += ; a+=b ; a=a+b ; a-=b ; a=a-b

```
<script>
  var a=6; var b=3;
  a+=b;
  document.write (a); </script>
```

Result; 9

Note: It means the same a=a+b & you can also write like this & then document.write(a)

```
<script>
  var a=6; varb=3;
  a*=b;
  document.write(a); </script>
```

Result: 18

```
<script>
    var a=5 ; var b=6;
    console.log(a);
    console.log(6);
    console.log(b); </script>
```

Result:

An error will occur because there is no c.

→ Comparison in Data Type or Value

x=5 ; y=5

- (x==y) Result : True (how to read: x is equal to y)
- (x!=y) Result : False (x is not equal to y)

== equal to > greater than

=== equal value & equal type < less than

!= not equal <= less than or equal to

!== not equal value & not equal type

>= greater than or equal to

```
<script>
    var x=5 ; var y=6
    consok.log(x==y); </script>
```

Result shown in console:

False

```
<script>
    var x=5 ;    var y=5
    consde.log(x==y) ; </script>
```

Result in console: true

```
<script>
    var x=5 ;  var y="5" ;
    console.log(x===y); </script>
```

Result in console: false

```
<script>
    var x=5 ; var y=5;
    console.log(x!=y); </script>
```

Result in console: false

```
<script>
    var x=5 ; var y=5;
    console.log(x!==y); </script>
```

Result in console: false

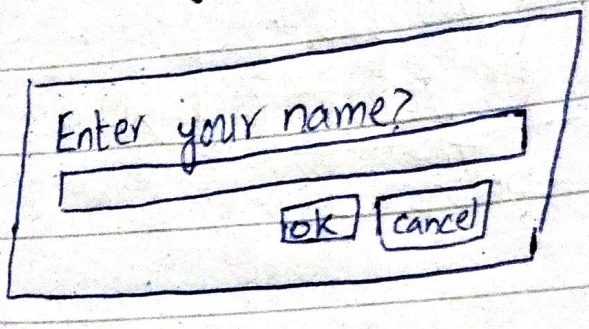
```
<script>
    var x=5 ; var y="5" ;
    console.log(x!==y); </script>
```

Result in console: true

Prompt box

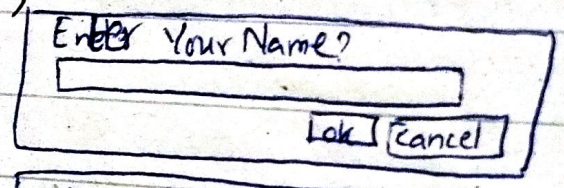
```
<script>
prompt('Enter your Name?') <script>
```

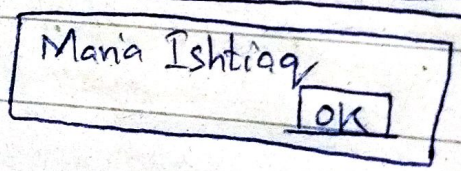
Result:




```
<script>
var a=prompt('Enter Your Name?');
alert (a);
</script>
```

Result:

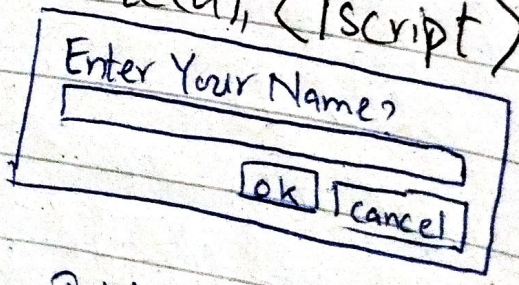




Note:
Enter your name & press ok; Then

```
<scrip>
var a =prompt('Enter Your Name?');
document.write(a); </script>
```

Result:



Note:
Enter your name & press ok, then

Maria Ishtiaq

Loop

There are 3 types of loop (i) for loop
(ii) while loop (iii) do while loop

```
<script type="text/javascript">
for(let i=1; i<=5; i++){
  document.write(i);} </script>
```

Result: 12345

```
<script type="text/javascript">
 for(let i=1; i<=5; i++ ) {
  document.write(i+'<br>');} </script>
```

Result: 1
2
3
4
5

```
<script type="text/javascript ">
for(let i=10; i<=5; i++){
  document.write(i+'<br>');}
  document.write('Done'); </script>
```

Result: Done

```
<script type="text/javascript">
for(let i=4 ; i<=5 ; i++){
  document.write(i+'<br>');}
```

```
document.write('Done'); </script>
```

Result: 4
5
Done

```
<script type="text/javascript">
 let i=1
 while (i<=5){console.log(i);}
```

Result:

Note: As here we tell initial expression (i=1) & condition (i<=5) but we don't tell incriment (i++). That's why it will print infinite i & computer will hang.

```
<script type="text/javascript">
 Let i=1;
 while(i<=5){
  document.write(i+'<br>');
   i++;}
```

Result: 1
2
3
4
5

```
<script type="text/javascript">
let i=1;
do{document.write(i+'<br>');
 i++;
}while(i<=5);
document.write('Done'); </script>
```

Result: 1
2
3
4
5
Done

```
let i=10;
do{document.write(i+'<br>'); i++;
 }while(i<=5);
 document.write('Done') </script>
```

Result: 10
Done

```
for (let i=1 ; i<=10 ; i++){
  if (i %.2 ===0)
   document.write(i+'<br>');}
```

Result: 2
4
6
8
10

```
for (let i=1 ; i<=10 ; i++) {
   if (i %.2 !==0)
   document.write (i+'<br>');}
```

Result: 1
3
5
7
9

Array

An Array can hold many values under a single name, & you can access the values by referring to an index number.

```
<script type="text/javascript">
let subject=['Math', 'Pysics', 'Chemistry',
  'Bio'];
document.write(subject);
</script>
```

Result: Math, Physics, Chemistry, Bio

```
<script type="text/javascript">
let subject=["Math", 'Pysics', 'Chemistry', 'Bio'];
document.write(subject[1]); </script>
```

Result: Physics

```
<script type="text/Javascript">
let subject=['Math', 'Physics', 'Chemistry',
  'Bio'];
subject[3]='English';
document.write(subject[3]); </script>
```

Result: English

Array Methods

```
<script type='text/javascript'>
let software=
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
console.log(software.length); </script>
```

Result: 4

Note: '.length' tells how many things are there in an array.

```
<script type="text/javascript">
let software=
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
software.push('Premiere');
console.log(software); </script>
```

Result: 'Illustrator', 'Photoshop', 'Corel', 'Indesign', 'Premiere']

Note: ".push" add the thing to the last of the list.

```
<script type="text/javascript">
let software=
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
software.unshift('Premiere');
console.log(software); </script>
```

Result: 'Premiere', 'Illustrator', Photoshop', 'Corel', 'Indesign'

Note: '.unshift' adds the thing in the start of the list

Let software =
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
software.pop();
console.log(software); </script>

Result: Illustrator, Photoshop, Corel

Note: '.pop' removes the last thing from the list

Let software =
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
software.pop();
software.pop();
console.log(software); </script>

Result: Illustrator, Photoshop

Note: In this I used 2 time '.pop' that's why it removes 2 things from last

if I use:
software.shift();

Result: Photoshop, Corel, Indesign

Note: '.shift' removes the thing from the start of the list

Let software =
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
software.splice(1,1);
console.log(software) </script>

Result: Illustrator, Corel, Indesign

Let software =
['Illustrator', 'Photoshop', 'Corel', 'Indesign'];
software.splice(1,2);
console.log(software); </script>

Result: Illustrator, Indesign

Let software =
['Illustrator', 'Photoshop', 'Corel'];
software = [];
console.log(software); </script>

Result: [ ]

Let software = ['Illustrator', 'Photoshop', 'Corel'];
software.length = 0; console.log(software); </script>

Result: [ ]

```
let software =
['Illustrator', 'Photoshop', 'Corel'];
let position =
software.indexOf('Corel');
console.log(position); </script>
```
Result: 2

```
let text = 'My Name is Maria';
let wordarray = text.split(' ');
console.log(wordarray); </script>
```
Result: 'My', 'Name', 'is', 'Maria']

```
let text = 'My Name is Maria';
let wordarray = text.split(' ');
console.log(wordarray[3]); </script>
```
Result: Maria

```
let text = ['Illustrator', 'Photoshop', 'Corel'];
let textarray = software.join('-');
console.log(textarray);
```
Result: Illustrator-Photoshop-Corel

```
let software = ['Illustrator', 'Photoshop', 'Corel']
let software2 = ['Premiere', 'Indesign'];
let newsoftware = software.concat(software2);
console.log(newsoftwar); </script>
```
Result: Illustrator, Photoshop, Corel, Premiere, Indesign

Note: 'concat' joins the lists to make 1 list. If you have many list, then you can also join them like: software.concat(software2, software3);

```
let software = [
['Illustrator', '1 Months'],
['Photoshop', '3 Months'];
['Corel', '2 Months'] ];
console.log(software[1][0]); </script>
```
Result: P

Note: [1] represents thing on index 1. [0] repre
1st letter of [1] thing.

① Use F12 to open console
② In 'Var' we can declare value again & again
③ 'let' & 'const' are use in advance JS
④ Data types: string, number, boolean, null, undefined, Array, objects, functions.
⑤ Variable rules: ① case sensitive, ② 'var' name shouldn't keywords, 'var' are
(iii) 'var' could be number, letter, _, $ (iv) we can't start varb with number (v) 'var' could start with letter, _, $
⑥ Variable is used to store information; it reserves space in memory, its data can vary but memory location will always remains same. As a good pragrammer, if you want 2 words join in variable name, so first word start with small letter & 2nd word start with capital letter. eg fullName etc
⑦ In variable name, no space allowed.
⑧ 'Const' value can't be changed, its value must be assigned at the time of declaration.
⑨ Block scope variable: If variable declare in block of codes '{ }' it will alive only in block & will not be accessable after curly braces
⑩ Global scope variable could use globally in whole program.
⑪ Single line comment // & double line comment /* */
⑫ Print / Display in js: Writing in Html element: .innerHTML.
On Browser : window.document.write("hina");
In Console : Console.log("hina");
popup : window.alert("hina");
for input from user: let abc = prompt("Are you a girl. Y/n ?:");
⑬ Array store multiple values in single variable & values are written in a square brakets.
⑭ Object store multiple values in a single variable & values written in curly backet { } in pairs with keys. eg. let student = {name "hina", rollno: 23}
⑮ Operators: Arithmetic, Assignment, comparison, logical or conditional operator
⑯ '4+5' is called expression; '4' is operand, '+' is an operation
⑰ Arithmetic operator: +, -, *, /, %, Exponentation: increament & Decrement

(18) Unary operator:
a++ is Post increment, ++a is Pre increment, Post decrement a--, --a is Predecrement

(19) Assignment operators: (i) a= (a=2, to assign value) (ii) += (a+=4 means a=a+4) (iii) -= (a-=4 means a=a-4) (iv) *= (a*=4 means a=a*4) (v) %= (a%=4 means) (vi) **= (a**=4 means a=a**4;)

(20) Comparison Operator: [==] [!=] [>] [>=] [<] [<=] [===] [!==]

(21) Logical Operator: [And &&] [OR ||] [Not !]

(22) Conditional Operator: [if] [if-else] [if-else if]

(23) String is a sequence of characters used to represent a text.

(24) It is a primitive data type, we can create a string by using template literal && in single && double quotation.

(25) How to use template literal:
- for next line \n
- for tab(space) \t
- for print \ in string
- for write variable in string $(variable name)
- for double quotation *hello \"hello"

(26) Array is non primitive data type, store multiple values in single variable, values written is square bracket [], values are separeted by comma and each position is called index &, each value call through index number starts with zero.

(27) we use info=[]; (to empty any array) (any variable)

(28) let book=["Math", "Eng", "Urdu"];

(29) book.length (To find length of array)

(30) book.push("Comp") (to add word in the end of array)

(31) book.onshift("Comp") (to add word to the start of array)

(32) book.shift(); (to remove word from the start of array)

(33) book.pop(); (to remove word from last && return updated array)

(34) book.toString() (to convert array in string)

(35) book.concat(book2) (to join 2 or 3 array in new array don't change original array)

(36) book.indexOf("Urdu"); (to find any word's position in array)

(37) book.slice(start idx, how many) e.g book.slice(2,3)

(38) book.splice(start idx, delete how many, replace) e.g book.splice(2,1,"Phy","Bio")
The first number tells from which index to remove, the 2nd tells how many numbers to remove, && the third tells what to

36) Object store multiple values in single variable, & values are written in curly brackets { } in pairs with keys. e.g
let x = {name: "hina", rollno: 23, class: "computer"}

37) A Javascript function is a block of code designed to perform a particular task & executed when someone call it.

38) function define
function fname()
{ block of codes }

function invoke (call)
fname()

39) Arrow function
const fname = () => {
block of code }

fname()

40) You can place any number of scripts in an HTML document. Scripts can be placed in <body> or in <head> section of an html page, or in both.

41) for scripting in html page use <script></script>
- for linking with external file use <script src="xyz.js"></script>
- for linking with some web link of Js use <script src="http://www.w3schools.com/js/myScript.js"></script>
- for tell path to link use <script src="/js/xyz.js"></script>

42) To access an HTML element, JavaScript can use the document.getElementById(id) method. The id attribute defines the HTML element. The innerHTML property defines the HTML element.

43) Using document.write() after an HTML document is loaded, will delete all existing HTML. The document.write() method should only be used for testing.

44) Semicolons separate JavaScript statements. Add semicolon at the end of each executable statement

45) JavaScript ignores multiple spaces. You can add white space to your script to make it more readable.

46) Javascript code blocks are written b/w { and }.

47) Variable declared with let have Block Scope & must be declare before use. & cannot be redeclared in the same scope.

48) Variables declared inside a { } block can't be accessed from outside the block.