Java Script

→ By using javascript we can write functionality to the HTML tags.
→ Initially Javascript is called as ~~live~~ "live script"
now, it is called as "ECMS" script
→ there are 2 types of scripting languages:
1) client-side scripting
2) server-side scripting
→ the scripts which are executed on the browser is called client-side scripting
→ the scripts which are executed on the server is called server-side scripting
Ex: JSV, ASV, PHP
→ we can run the javascript in 3 ways:
1) with the help of the HTML by using <script> tag
2) directly we can execute on the server
3) by using browser console
4) In any device where we have javascript engine (javascriptvirtual Machine)
→ chrome, opera, edge = V8
→ FireFox = spidermonkey
→ ~~chakra~~ IE (Internet Explorer) = chakra

Javascript	Java
1) loosely typed	1) strongly typed
2) Interpreted based	2) Compiled based
3) object based	3) object oriented
4) extension = .js	4) extension = .java

→ variables
→ ~~it is~~ → It is a ~~name~~ name of memory location where data will be stored.
→ there are 3 ways to declare the variables in js:-
1) by using let, var, const
Ex: let i=10;
var j=20;
const name='isaa';
console.log(i); → displaying content on the server

→ we can write javascript code in any where in the HTML

→ In the HTML 4 version we have to mention which type of scripting language we are using in "type" attribute

Ex: <script type="text/javascript">

but in HTML5 version we don't need to mention because by default type = "text/javascript": if we use another scripting language. compulsory we have to mention.

→ In javascript we can represent the msg in 4 ways:

1) by using document.write()
2) by using console.log()
3) by using alert() dialog boxes
4) by using inner HTML.

→ In js we can include tabs, spaces & new lines

→ Declaring the variables in diff. ways:

1) let i=100;
   let j=2000;

2) let i=100,
   j=200,
   k=300;

3) let i=10
   ,j=20
   ,k=30;

Ex:- I=00; error because compiler is in confusion which type of decimal format it is.

→ Rules for declaring the variables (identifiers)

1) It is a combination of alphabets A-Z, a-z & numbers 0-9, & two symbols $, _
2) identifiers should start with either alphabets / numbers / $/_ but it should not start with digits.
3) keywords should not be declared as identifiers.

Ex: let _=10; (semicolon is optional)
    let $=20;
    console.log(_+$); → 30.

Difference b/w let, var & const keyword

var	let	const
→ var can be either local/global scope	→ var is local scope	→ variable is local scope
→ redeclaration/updation is possible	→ only updation is possible	→ updation & redeclaration is not possible
→ hoisting is possible variables initialized with undefine	→ hoisting is allowed but variable is not initialized with value	→ hoisting is allowed variables not initialized with values.

→ can be declared without initialization

ex: var a=10;
function fi(){
var b=20;
}
console.log(a);
console.log(b);
o/p: 10
20

ex: console.log(z);
var z=100;
o/p: undefine

var z;
console.log(z);
o/p: undefine

ex: var x=100;
var x=200;
var y=10;
y=20;

ex: var z;
console.log(z);
var z=20;
o/p: undefine

→ can be declared without being initialized

ex: let x=100;
let x=200;
redec. not possible

ex: let y=200;
y=400;
updation is possible

ex: let x=20;
function fi(){
let y=30;
}
console.log(y);
o/p: runtime error

→ at the time of declaration we have to initialize the value.

ex: const name1='isa';
name1='pav';
o/p: runtime error

→ In JS string can be represented in 3 ways:
1) by using single quote (' ')
2) by using double quote (" ")
3) by using back ticks (` `)

Data types:

→ Data type specifies type & size of the data.
→ JS support 8 basic datatypes:
1) number: It represents both integer & floating point values (int & float)
2) bigint: It is used to ~~declare~~ store the long int values.
ex: let a=1141414141n; → it represents bigint
3) string:
ex: let name1='isa'
c.log(`my name is ${name1}`); → my name is isa
c.log(`${10+20}`); → 30.

→ by using back ticks (` `) we can embeded a variable/expression into another expression

t the time of
rclaration we have
initialize the value.

```
ar st rore1 = 'isa';
ore1 = 'pav';
// runtime error
```

point values
int
g values.
bigint

iza

riable/

4) boolean:

Ex: let i = true;
c.log(typeof i);
o/p: boolean

→5) null: it defines null value. it doesn't belongs to any datatype
→it represents "empty / nothing".

6) undefined: Declaring the variable but not initializing any value to the variable then its value is undefined

Ex let i;
c.log(typeof i); → undefined.

7) object & symbols:

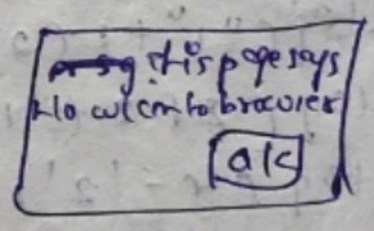
object: In primitive datatypes we store only single value but in object datatype we can store collection of data. & complex entities.

Ex: c.log(typeof Math); → object

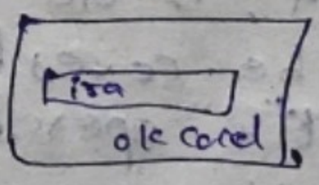
→ "symbol" datatype is used to define unique identifiers.

→In JS we have 3 ways to interact with the user:
1) alert(): it shows the msg & waits for the user to press ok button.

Ex: <script>
alert("Hello welcm to browsers"); →
</script>



Ex: <script>
prompt("which river?", "isa"); →
var i = prompt("enter 1st no"); →22
var j = prompt("enter 2nd no"); →44
document.write(i+j); →2244
alert(`${i+j}`); →2244



→there are 3 ways of writing comments in JS
1) by using double slash (//) → single line comment
2) By using multiline comment (/* ---- */)
3) by using HTML comments (<!-- -->)

Type casting

→ Converting one datatype to another datatype.

1) string conversion :- let value = true;
s = string(value);
c.log(type of s); → string.

2) Number() : by using this function we can convert other datatype to number datatype

Ex :- let i = "11";
let j = Number(i);
c.log(typeof j); → Number.

Ex :- Number(true); → 1

3) Boolean() :-

Ex :- Boolean("isa"); → true.
Boolean(""); → false
Boolean("0"); → true

operators

→ operator is a symbol which is used to perform operations on the operands

1) Arithmetic operators :- these operators are used to perform basic mathematical operations.

Ex :- +, -, *, %, /, ** (power/exponential operator)

c.log(4**(1/2)) → 2.
c.log("1"+2+2) → 122
c.log(2+2+"1") → 41
c.log(4-'2') → 2.

Ex :- Number("hello"); → Nan not a number

c.log('6'/'2') → 3

→ For "+", if one argument is string then another argument will be converted to string. this rule is applicable for "+" operator only.
for other operator -, **, *, /, %. if one argument is number then another argument will be converted to number datatype.

2) Numeric conversions :-

"+" for unary operator

Ex :- c.log(+true); → if we write + other datatype will be converted to number datatype. like

o/p :- 1

Ex :- c.log(+"");) → 0

Ex :- let is 4;
let j = 5;
c.log(Number(i) + Number(j)); → 9
c.log(+i++j); → 9

3) Assignment operators :-

→ By using this operator we can asign a value to the variables

→ It's support 2 types of assignment operators :-

1) simple assignment operator (=)

2) compound assignment operator (+=, -=, /=, *= ...)

Ex :- let n = 2;

c.log(n* = 5+4); → n* = 9 → n = n x 9 → 2 x 9 = 18.

4) Increment & Decrement operators :-

→ there are 2 types of increment & decrement operators :-

1) pre increment 2) post increment

3) pre decrement 4) post decrement

Ex :- let a = 1, b = 1;

let c = ++a;

let d = b++;

c.log(a); → 2

c.log(b); → 2

c.log(c); → 2

c.log(d); → 1

5) Bitwise operators :- these operators performs operations on the bits

1) & (bitwise AND)

2) | (bitwise OR)

3) ^ (bitwise X-OR)

4) ~ (bitwise complement)

5) >> (right shift)

6) << (left shift

Ex :- c.log(10 & 20) → 0

2 | 10 →
2 | 5 - 0
2 | 2 - 1
1 - 0

2 | 20
2 | 10 - 0
2 | 5 - 0
2 | 2 - 1
1 - 0

10 = 0000 1010
20 = 0000 0100
0000 0000

$2^4 2^3 2^2 2^1 2^0$

→ 0_4

Ex :- c.log(10 ^ 20);

10 = 0000 1010
20 = 0001 0100
000 1 1 1 1 0

$2^4 2^3 2^2 2^1 2^0$

16 + 8 + 4 + 2 + 0 ⇒ 30_4

Ex :- c.log(10 << 2) :-

we are shifting 10 for 2 times to the left side

10 = 0000 1010
0001 0100 → 10 << 1
0010 1000 → 10 << 2

6) special operators :

1) typeof : this operator is used to find datatype of given variable

2) comma operator : when it is used b/w the expressions the last expression will be evaluated & displayed

Ex : c.log(2+2, 4+4) → 8

c.log('4px' - 2) → NAN not a number

c.log(typeof alert) → function.

Arrays : Array is a collection of different types of elements.

→ there are 2 ways to create an array :

1) var array-name = [ ]

2) var array-name = new Array( ) ← Array constructor

Ex : var arr = [10]; → considered as element

var arr = new Array(10); → considered as size not an element

Ex : var arr2 = [new Array(4);

arr2[0] = 10;

arr2[1] = 20;

c.log(arr2); → 10 20 undefined undefined

Ex : var arr1 = [10, 1.2, 'ida']

c.log(arr1);

c.log(arr[0]); → 10

simple for loop

Ex : var arr = [10, 2, 'ida', 1.5];

for(var i=0; i<arr.length; i++)

{

c.log(arr[i]);

}

o/p : 10
2
ida
1.5

Ex : ~~var arr~~

→ by using tostring() method we can convert array to string datatype

→ Array is an object in JS

under this object we have following methods :

1) push(arguments) : by using this method we insert element from ~~front~~ backend / rear end.

Ex : var arr = [10, 20];

arr.push('ida');

arr.push(22, 33);

c.log(arr) → [10 20 ida 22 33].

datatype of

expressions
e evaluated

es of elements.

ze not an
element
0, 1, 2, "isa"]
;
);→10

array to

rest

2) unshift() :- this method is used to add elements from the front of the array.

Ex :- var arr = [10];
arr.unshift('isa');
arr.unshift(11, 22);
c.log(arr); → [11, 22, isa, 10]

3) ~~pop() :- this~~

3) pop(no argument) :- this method is used to remove the last element of the array.

Ex :- var arr = [10, 20];
arr.pop();
c.log(arr); → [10]

4) shift(no argument) :- It is used to remove the first element of the array.

Ex :- var arr = [10, 20];
arr.shift();
c.log(arr); → [20]

5) splice() :- this method is used for insertion & deletion of elements blw the array

syntax :- array.splice(}start, deletecount, item1, item2....)

→ when deletecount = 0, no elements is deleted

Ex :- var arr = [20, 40, 50, 60, 70];
arr.splice(1, 2);
c.log(arr) → [20, 60, 70]
arr.splice(1, 2, 4, 5); → [20, 4, 5, 50, 60, 70]
↓ not deleting ↓ inserting in 1st index
which position
we are inserting 4, 5.

6) sort() :-

var arr = [10, 30, 1]
c.log(arr.sort()) → [10, 30, 1]

→ sort() will return a value ~~but this method will not work on the server but this works on the client by using document.write()~~.

var arr = [20, 5, 2, -120, 100, 40, 700];

Ex :- var arr1 = [100, 2, 50, 20, 70, 200, 0]
c.log(arr1.sort()); → [0, 100, 2, 20, 200, 50, 70]

→ var arr = [100, 2, 50, 70, 200];
arr.splice(1, 1, "c", "java");
c.log(arr); → 100, 'c', 'java', 50, 70, 200
arr.splice(1, 2, "c", "java");
c.log(arr); → [100, 'c', 'java', 70, 200]
c.log(Array.isArray(arr)) → true
→ making the array empty
arr.length = 0;
c.log(arr); → [ ]
→ var arr = [10, , 2, 3, , 4];
c.log(arr.length); → 6
c.log(arr); → [10, <1 empty item>, 2, 3, <1 empty item>, 4];
Ex: var arr = [10, 20, 30];
for (var i of arr)
{
c.log(i); → 10
20
30
}
→ for of is used when you want to read the elements from a collection.
- for in : it is used when you want to read complex data
Ex; var arr = {'empro': 100, 'erorc': 'ii', 'add': 'hyd'};
for(var i of arr)
{
c.log(i, arr[i]); → empro 100
erorc ii
}
keys values add hyd
→ creating array with different objects.
Ex; var arr = new Array(
[10, 20, 30],
{'empro': 100, 'emprc': 'sa'},
function f1()
{
alert("Hi");
}
)
c.log(arr[0]); → [10, 20, 30]
c.log(arr[0][0]); → [10]
includes(): this method is used to check whether the specific element is available or not
Ex; var arr = [10, 20, 30];
c.log(arr.includes(30)); → true

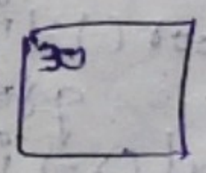
Ex: var arr = [10, 20];
arr.more = 'isa';
c.log(arr.more); → isa
c.log(arr['more']); → isa } → 2 ways to access

Ex: var i = 2;
var j = '2';
c.log(i == j) → comparing values → true
c.log(i === j) → comparing datatypes → false.

objects in JS:
1) document: document is an object which refers to webpage which will display the browser window.
→ under document object we have following methods:-
1) write(), writeln()
2) open()
3) close()
4) links()
5) forms()

links(): It is used to hold no. of links in webpage
→ dont use document.write() in js & script.

```
<script>
    document.write("welcome");
    var i = 10;
    var j = 20;
    var sum = i+j;
    document.getElementById('pisa').innerHTML = sum
</script>
<img src="Admin.png" width="200" height="200">
<p id="pisa"></p>
```



forms(): This method returns no. of forms on the webpage

```
<body>
  <p><form id="form1">
      username <input type="text" value="isa">
    </form></p>
  <p>
    <form id="form2">
      password <input type="text"></form></p>
  </body>
  <script>
let ll = document.forms.namedItem('form2');
</script>
```

cp id:

2) window object

this object is used to display window or the browser

methods:

1) open() → it is used to open a new window.

2) scroll() → it is used to scroll the window.-
→ it takes two arguments x, y. (x is x coordinate
(y is y coordinate)

3) prompt() → it is used to display prompt window on the webpage where user can enter the data.

4) close() → it is used to close the current window
→ it will not take any argument

3) Math object :

→ By using Math object we perform mathematical operations

methods:-

1) min() → it is used to display minimum of two numerical values

2) max() → it is used to display maximum of two numerical values.

3) abs(x) → it returns absolute value of a given number

4) ceil(x) → it returns nearest integer value not less than x

5) floor() → it returns nearest integer value not greater than x

6) round() → it returns nearest integer value

7) Ex: c.log(Math.min(2,4)) → 2
.max(2,4)) → 4
.abs(-100)) → 100
.ceil(11.15)) → 12
.floor(11.56)) → 11
.round(11.89)) → 12
.pow(3,2)) → 9
.sqrt(16)) → 4
.sin(30) → 1/2 → 0.5
.log(2.71) → 1

4) String object :

methods :

Ex: var rone1 = "welcome";
c.log(rone1.toLowerCase()); → welcome
c.log(rone1.toUpperCase()); → WELCOME

1) concat() :-

Ex: var rore1 = "Hlo" 2) charAt(index)

ex: var rore1 = "welcome";

c.log(rore1.charAt(2)); → 0

3) substring() :- It is the part of the given string

Ex:- substring(startingindex, endingindex);

Ex: var rore1 = "welcome"

c.log(rore1.substring(2,4));

4) indexOf() :- It returns index of given character

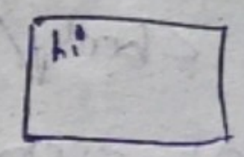
ex :- var rore1 = "welcome";

c.log(rore1.indexOf('e')) → 1

c.log(rore1.lastIndexOf('e')) → 6

5) length: c.log(rore1.length) → 7

```
<body>
  <p> welcome </p>
  <img src="Admin.png" width="100" height="100">
  <p id="ipd"></p>
  <script>
  document.getElementById('ipd').innerHTML = "hi"
  </script>
</body>
```



window object

Ex:- <script>
window.open("http://www.google.com")
</script>

Ex2 <script>
function f1() { object
mywindow = window.open("", "200", "200")
w h
object function f2() { → we can store a object in variable.
mywindow.close();
}
</script>
<button onclick="f1()"> open </button>
<button onclick="f2()"> close </button>

→ open() will contain 4 arguments i.e.,
open (url, title, width, height)

→ Moving the window by using moveBy (x, y).

Ex: function B()
mywindow.moveBy(200,200);
}

dhtml (dynamic Html)

forms()

Ex: <body>
<form action="" </form>
"
"
"
<p id="idp" ipd"> </p>
<script>
document.getElementById('idp').innerHTML=
document.forms.length
</script>
</body>

Ex: <script>
function F()
{
document.open();
document.write("welcome");
}
</script>
<button onclick="F()"> click </button>

random()

→ by using random() method we can generate random values b/w 0 & 1

sign() → it returns the sign of a number.

⇒ negative number = -1
→ positive number = +1
→ for 0 = 0

Ex: log trunc() → this method will return only integer value

Ex: Math.random() → 0.76xe2
" .cbrt(8) → 2
" . trunc(8.76) → 8
" .sign(-100) → -1

string concepts

1) at() :→ it returns indexed character from a string

ex: s="welcome"
c.log(s.at(3)); →c

2) charCodeAt() : it returns unicode value of a character based on index.

ex: s="welcome"
c.log(s.charCodeAt(0)); →101

→ Html, Java, Javascript is a unicode system.

3) endsWith() : it returns if a string ends with a specific value. or not.

ex: s="welcome too"
c.log(s.endsWith('too'); → true

4) padEnd() :

let s="5";
s = s.padEnd(4,'0')
c.log(s); → 5000

5) trim() : removing the spaces from front & end of the text.

s=" welcome "
c.log(s) → welcome
c.log(s.trim()); → welcome

6) repeat() :

ex: s="welcome"
c.log(s.repeat(3)); → welcomewelcomewelcome

7) replace() : the first occurence of word/character will be replaced.

ex: s="welcome too"
c.log(s.replace('too','to js')); → welcome to js
c.log(s.replace('e','2')); → w2lcome too js

8) replaceAll() :

c.log(s.replaceAll('e','2')); → w2lcom2 too js

9) split() : it returns array of substrings

s="welcome to ira"
const myarr = s.split(" ");
c.log(myarr); → ['welcome', 'to', 'ira']

10) trimStart() & trimEnd()

11) slice(): It will extract a part of a string from a given string

Ex: s = "welcome" to u
c.log(s.slice(7)); → welcome

Date object: By using this object we can manipulate date & time operations.

→ By using Date() constructor we create a reference variable. By using this reference variable we manipulate methods

1) var dt = new Date();
c.log(dt) → current date & time we will get

2) toString(): converting Date object to String datatype.
Ex: var dt = new Date()
c.log(dt.toString())

3) getDate(): It returns date of the month from 1 to 31
Ex: c.log(dt.getDate()) → 14

4) getDay(): It returns day of the week
Ex: c.log(dt.getDay()); → 6
0 → sun
1
6 → sat

5) getMonth(): It returns month
Ex: c.log(dt.getMonth()) → 4
0 → Jan
1
11 → Dec

6) getFullYear(): It returns 4 digit year number
Ex: c.log(dt.getFullYear()) → 2024

7) getHours(): It returns 0-23 hours

8) getMinutes(): It returns 0-59 minutes

9) getSeconds(): It returns 0-59 seconds

Ex: c.log(getHours()) → 17
c.log(getMinutes()) → 56
c.log(getSeconds()) → 23

10) setFullYear()

11) setDate() (1-31)

12) setDay() (0-6)

13) setMonth() (0-11)

functions:

→ the group of statements is called function
→ the main advantage of functions is code reusability
→ By using "function" keyword we can create a function
→ procedure will not return any value. function will return a value.
→ while creating a function, function prototype is mandatory
i.e. function functionname
→ Inside the function return keyword should be the last statement. it indicates the end of the function.

syntax: function functionname (parameters)
{
statements;
return value;
}
→ function prototype

ex: function f1(){
document.write("Hello");
}
f1(); → Hello.

parameters:

→ parameters are the inputs to the function
→ while calling the function we have to pass values to the parameters.

ex: <script>
function f1(id, name){
document.write(id, name);
}
</script>
<input type="submit" onclick="f1(10, 'Hello')" value="click">
o/p: 10Hello

ex: <script>
function fullname(fname, lname){
var full = fname+lname;
return full;
}
function f1(){
let result = fullname('ird', 'sharp');
document.write(result);
}
</script>
<input type="submit" onclick="f1()" value="click">

Nested Function

→ the function inside the other function is called Nested function.

Ex: <script>
function f1(a,b){
function square(x){
return x*x;
}
return Math.sqrt(square(a)+square(b))
}
function f2(){
let result= f1(4,5)
document.write(result);→
}
<input type="button" onclick="f2()" value="click">

→ by using Function() constructor we can create a function

syntax:
var referencevariable = new function(arg1, arg2, return value);
var f1=new function("x","y","return x*y");
var result = f1(4,5);
function
core
console.log(result); → 20.

→ string s="java" → using string literal
string s=new string("java") → using string class

creating the function with function literals

Ex: var result = function f1(x,y)
{
return x*y;
}
console.log(result(10,20)); → 200

passing array as a parameter to the function

→ older versions of JS we pass array as an function argument then we use apply() method null. we use null to make the code unclean.
→ By using spread operator (...) we don't use apply() method.
Ex: In java, ps vm (String ...java) → command line args

ex: function fn(a,b,c)
{
c.log(a);
c.log(b);
c.log(c);
}
var arr1 = ['Hlo', 'Hii', 'bye'];
fn.apply(null, arr1); → Hlo Hii bye
fn(...arr1); → Hlo Hii bye

→ react js is the extension of JS.
→ JS (older version) is object based & oriented