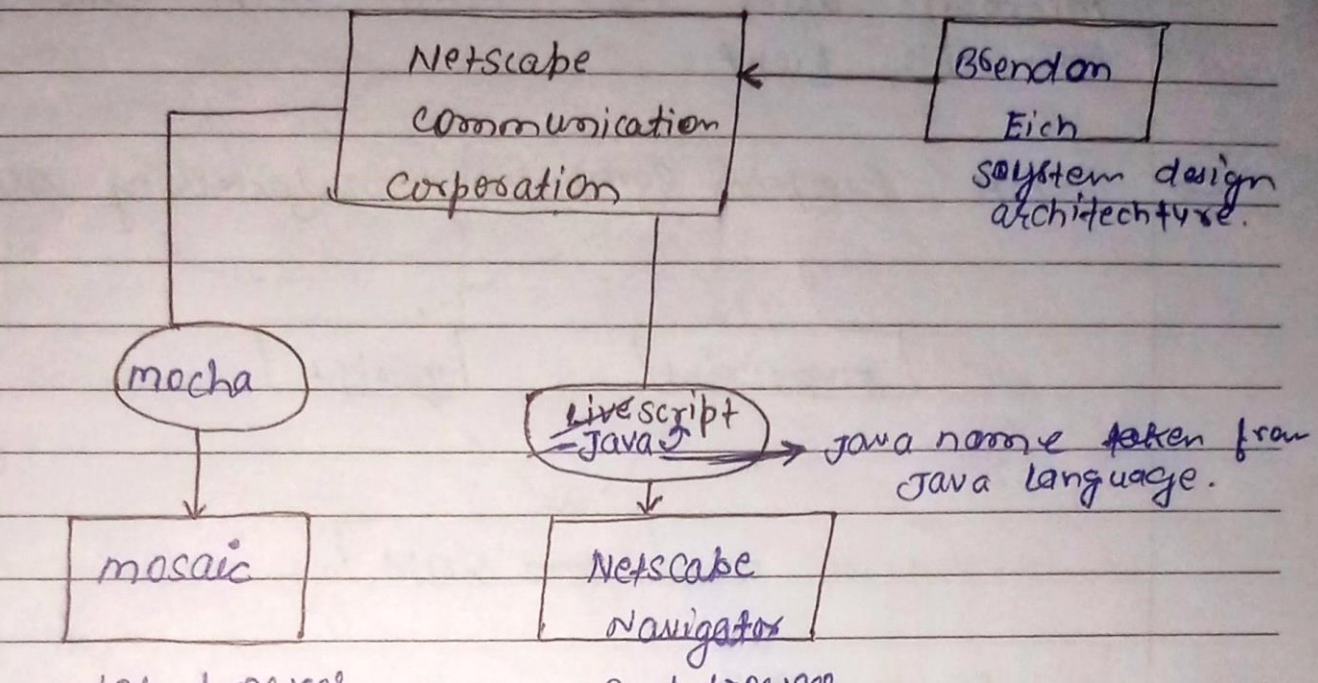
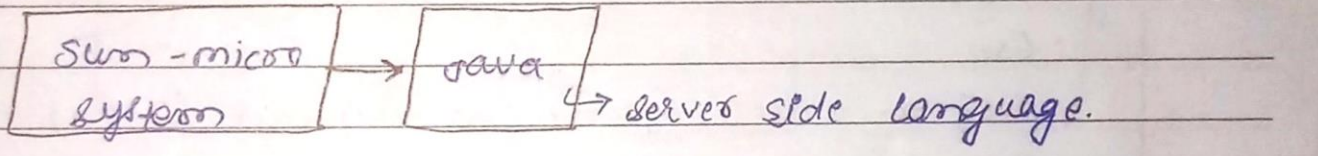
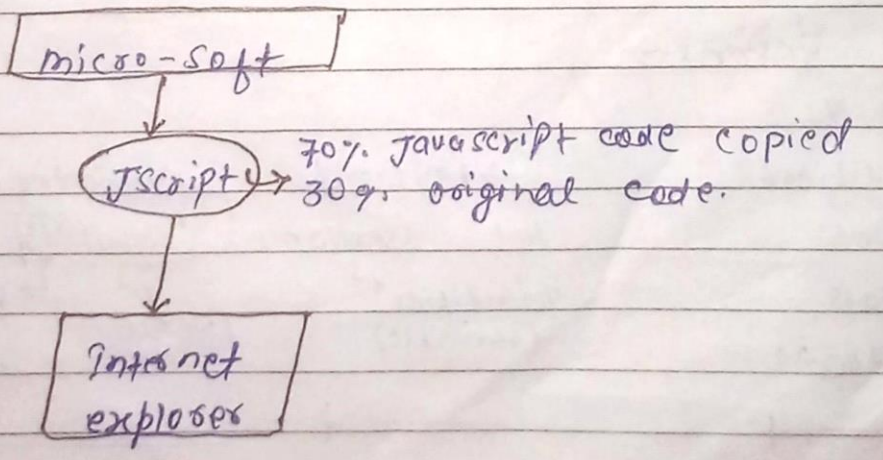
DATE 00

History of javascript:





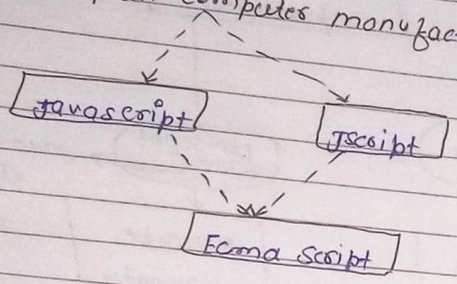
Note: At that time lots of fights happened b/w SMS and NCC. b/c of name taking without permission but later things are normal and sms allows to take his language name.



PAGE

Note: Again lots of fights happen b/w NCC and microsoft. after that ECMA script Come for solve this fight.

↳ ECMA (European Computer manufacturing Association) :



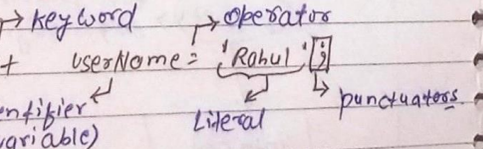
- ES1 → 1997
- ES2
- ES3
- ES4
- ES5 → class, object, function, var (2009)
- ES6 → 2015 : let, const etc.

Tokens :

It is smallest unit of programming language.

↳ Types of tokens →

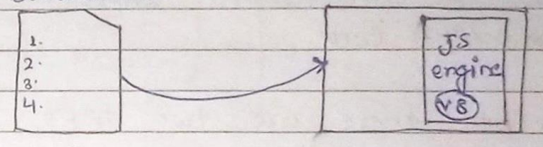
1. Keywords
2. Identifiers
3. Operators
4. Literals
5. punctuators



Javascript code executed where and how :-

* JRE (Javascript Run time) :-

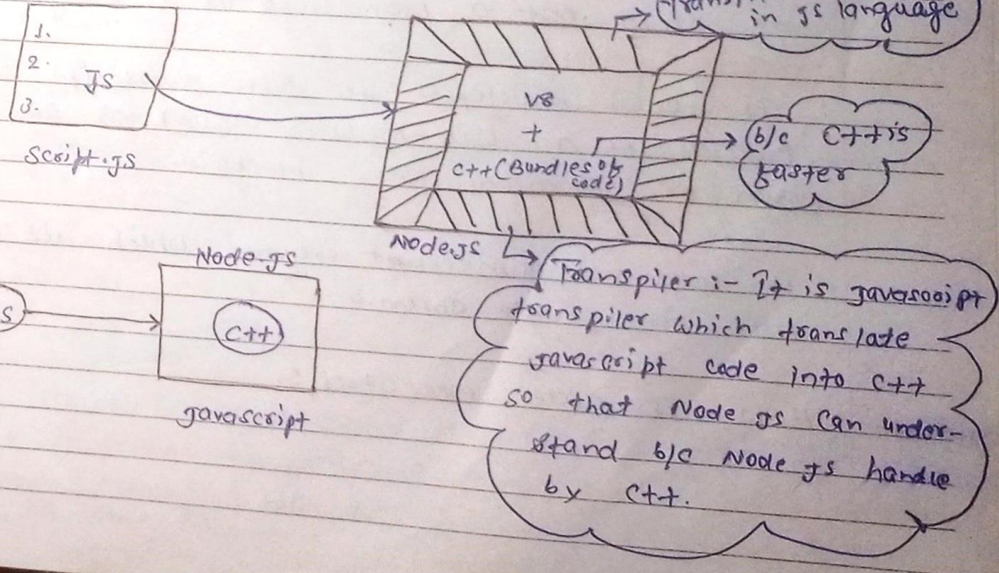
1. Browser →



↳ Types of engine in different browsers →

S.No	Name	JS engine
1.	Google Chrome	V8 (fastest)
2.	mozilla firefox	spider monkey
3.	Safari	Javascript core
4.	ms Edge	Chakra.

2. Node JS →



Code execution

JRE :-

JRE provides the environment where we can run our javascript code.

- In Javascript there are two JRE's,
  (i). Browser
  (ii). Node.JS

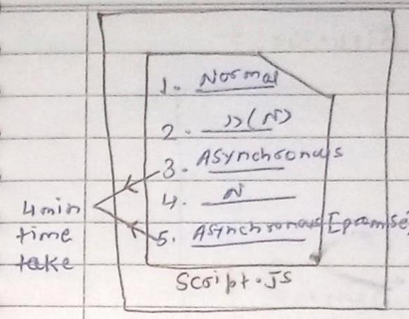
(i) Browsers →

- It is an application that is used to access the internet and view the information on WWW.
- It allows users to interact with web page, multimedia content, and surf the internet.

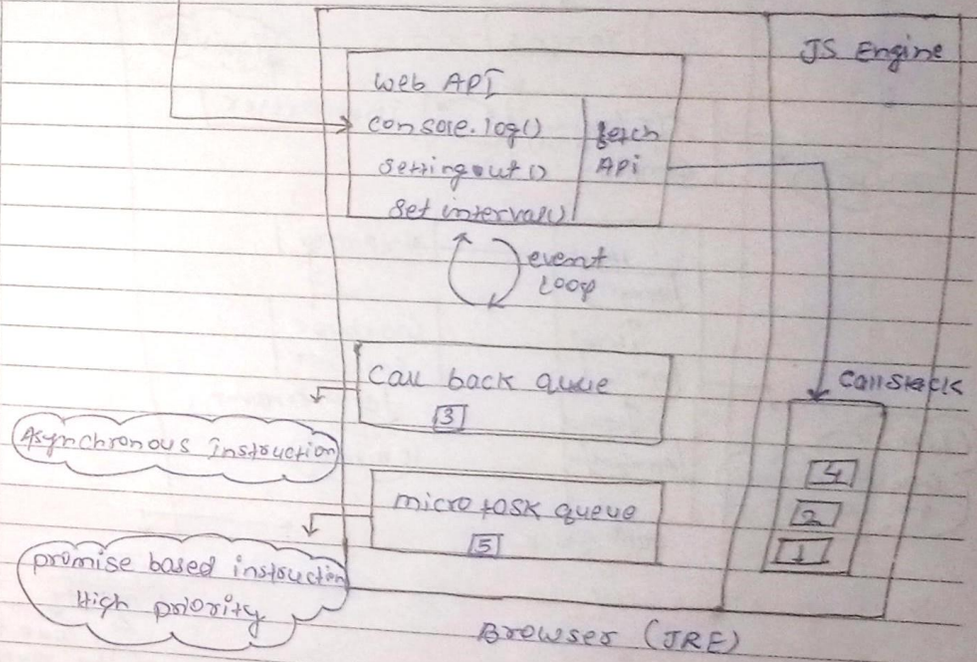
(ii). Node JS →

- It is a software application that executes JS code. It is not a framework or a library.
- It allows developers to run javascript code outside of a web-browser, such as on a server or command line interface.
- It uses v8 javascript engine which is also used in google chrome.

How JS code executed :-



VS code

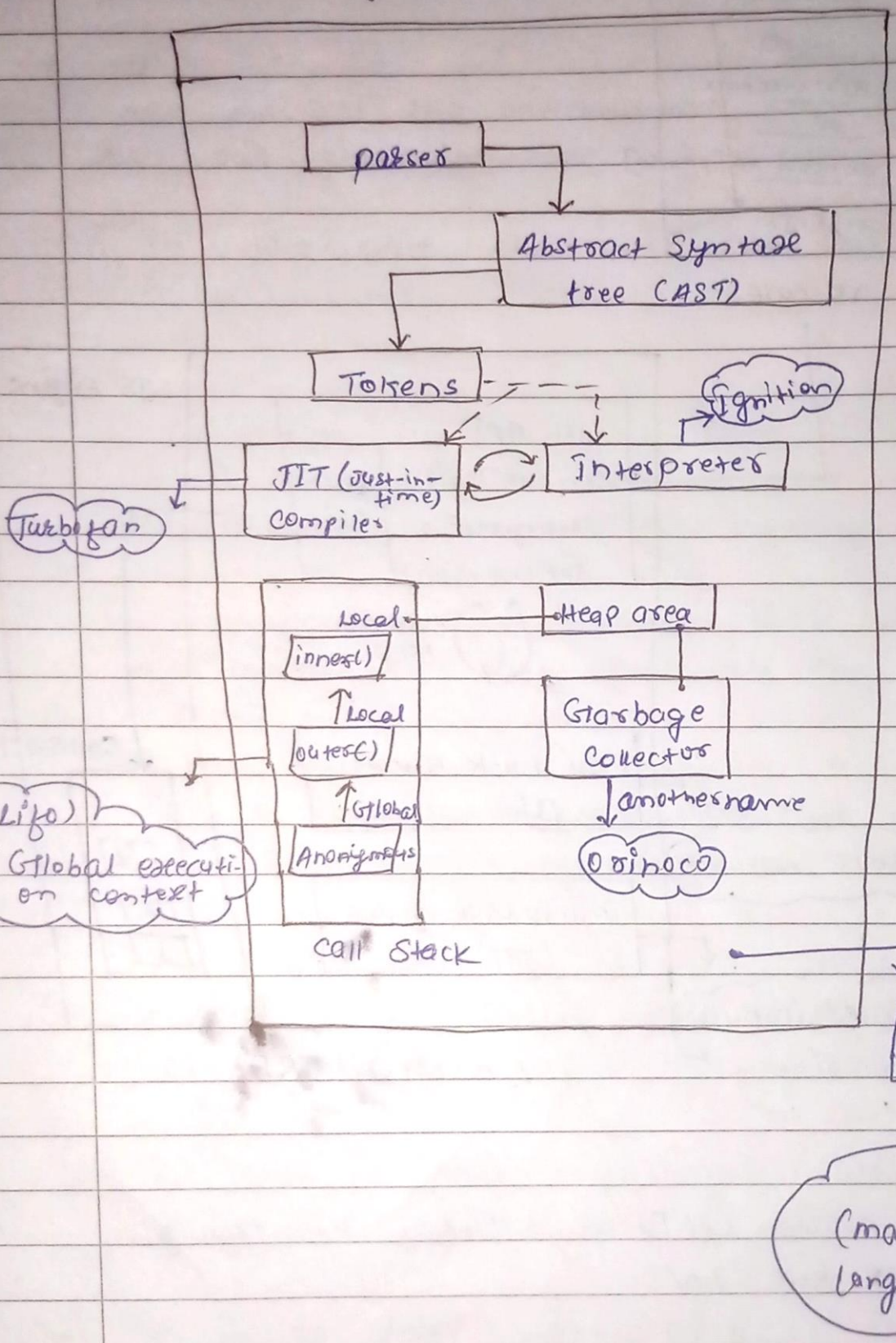


Note:- VS code, Browser (JRE) se Web-API ke through communicate krta hai.

- Web-API Asynchronous task to call back queue, Asynchronous promise task ko microtask queue and normal task ko call stack me distribute krta hai.

DATE

JavaScript Engine detail structure :-



Note: JS engine → • It is computer program that executes JavaScript code.

• It is core component of Web-browser, server side JS platforms and other JS base environment.

PAGE

Introduction to javascript :-

(i) JS is object-oriented and purely object-based language.

eg:- int x=5;	let x=5;
yaha pr, int x=5; ek variable hai	yahan pr ye variable bhi object hai.

2. JS is dynamically typed language →
• programming language that assigns a type to a variable at run-time based on the variable's value.

• Dynamically typed languages don't require any pre-defined date type for any variable.

3. It is an interpreted language →
It executes the code line-by-line.

4. JS is synchronous single threaded language.

5. JS is Scripting and programming language.
Already pre-written code
We are writing some logic from scratch.

6. JS is used to add functionality to our website.

Types of JS / ways to write JS code :-

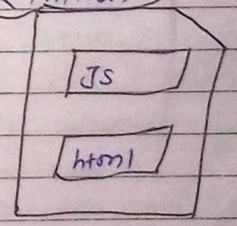
1. Inline javascript :-
It is applied as an attribute inside opening of any tag.

Syntax →
Login

2. Internal javascript :-
It is written inside body with the help of <script></script> tag.

Syntax →
```
<html>
  <head></head>
  <body>
    <p>This is internal JS structure</p>
    <script>
      alert('Namastey...!')
    </script>
  </body>
</html>
```

Looks like →



• Html

DATE

3. Exetesnal JavaScript :-

Syntax:

```
<html>
  <head> </head>
  <Body>
    -
    -
    -
    -
  <script src=" path " > </script>
  </Body>
</html>
```

eg: one.js

→ Absolute (From Root)
→ Relative

DATE

How to print any statement in js :-

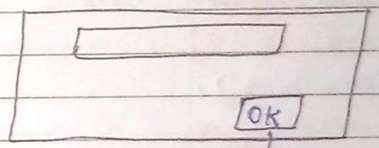
1. document.write (" Gigaan hai aap ... ?") →

↳ means, html file → Iska mtlb go inside html file and write gigaan hai aap there.

2. alert (' do you want to share your location') →

↳ popup

eg:- let result = alert (' Hello')
c.log (result)
↳ gives output = undefined.



↳ alert me ye pop up show hota hai, fir OK pr click karne pr ek value return hoga jo, result object me store hoga.

3. console.log (" Hello developers") →

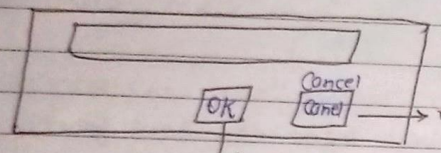
- console.log me log ek method hai jo console ke andr "hello developers" ko write krta hai.

- Console ek area hai chrome ke js engine ke andr.

4. confirm (' Hello ');→ (→ popup, → optional)

eg:- let result = confirm ('Do you want to share your location')
console.log (result);

↳

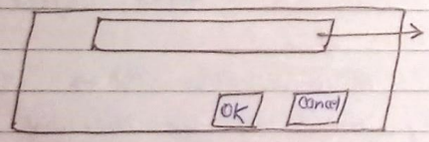


ways to take input from user :-

1. prompt (' Enter something');

eg:- let username = prompt ('Enter your full name');
console.log (username)

↳



Note:- prompt hamesha input ko string me hi leta hai.
Baad me ham usko fir type conversion krke dosre ya uske actual type me convert karte hai.

eg:- (i). let a = prompt ('123')
console.log (typeof(a)) → string

(i). let userId = prompt (' Enter your userID')
console.log (typeof (userId)) → convert string to numbers
↳ number

(ii). const a; ✗
const a=5; ✓
const ko sirf declare nahi krke side me kar sakte hai

② let & var :- let ke case me js engine undefined execution phase me store karta hai, and var ke case me js engine undefined variable ke andr variable phase me store karlata hai.

Keywords :-
Keywords are pre-defined words which is used to perform some specific task.

eg:- for, while, do, break, if, else, continue, try, catch, finally ..etc

Diff b/w var, let and const →

var (Aworo)	let (Pagal)	const (Deewana)
var gf = 'Naina';	let gf = 'Urvashi';	const gf = 'Anushka';
var gf = 'Sanjana';	let gf; ✗ can't happen	const gf; ✗ twice declaration of same variable can't happen here.
var gf = 'Sonam';		
① multiple time declaration of some variable.	① One time declaration.	① One time declaration.
② multiple time initialization of same variable with diff value.	② multiple time initialization eg:- let gf = 'Urvashi'; gf = 'Sanjana';	② One time initialization. eg:- const gf = 'a'; gf = 'b'; ✗ Can't initialize more than one time
③ global scoped	③ block scope.	③ block scope.

Identifiers :-

- Identifiers are user given names.

- Identifiers are the names of a class, name of a variable, name of a method, name of an interface, name of a package or name of any other class members.

* Rules of an identifier :-

1. An identifier can be written by using alphabets (A-Z or a-z), numbers (0-9) and only two special characters ($ and _).
   dollar ↵ ↳ underscore.

2. An identifiers should not starts with a number.
   eg:- var 1name = 'Ram'; → ⊗
   ↳ wrong

3. Any keyword must not be used as an identifier.

4. Any space is not allowed for an identifier.
   eg:- mob_NO = 123;
   ↳ can't write/use space.

* Conventions for an identifier :-
Conventions are set of guidelines for good programming practice.

→ convention for a variable name →
1. one word variable name = write in small letters.

2. more than one word;
   (i) my_name = 'mahtab';
   (ii) Camel casing, myNameIs = 'mahtab';

Operators :-
Operators are predefined symbols which is used to perform some specific tasks.

ex→ +, −, *, /, %, **, =

↳ Operands :-
Operands are value on which operation performed using diff. operators.

ex→ c.log(10 + 10);
↳ operand-1 ↳ operand-2

* Types of operators :-

(1). Unary operator :-
(i). Increament / decreament (++, --)
(ii). Logical Not (!)

(2). Binary operator :-
(i). Arithmetic operator (+, −, *, /, %)
(ii). Assignment operator (=)
(iii). Compound assignment operator (+=, -=, *=, /=, %=).
☆ (iv). Relational operator (>, <, >=, <=, ==, !=)
↳ strict equality (===)
↳ strict not equality (!==).
(v). Logical operator (&&, ||, !)
Logical and ↵ ↳ Logical OR or double pipeline.

3. Ternary operator :-
(i). conditional operator.

operand-1 ? operand-2 : operand-3;

DATE

2(i). Arithmetic operator :-
+, −, *, /, %, ** (power)

Ex→ (i) c.log(5 a**b 2);
⇒ 5^2 ⇒ 25

(ii) c.log(10 a%b 2);
⇒ 0

divisor / dividend
2) 10 (5 ↳ quotient (1)
0
↳ Remainder (%).

↳ Some special cases where we (+) operator two work at a time.
(i) It work as addition (to perform)
(ii) It also work as concatenation (to perform)

eg:- (a). c.log('Hello' +5);
Hello5

(b). c.log(5+5+'Hello' +5+5);
10 + 'Hello' +5+5 ⇒ '10Hello'+5+5 ⇒ '10Hello5'+5
⇒ 10Hello55

(c). c.log(2+3+'Dev' + (3+2) +5);
(first solved this)
⇒ 2+3+'Dev'+5+5 ⇒ 5+'Dev'+5+5 ⇒ '5Dev'+5+5
⇒ '5Dev5'+5 ⇒ 5Dev55.

(d). c.log('Ram' - 10);
⇒ Ram - 10
↳ Isko number me convert kr dega.
⇒ NaN (not a number).
↳ Kr pta lagega ki Ram to numbers hai hi nahi pr typeof number hi dega.

DATE

2(ii) Assignment Operator :-
Used to assign the value to LHS after performing calculations on R.H.S.

eg:- let a = 10+10 ; (20)
↳ pehle RHS calculate hoga fr value LHS me assign hogi.

2(iii). Compound assignment operator :-
It is a combination of Arithmetic and assignment operator.

eg:-
+=
−=
*=
/=
%=

↳ (i) let a = 10;
a += 20; { a = a+20; ⇒ a = 10+20; ⇒ a = 30; }
c.log(a);
↳ 30.

2(iv). Relational operator :-
It perform comparison and return, Boolean values → true or 1
→ false or 0

↳ >, <, >= (minimum), <=, == (maximum).

eg:- (i). c.log(10>5); true.
(ii). c.log(2>=2); true.

DATE

↳ strict equality :-

```
let a = '10';
let b = 10;
```

Type conversion happened

(i) c.log(a==b) → true

(ii). c.log (a===b) → false.

↳ It prevents automatic / implicit type conversion.

more on Relational operator :-

<, >, >=, <=, == → Type coersion (Type conversion.).

↳ Loose / Abstract equality.

Ex:- (i). let a = '10'; // string
let b = 10; // Number

⇒ c.log (a >= b)
string Number
↓
Number

⇒ c.log ('10' >= 10)

⇒ c.log (10 >= 10) → Ans = true.

loose equality :-

⇒ (==) It doesn't do conversion of null and undefined.

(ii). let p = null;
let q = 0;

It does not consider their type. means data ka diff type value me equal ho sakte hai yahan.
eg. 1 == '1' → true

DATE

⇒ c.log (p==q) → Number
↓
null

⇒ c.log (null == 0); → Number
↳ null → In the case of abstract equality null and undefined do not convert into number or can't happen any other type conversion implicitly.

⇒ false

(iii). let p = null;
let q = 0;

⇒ c.log (p >= q) → Number
↓
null
↳ convert it into number.

⇒ c.log (null >= 0)

⇒ c.log (0 >= 0) → Ans = true.

Note:- When converted to integers, both undefined and null becomes 0, because undefined is converted to NaN, which also becomes 0.

⇒ strict equality (===) :-
• ye type conversion nahi karta hai.

syntax :- Operand-1 === Operand-2
↳ Check type of data / data type.

PAGE

DATE

Exe :- (iv) let a = '10';
let b = 10;

⇒ c.log(a === b);

⇒ c.log('10' === 10); → Ans = false.
string number

(v). c.log(true === 1) ⇒ false
c.log(null === 0) → false
null number

⇒ (!=) Not equal to :-

Exe: (vi). let a = '10';
let b = 10;

⇒ c.log(a != b)
string number
conversion happen
Number

⇒ c.log('10' != 10);
⇒ c.log(10 != 10); → false

⇒ Strict not equality (!==) → type of data/ data type.
(vii). let a = '10';
let b = 10;

⇒ c.log(a !== b);
string number

DATE

⇒ c.log('10' !== 10) → true

Note ⇒ Read like this, string (ins) 10 is not equal to number 10 in numbers.

(2(v)). Logical operator :-

(1). Logical AND (&&) :-

Operand-1 && Operand-2

↳ Relational Operators

↳ Relational operators.

Operators ↳ kyuki, operand-1 and operand-2 ki value boolean me honi chahiye logical and, or, not me. And Relational operator hi hai jo boolean values ko return krta hai.

Ex:- (i). let a=5;
let b=10;
let c=15;
let d=20;

let result= a>b && c<d;
↳ op-1 ↳ op-2
↓ ↓
false true.

c.log(result) → false

DATE

↳ Logical and output table :-

S.NO	Operand-1	Operand-2	Result
1.	true	false	false
2.	true	true	true
3.	false	true	false
4.	false	false	false.

(2). Logical OR (||) :- Double pipeline.

Ex:- (ii). C.log (5>10 || 15<20) → true
↳ operand-1 ↳ operand-2
↓ ↓
false true

↳ Logical OR output table :-

S.NO	Op-1	Op-2	Output
1.	True	true	true
2.	true	false	true
3.	false	true	true
4.	false	false	false.

(3). Logical Not (!) → Unary operator

Ex:- (iii). C.log (! (5>2));
↳ true
convert opposite = false

↳ table :-

S. NO	Op-1	output
1	true	→ false
2	false	→ true

PAGE

DATE

Increment / Decrement operator :-
(++) (--)

1. pre-increment :- first increment then perform operation.

(i). let a = 1;

(ii). ++a ; (1 + 1)

(iii). a = ++a ; (1+2=3)
LHS RHS → pehle right hand side ko solve karo fr assign.

(iv). C.log (a);
↳ 3

2. pre-decrement :- Kind of same like pre-increment.

(i). let b = 5;

(ii). -- b; (-1 + 5 = 4)

(iii). b = -- b ; (-1 + 4 = 3)

(iv). C.log (b);
↳ 3

3. post increment :-
first assign or perform operation then increase.

PAGE

DATE

(i) let a = 1;

(ii) a++;
1 + 1 = 2

(iii) a = a++;
2 + 1 = 3
↳ because, a post increment ko dekh nahi pata LHS wala hai. usko lagta hai 'a' ke andr 2 hi hai; RHS wale

(iv) c.log(a);
↳ 2.

4. post decrement (--) :- Same like post-increment.

(i) let x = 5;

(ii) x--;
5 - 1 = 4

(iii) x = x--;
4 - 1 = 3

(iv) c.log(x);
↳ 4

Ex:- (i) let a = 1;

(ii) a = a++;
1 + 1 = 2

(iii) a = a++;
1 + 1 = 2

c.log(a++);
↓
1 x + 2

DATE

(iv) a = a++;

(v) a = a++;

(vi) c.log(a);
↳ 1

(vii) c.log(a++);
↳ 1 (Iske baad a increment ho jayega).

(viii) c.log(a);
↳ 2

a	b	c
2	2	7

(ii) ① let a = 1;
② let b = 2;
③ a++;
1 + 1 = 2
④ b--;
2 - 1 = 1
⑤ let c = ++a + b++ + a--;
1 + 2 = 3 ; 1 + 1 = 2 ; 3 - 1 = 2
3 + 1 + 3 = 7
c can see these values
⑥ c = c--;
7 - 1 = 6
⑦ c.log(a)
↳ 2
⑧ c.log(b)
↳ 2
⑨ c.log(c)
↳ 7

PAGE

DATE 20 03 2024

Ternary Operator :-

(ii). Conditional Operator :-

var result = OP-1 ? OP-1 : OP-3 ;

OP-1 → True/false → Boolean value

Relational Operator

Logical Operator

Ex:- const a = 10; (Greater among three.)
const b = 20;
const c = 30;

const result1 = a>b ? a : b;
(true / false)

const finalResult = result1 > c ? result1 : c;

c.log (final_result);
→ 30

Note:- Agr variable ki value print krwani ho to :- backtiks ke andar ${variable name}

year%100==0 && year%4==0
|| year%400==0 ? 'leap year : ${year}' : 'not a leap year : ${year}';

DATE

Literals :- Literals are the values which is used in writing a program.

Ex:- let username = 'manny';
Literal

★ Types Of Literals / Data type :-

(1). primitive Literals :- Single value Literals / Immutable (can not change) →

- In javaScript, a primitive Literals is a single value literal.
- javaScript treats primitive values are immutable value (value that can't change).

Ex:- (1). Single value :-
var userId = 2049;

userId
2049

(2). Immutable (value that can't change) :-

let my_fav_song = 'Brown Rang';

myFavSong = 'vigdiyan heeran';

myFavSong → 2256 → Brown Rang → garbage collected
address of this memory block

vigdiyan heeran
2257

- Brown Rang wali memory block garbage value me badal jayegi. myFavSong vigdiyan heeran wali address ko Point karne lagega.

- myFavSong ——→ ~~Brown~~ Rang vigdiyan heeran.
  - ↳ agr isi memory block me new value ko overwrite karte to wo mutable hota.

(2). Non-primitive literals :-
multi-valued literals (mutable can change).→

↳ trick to identify Non-primitive literals,
Brackets ⇒ ( ), { }, [ ]

- In javascript, a non-primitive literals is a multi-valued literals.

- javascript treats non-primitive values as new table values (value that can change).

- Non-primitive literals are object, array, etc.

Ex:- (1). multi-valued literals→

(i) let courses = ['html', 'css', 'js'];

courses

↳ | 'html' | 'css' | 'JS' |

(ii). let arr = ['Hello']; → this is also multi-valued.

Type of primitive literals :-

1. Number :-
- It represent numeric value or floating value.
- Its range is $2^{-53}-1$ to $-2^{53}-1$

eg:-
```
let age = 24;
let price = 559.55;
```

2. BigInt :-
↳ if number literals exceeds its limit then we can store that as a bigint value.

↳ Its range is $-\infty$ to ∞.

↳ we use 'n' as a suffix value.

eg:-
```
let age = 25n;
```

3. NaN (not a number) :-
↳ It is computational error given by JS engine

eg:-
```
let user = 'Ram';
let x = 2;

c.log (user - x);
```
user → string → number; x → number

Ram − 2 = NaN.

JS engine ne 'Ram' ko number me convert kr diya pr fr usko pta laga ki ye to number hai hi nahi. isiliye NaN output aaya.

→ Single line String :-
(1). It is enclosed (packed) by single or double quotes.
(2). It does not allow line break or white spaces (blank space) (10 or more blank spaces using tab = It consider only one blank spaces).

4. Boolean :- ↳ It represents a logical entity.
↳ It can true or false.

eg:- let isWorking = true;
let isLogged = false;

5. String :- collection of characters →

(1). Single line string → let userName = 'Chintu'; (single quote)
→ let lastName = "xyz"; (double quote)

eg:- let name = "Hello everyone, kem chho?";

(2). multi-line string :- (Template String) →

↳ Its writen inside ` ` (backticks symbol).

↳ We can also print value inside variable (let, var, const) with rest of string but it must be written inside ` ` (backsticks symbol). It also called interpolation → ` ${var_name} `.

eg:-
(i). multi-line string Example →

let meme = ` Khatam
Tata
Bye Bye
good Bye ` ;

(ii). Interpolation example :-

DATE

let userName = prompt('Enter your full name');
c.log(` How are you ${userName} ?? `);

↳ points :-
↳ It is enclosed by backticks.
↳ It allow line break and white spaces (here to blank or white space = to white space).

(6). NULL :-
↳ Isme hame koi exact value nahi pta ho and hame variable ko initialize karna hota null se karte hai.

eg:- let newEmp = 'Akshay'; (newEmp → Akshay)
let salary_of_newEmp = null; (salary_of_newEmp → null)

7. undefined :-
↳ Js engine store karwata hai ags variable me hamne kuchh v store nahi karwaya hai.

eg:- let newEmp = 'vivek'; (newEmp → vivek)
let salary_of_newEmp; (salary_of_newEmp → undefined) (Js engine)

DATE

8. Symbol :- (Symbol()) → (S = capital)

↳ Symbol hamesha unique banata hai value ko convert karke kisi value ko chahe wo values same hi ho.

eg:- (i) normal example →

(a) let a = Symbol() ;
c.log(a);
↳ output = Symbol()

(b) let b = symbol('Hello');

* c.log(b) → Symbol(Hello)
* c.log(b.description) → Hello

(ii). Same value inside Symbol is not equal but these description is equal →

let a = Symbol('Namastey dev'); let b = Symbol('Namastey dev');

eg: * c.log(a === b) → false

* c.log(a.description === b.description) → true

DATE 26 03 2024

punctuator :- These are also a special symbols [{}, [], ()] used to group the code or separate code.

↳ Grouping Symbols → { }, [ ], ( )

↳ Separation Symbol → , ;

↳ Assignment Symbol → :

↳ To access key value pair inside an object → . (dot)

eg:-

let obj = { → P
username : 'chambu' ;
↳ P
}

c.log(obj.username) ;
↳ punctuator ↳ P

What is Js. — (1)
Type of Js. — (2)
Input & output — (3)
Keyword — (1)
Identifier — (2)
Operator — (3)
Literals — (4)
Punctuators — (5)

(1) (2) (3) (4) (5) } Basic components

Flow control statements — (5).
Type coersion — (6)
Loop — (7).

DATE

(5). Flow-control statements :-

1. if
2. if-else
3. else-if
4. Nested-if.

1. if :-

```
let num = +prompt('Enter the no')

if (num === 10) {
    C.log ('Hello World');
}

C.log ('Outside Block');
```

2. if-else :-

```
let num = +prompt ('Enter the no')

if (num === 10) {
    C.log ('Hello Hi');
}
else {
    C.log ('Bye');
}
```

3. if-else if-else :-

```
if (num === 10) {
    C.log('Hello');
}
else if (num === 8) {
    C.log ('Hi');
}
else { C.log ('Bye'); }
```


DATE

(6). Type coercion / type conversion :-

Number ()
Bigint ()
String ()
Boolean ()
Symbol ()

↳ The process of converting one type of data into another type of data by js engine is called type coercion.

* Type of type coercion :-

(1). Implicit type coercion → The process of converting one type of data into another type of data by js engine automatically is called implicit type coercion.

eg- let a = '10';
let b = 5;

C.log (a == b); or C.log (a - b);

string → Number

js engine do conversion into number automatically

Here also same thing will be happen.

(2). Explicit type coercion :- process of converting one type of data into another type forcefully by js engine is called explicit type conversion.

↳ typeof operator →

- Keyword and unary operator.
- used to get/check data type of literal.
- It returns type of literal in string format.

eg:- ① let a = '5';

c.log(typeof (a); or c.log (typeof a);
↓ string ↓ string

↳ (type of hamesha value ko String me return krta hai.)

② let b = 5;

c.log(typeof b);
↓ Number

↳ (ye v string hi hai jo Number likha hua hai)

↳ Examples of explicit type coercion :-

(i). diff type into Number →
ii). String into Number,

let num = '10';

1st way ⇒ let output = Number (num);
2nd way ⇒ let output = +num;

c.log (output); → 10
c.log (typeof output); → Number.

(ii). Character value into number,

let userName = 'Rahul';

const res = + userName;

c.log (res); → NaN
c. log (typeof (res)); → Number.

(iii). Boolean into Number,

const isWorking = true;
const res = Number (isWorking);
console. log (res); → 1
co.log (typeof res); → Number

2. Conversion into boolean :-

(i) Number into boolean,

case1:
let a = 0;
c.log (Boolean (a); → false
c.log (typeof a); → boolean

case2:
(0 -1, -0.1)
let a = 0.1;
c.log (Boolean (a)); → true
c.log (typeof a); → boolean

Note: false at → 0
true at all except 0 → 0.1, 1, -1, -20

(ii). String into boolean,

(a) const a = 'a';

DATE

```
const res = Boolean(a);
c.log(res); → true
c.log(typeof res); → boolean
```

(b). let a = " ";
↳ one space.

```
c.log(Boolean(a)); → true
c.log(typeof a); → boolean
```

let a = "";
↳ no space.

```
c.log(Boolean(a)); → false
c.log(typeof a); → boolean
```

↳ false = '', empty string.
↳ true = ' ', 'a', 'abc'.

3. Conversion into Bigint :-

(i). Number into Bigint,

```
let num = 10;
c.log(Bigint(num)); → 10n
c.log(typeof num); → bigint
```

(ii). String into Bigint,

```
let num = 'abcd';
c.log(Bigint(num)); → error
c.log(type of num);
```

But,

```
let num = '10';
c.log(Bigint(num)); → 10n
c.log(typeof (num)); → bigint
```


DATE

4. Conversion into Symbol :-

(i) Numbers into Symbol,

```
let num = 1; const res1 = Symbol(num);
c.log(res1); → Symbol(1)
c.log(typeof res1) → symbol
```

5. NULL into diff type :-

```
const salary = null;
c.log(typeof salary); → object
```

① Number, c.log(+salary) → null = 0
② Bigint, c.log(Bigint(salary)) → null = error
③ String, c.log(String(salary)) → null = null
④ Boolean, c.log(Boolean(salary)) → null = false
⑤ Symbol, c.log(Symbol(salary)) → null = Symbol('null');

6. convert undefined into diff type :-

```
let age;
c.log(age); → undefined
c.log(typeof age); → undefined
```

① Number, console.log(+age) → undefined
" (typeof age) → undefined

② Bigint, c.log(Bigint(age)); → error.
" (typeof age);

DATE

(iii). String, c.log (String(age)); → undefined
c.log (type of age); → undefined

(iv) Symbol, c.log (Symbol(age)); → Symbol()
" (type_of age); → undefined

7. Loop :- Loop means something that happens in sequence like (2, 4, 6, 8) Here value inc. by 2 every time.

Types of loop :-

(1). Entry control loop →
(i). while
★ (ii). for.

(2). Exit control loop →
(i). do while.

(ii) While loop :-
(a). first initialize
(b). check Condition
(c). Increment / decrement

Ex :- (1). let num = 1;
(2). while (num <= 5)
(3). {
(4). console.log(num);
(5). num++;
(6). }

output :- 1 2 3 4 5

(ii). For loop :-

```
For (let i=1; let j=1 ; i<=100 ; i++, j++)
{
}
```

Condition (Boolean)
variable decalaration & initialization
increment / decrement

DATE

(2). Exit control Loop:-

(i). Do while Loop →

(a). Initialize

(b). Increament / decreament

(c). condition check.

eg:- let i=1;

```
do {
    console.log(i); i++;
} while(i<=5);
```