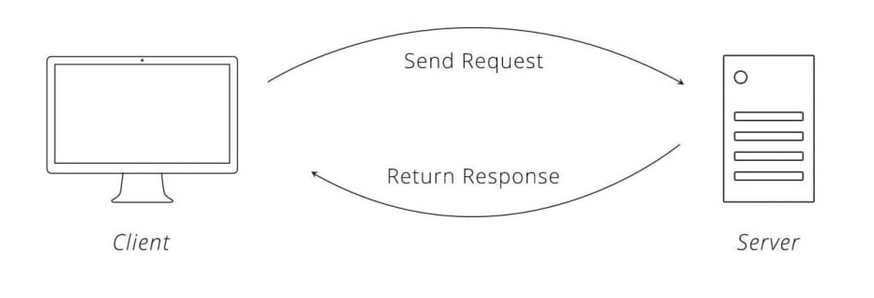
***JAVASCRIPT***

-only programming language understood by your browser

**Programming language**: way of communication through writing your own logic

**Browser/client** :firefox, brave, chrome,etc

**Server:** network computer, computer program, or device that processes requests from a client.

***Request-reponse cycle***

**Primitive/primitive data**: smallest possible division ,which cannot be further broken down (building block of your program).

for ex- sahi paneer ki sabji banani hai to ***milk*** as a building block hai uske liye to wo as primitive data ho jayega.

Q-1: **how many primitive data in javascript?**

Ans- there are 7 types-

1. Number
2. Boolean
3. String(char/string)
4. Undefined
5. Null
6. Symbol
7. big int

Q-2: **what is variable?**

Ans: not a container, variable is a name which holds some value.

For ex. let age=23;

let umar=Akmal;

**Note**: there are three type of declarative which helps define varbiable are-

1. var
2. let
3. const

**Nomenclature**: variable ko kaun kaun se name de skte hai.

Rule for define variable

\*(a-z),(A-Z),(0,1,2…9)

\* \_(underscore), $(dollar)

\* number se start nhi kar skte name ko

\* prefined words ko as a variable use nhi lar skte

\*special character me bss (\_ ,$) use kar kte hai name ki starting me

Ex: let mann=10;(possible)

Let man2=4;(possible)

Let – =Aryan;(possible)

Let 2man=Aryan(not possible we cannot start with no.)

Let break =Aryan(not possible because it is predefined word)

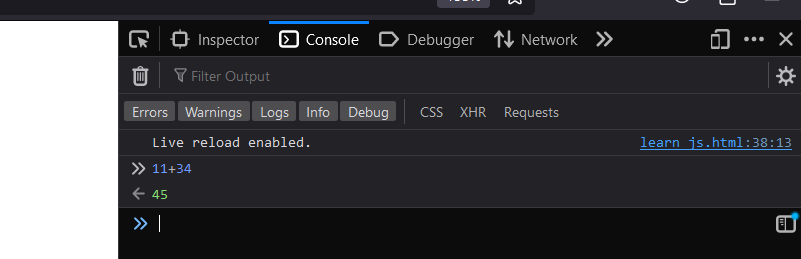
**Note:** varible kitna bhi bda ho skta hai but ***readability*** sahi nhi hoti bde name ki .

For better readability we have ways of defining variable-

* Normal case: ex- Let venugopaliyer=”goa”;
* Camel case: ex- let venuGopalIyer=”goa”;(enchance readability )
* Snake case: ex-let venu\_gopal\_iyer=”goa”;(good way it is not preferable as much as camel case)
* Kabab case: ex- let venu-gopal-iyer=”goa”(it not in javascript kyuki hum hypen use nhi karte variable ke name me).

\*\*Q-3: what is console ?

Ans : REPL(R stands for read, e stand for evaluate,p stand for print ,l stand for loop).



*Kaise console read , evaluate, print and loop check kar rhe hai is image me.*

Note : two type language –

* Interpreter: line by line excute karta hai
* Compiler: ek .exe file banta then usko excute karta hai

**JAVASCRIPT:**  IT IS AN *INTERPRETER LANGUAGE* , *WEAK TYPE*, *DYNAMICALLY TYPED* , *SINGLE THREADED*.

**Weakly typed**: where the data type(primitive)of the variable is not fixed.

ex: let a=17;

or (data type fixed nhi hai kuch bhi de skte ho)

**Strong typed:**

(c++ is strongly typed language)

**Dynamically typed:** aese programming language jinka data type *runtime* par define hota hai .

**Static type:**

**Single threaded:** only one task is possible at one time , multiple tasks cannot be performed.

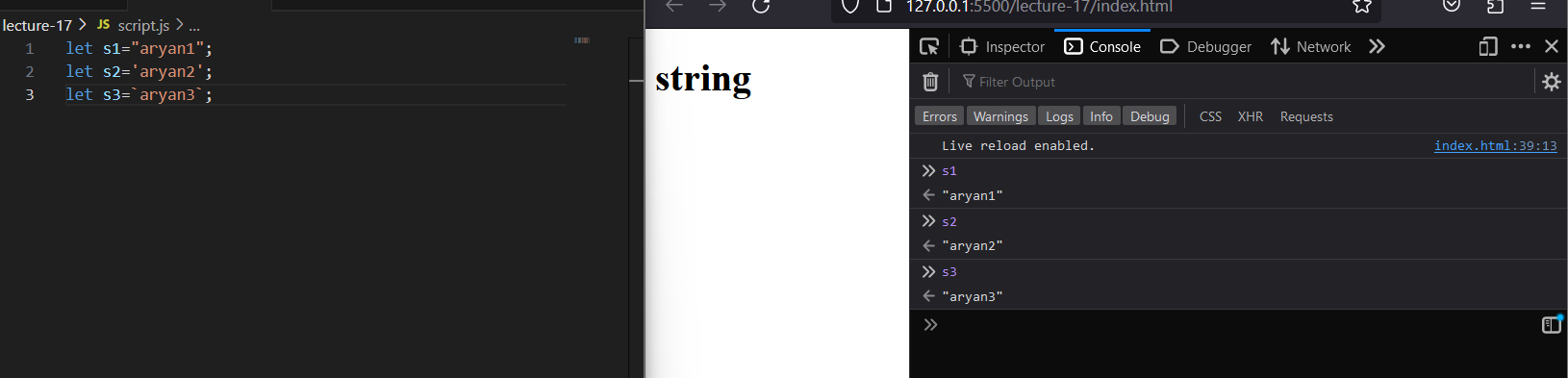
**String:** for ex: let var\_name=”aryan”; (it is only point the Aryan not hold the aryan)

**Note:** jo value hai bo memory mai stored ho jati phir variable use point karta hai.

**Ways of string define in JS-**

i.**single inverted**: for ex-let a=’gla’;

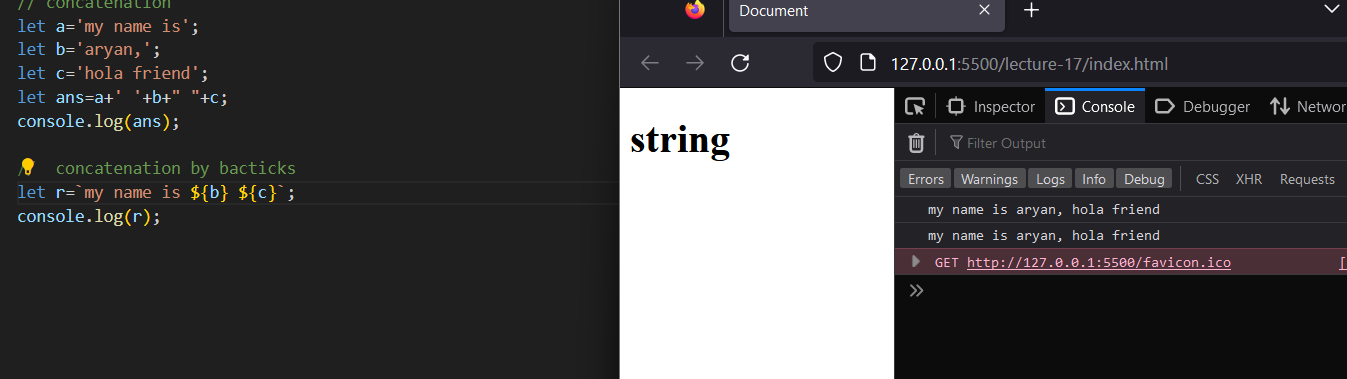
ii. **bacticks**: for ex- let a=`gla`;



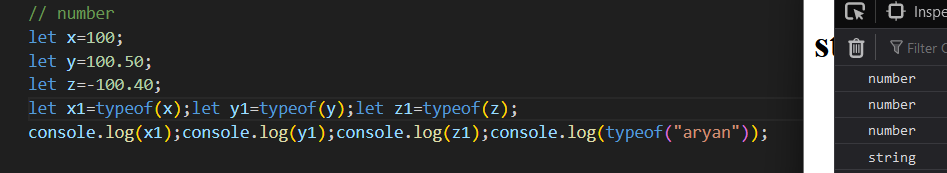
**Note:** semicolon se line terminate ho jati hai .

**Concatenation:**  string ko jodna (linking)-

**Note:** kebal bactick hi ${} use karke hum variable add karte hai.



**Number:** typeof() is used to check the type of variable;



**Note:** all operation is valid on number like ( +,-,\*,/,%,\*\*,etc).

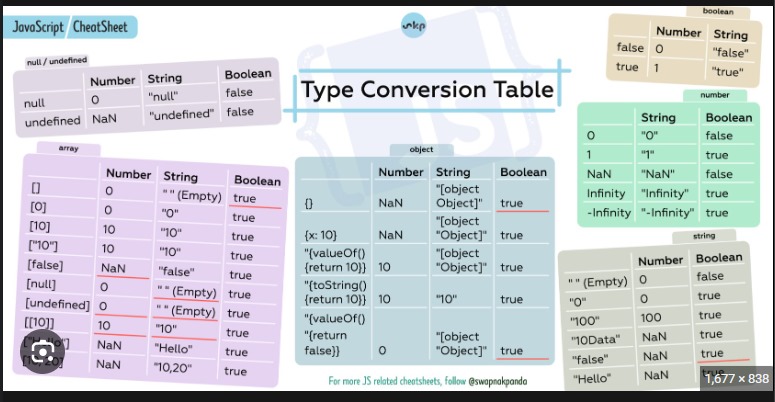
**Order of evaluate the operation:** (PEMDAS) p stand for parenthesis, e stand for exponent , m stand for multiplication, d stand for division , a stand for addition ,s stand for subtraction;

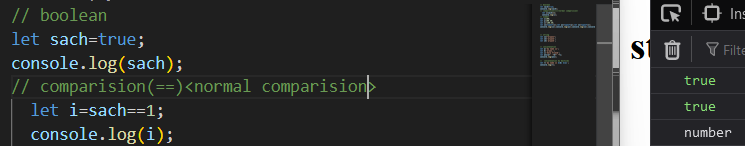
**Boolean:** for ex- let sach=true;

**Comparision :** there are two type equality-

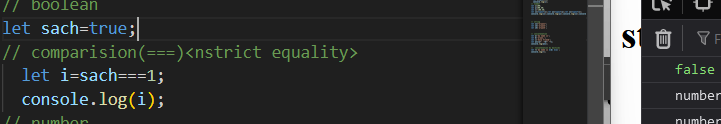
1. **General equality/normal comparision(==)**: Do not compared data type .

And typingcasting hoti hai.

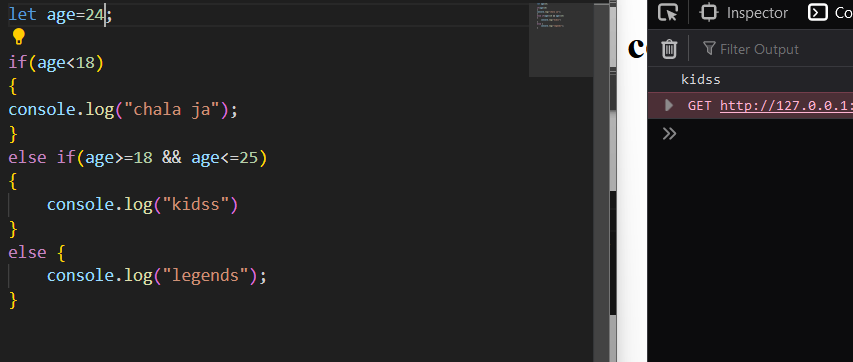




1. **Strict equality (===):**  isme type casting nhi hoti but data type check kiya jata hai .

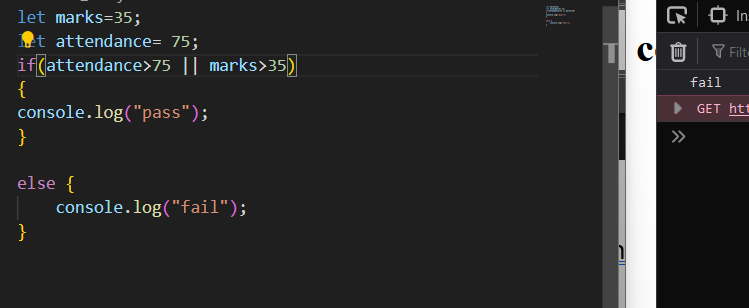
****

**If, else if, else:** use the for condition example are given below-

****

**Note:**  or operator(||), And operator(&&);

Mtlb short cicuuiting hoti hai js me bhi in case of or ,and operator.



**Note:**  not operator is used to like normal not operator in js.



Browser has two engine –

1. Js engine
2. Layout engine

For ex- let kaju; isme js engine kaju ko undefined data type ki taraf point karta hai

Hum dete hai let kaju =null( primitive type);

**Object:** it is non linear data structure having collection of properties .

**For ex:** let person ={name:”kaju”,age:1, color:”red” ,key:value};

**Properties:** key value pair

**Linear:** array is a linear data structure (ordered)

**Non linear:** object is non linear data type(unordered)

***Arrays***

* Array is the **heterogeneous** and **ordered.**
* No need to define its type of which **data type** it is.

**Define array in js :** for ex:

Let arr=[];

**1- D Array**

Let arr=[10,20,30];{homogenous}

Let arr=[10,”aryan”, true,10.2] {hetrogeneous}

**2-D array:**

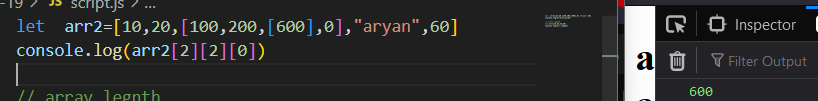
Let arr2=[10,20,[100,200,[600],0],”aryan”,60]

**Acess the value by index:**

arr2[1]=20

arr2[2]= [100,200,[600],0]

arr2[2][1]=200



Length is a property **:**  used to find the length of the array

For ex: let a=[10,20,30]

a.length=3

**Method of Arrays:** function which are define inside object. Some method are given below-

**1. push:** insert the multiple element in the the array from last (*if we store the push method in variable then it will return the length of the array after insertion* ).

For ex:

Let a=[10,20,30];

a.push(“aryan”)

output: a=[10,20,30,”aryan”]

**let b= a.push(“aryan”)**--length of array after insertion **output is 4**.

**2. pop:** delete the last element from the array only one element delete at one time .

For ex:

Let a=[10,20,30];

a.pop(30)

output: a=[10,20]

**let b= a.pop(30):** jo value delete hui hai wo aayegi (this is return type of this method)

**method for work from starting as similar as** *push* **and** *pop*

**unshift() work as push()**

**shift() work as pop()**

**note:** bilkul same kam karte hai ye dono same to same like pop and push bss ye aage se suru karte hai

Javascript is **single threaded**, **weakly typed**, **dynamically typed**, **interpreted** programming language.

**Note:** shortest code of js is **empty js file.**

Whenever a js code is run a **gobal execution context (GEC)** is created inside that gec we have two phases :-

1. **MEP(memory execution phase):** code chalne se pehle variable(isme value nhi hogi undefined hogi only variable stored hoga ) and function ko memory milegii .
2. **CEP(code execution phase) :** jab mcp chal jayega tab ye chalega aur pehli line se chalega and jo value a ko mili hogi wo mcp me update kar dega. Aur function aayege tab jaha function intialization kiya hai wo pehle chalega aur ek gec context banega same with function name after function execution ke baad wo gec jo bana tha function ka wo khtm ho jayega aur phir pura code chal ke gec khtm ho jayega .

**Note:** js is run by call stack,inside call stack gec is formed after execution of last line of code the gec removed from call stack

Whenever a function is called the new execution context is context is created with the name of the function which also as two phases (same as above).



**3-decalartive :**

|  |  |  |  |
| --- | --- | --- | --- |
|  | let | const | var |
| Reinitialization |  |  |  |
| Re declaration |  |  |  |

For ex:

Const a; output:(error)

a=10;

Console.log(a);

**reason**: const me undefined ho phir usko change nhi kar skte a=10 kiya to error aa gya kyuki undefined usme constant ho gya.

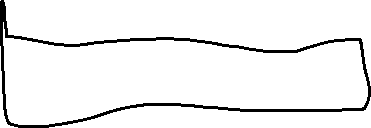
**Hoisting**: accessing a variable or a function before its declaration is called hoisting.

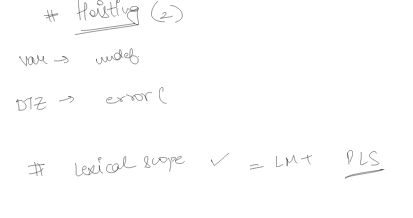
**Var,let, const all the three declarative hoisted but hoisting has two different type** **. var having different hoisting then let and const.**



**Note:** in var hoisting give **undefined** as a output.

And in case of **let and const** give **error(uncaught reference error: cannot access a before its initialization)** because they are **DTZ(dead temporal zone) hoisting.**



****

**Dead temporal zone:** it is the imaginary zone that come between MCP and CEP.it activated during let and const do not allow access unless you provide the value of that variable.

**Scoping:**

**Lexical scoping**: local memory(MCP) +parent lexical scope(which means parent local memory +its parent lexical scope and so no)(ye hota hai ki khud agar usme nhi hai to uske jis stack ke andar bana hai uske ek ek level karke upar badega aur check karega jab tak gec call stack me bana hai uss tak usme nhi mila tab error dega aur agar mil gya to acess karega for ex. Virasht ki rihast)

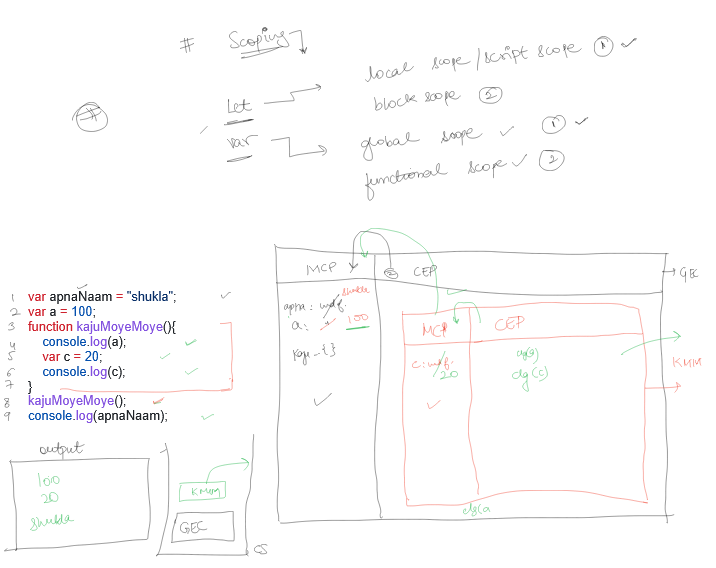
**Typing of scoping**:

Whenever a js code is run a global execution context (gec) is created ,along with it a global object is also created

in our case that is since we are using the browser the global object is window.(samjhne ke liye hai bss).

in case of scoping , let and var are considered to be having two different scopes

* **Var**- global scope or functional scope
* **In the case of let** , local scope(script scope) or block scope agar mep
* **Function :** it is also come in global scope.

When nothing is return in a function by default it return **undefined**. 

**Q1. why is the function called first class citizenship/which functions are first class functions?**

Ans- In JS , whenever you are able to assign a variable to a function then that function will be called as first class function and its concept of assigning a variable to a value function is called first class citizenship.

Ex- let b=function aru(){

Console.log(‘hello’);

}

aru();

**Q2. What are higher order function ?**

Ans. There are two possible situation for higher order function-

1. When you have two function(f1 ,f2) and any one of these function (assuming f2) is being sent as an argument to the other function (f1) then f1 is called higher order function.
2. When you have two function (f1 ,f2) if one of the function (f1) can return the other function(f2) from itself then f1 is called higher order function.

For ex: **statement i :-**

Function f1(fn) {

Console.log(‘f1’);

fn();

}

Function f2() {

Console.log(‘f2’);

}

f1(f2);

for ex: **statement ii :-**

function f1() {

console.log(‘hi’)

function f2() {

console.log(‘hello’)

}

return f2();

}

let returned =f1()

returned()

**Q3. What is the call back function?**

Ans. Call back function is basically when you pass it as an argument , to some higher order function and that passed function is being called insider that higher order function , this **calling criteria must** then only it will known **as call back function.**

**Array method :-**

1. Array.foreach(): ye return nhi karta bss iterate karta.

Arr.foreach(function(item,index){console.log()})

1. Arr.filter(): isme return karta hai but same size ki array nhi return karta bss jo **true** hai bss unne hi return karta hai( 0-array ki length)

let newArr=Arr.filter(function(item,index){

if(item>50)

{

Return true;

}

})

1. Array.map(): ye return karta hai ek array same size ki

Let newArr= Arr.map(function(item,index){return item\*2});

1. Array.sort():
2. Array.reduce():
3. Array.find():

**Closure:** whenever you return a function that function never returned along its always takes **along the dependent lexical environment** with it so that if in future we ever want to run that **returned function** it should not throw an **error**.

**For ex:** Function outer(){

Let a=100;

Function inner()

{

Console.log(a);

}

Return inner;

}

Let out=outer(); iska output : 100

out();

**Real time Usecase of closure**:

Before es6 the class syntax was introduced we had no ways to privatize are variable or function so in that case we use closure so that we can privatize the variable.

For ex:

Function counter(){

Let count=0;

Return {

getCount:function(){ console.log(count)},

increment: function(){count++}

decrement: function(){count--}

reset: function(){count=0}

}

Let count1=counter();

Console.log(count1);

Count1 console par print karenge to jitne function return me hai uthne aa jayege;

Count1.getcount(); ye dega count ka print

Count1.increment(); ye kar dena increase

Count1().decrement(); ye kar dega decrease

Count1.reset(); ye reset to 0 kar dena

Console.log(count) to error aayega kyuki count privatize hai us function ke liye only

**Prototype:**

(har object ka fallback source hota hai like object.prototype (ye bhi object hai ) )

jab mere object se me wo access kara rha hu jo uske pass nhi hai to wo aapne **parent par fall back** hota hai wo parent object.prototype

aur different object ke different prototype hote hai like array to iska prototype **Array.prototype and iska parent hoga object.prototype** and iska **null** . same with string

For ex:

Let obj={

a: 10,

b: “aryan”,

c:function(){console.log(“Aryan sharma”)}

}

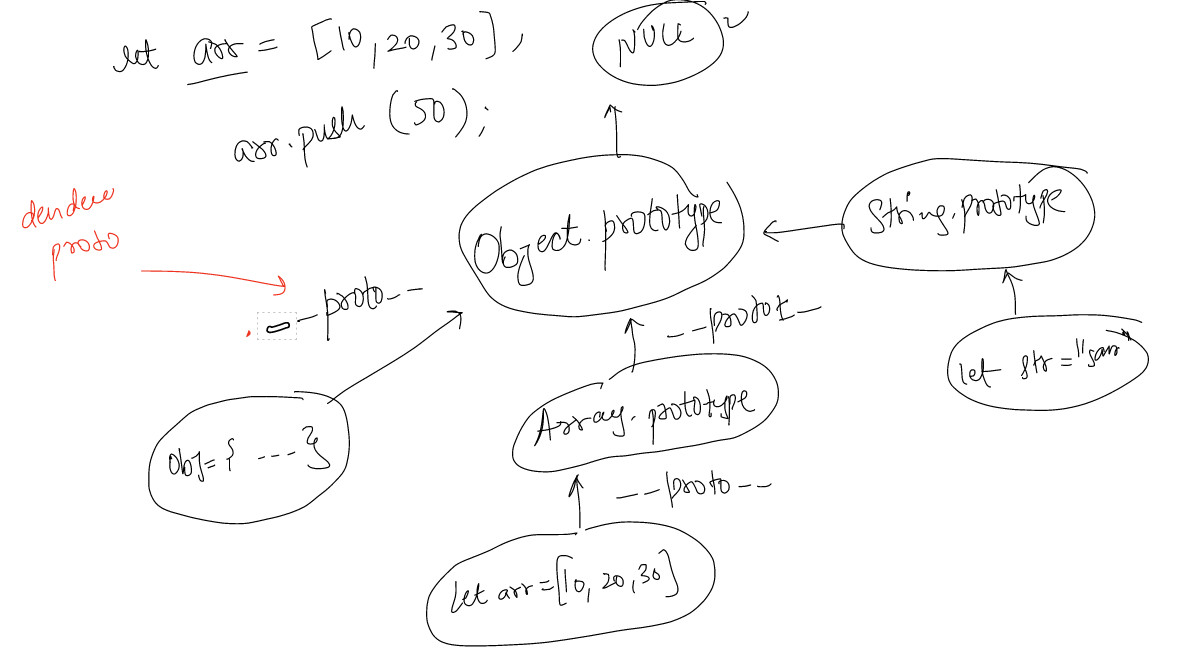
console.log(b) iska answer aayega **aryan**

console.log(c) iska **undefined** aayega kyuki iska return kuch nhi hai

console.log(d) iska answer **undefined**

console.log(e()) iska answer **uncaught type error** aayega

console.log(obj.toString()) iska answer **object,object**

****

**note:** kiska ka parent jana ho to use karte hai **dender proto(.\_\_proto\_\_)** last parent to koi hoga to hamesha kiska last parent **NULL** hoga.

Hand written object ka **dender proto** **object.prototype** hota hai.

**Object prototype ka parent null hota hai.**

**Constructor function**

by using **new keyword** kyuki mene banaya **aru() function** then **new keyword se function khud bana dega** aur **object return karega.**

For ex:

Function Aru(){}

**Function aru(){}**

Let out1=aru(); iska output **undefined**

Let out=new Aru(); iska answer hoga **Aru{}**

Console.log(out)

For ex1:

Function Aru(){

This.naam=”kaju”; **isme aese nhi likhte name:”kaju”**

**Isko likhte hai this keyword use karte hai**

This.umar=”2”;

}

Let out2=new Aru();

Console.log(out2) iska answer hoga Aru{**iska ki jo humne banaye properties likhi hai**}

Note: **constructer function ka parent jo handwritten function hai wo hota hai aur constructer function me hum kuch return nhi karte .**

**Arrow function:**

Syntax of arrow function teen tareek se likh skte hai:

---------***First Ways***-------

For example- let sum=(a+b)=>{return a+b};

Console.log(sum); iska answer aayega function pura;

Let ans=sum(10,20);

Console.log(ans); iska output aayega 30;

------------***second way***-------------

**Agar function me ek hi executeable statement hai to sab hata skte ho**

For example-let sum=(a+b)=> a+b;

let ans=sum(10,20);

console.log(ans);

-------***third way***-------------

**Agar function me humne ek hi argument pass kiya hai to hum aese use kar skte hai**

For example- let sqr=a=>a\*a;

Let ans=sqr(5);

Console.log(ans);

------------**“THIS” keyword**----------

This keyword depends on how it is being called upon.

1. Object/method calling- object par depend karega

Let obj={a:10, b:200, fn:function()

{console.log(this);}}

Obj.fn(); **isse output console par pura object**

1. Direct function calling

Function aru(){console.log(this)}

Aru(); **iska answer window aayega**

Example : how this keyword depend ki kaise call kiya gya hai

Let obj={a:10, b:200, fn:function()

{console.log(this);}}

Let ans=obj.fun;

iska mtlb ki poora function assign hua hai ans ko

Ans(); **iska answer bhi window aayega**

1. Constructor: iska kaam hota hai new object created karta hai aur this keyword point new object

Function Aru(){this.name=”aru”}

Let obj1=new Aru();

1. Indirect calling
2. Arrow function

Ye depend karta hai iska parent ka this kisko point kar rha hai jo parent ka this karega wahi arrow function point karega .

Let obj={a:10,fn()=>{console.log(this)}}

Obj.fn(); **iska answer window**

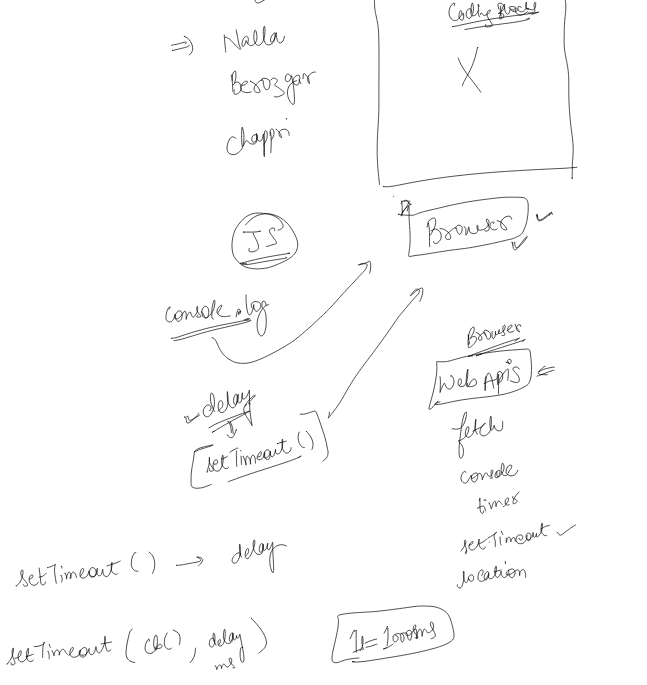
Note:

kebal console.log(“this”); ko chalne par window kyu aata

ans: isliye window aata hai kyuki window.console.log() aese use karte hai isko ab lagao ya nhi to bhi ye automatically call ho jata hai .

**Phase -3**

Javascript is a **synchronous** programming language but it can show **asynchronous.**

****

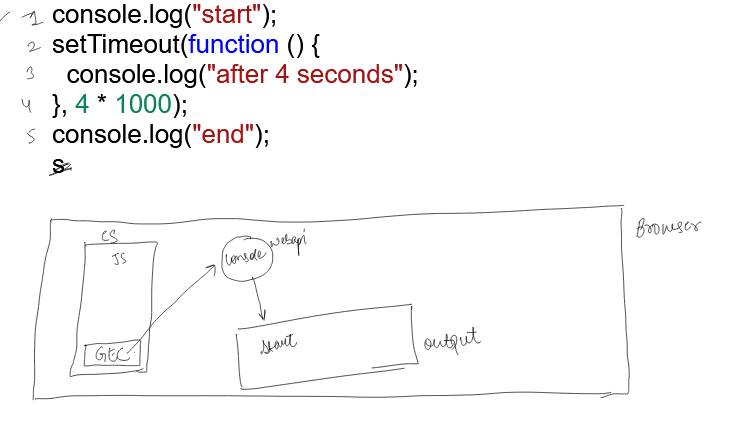
**Note:** js ko browser provide karta hai sab through web APIs.

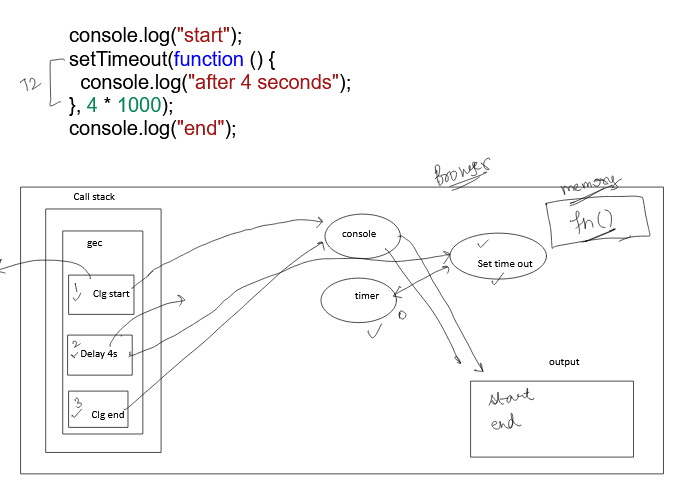
Ex: console,timer,fetch,location,settimeout

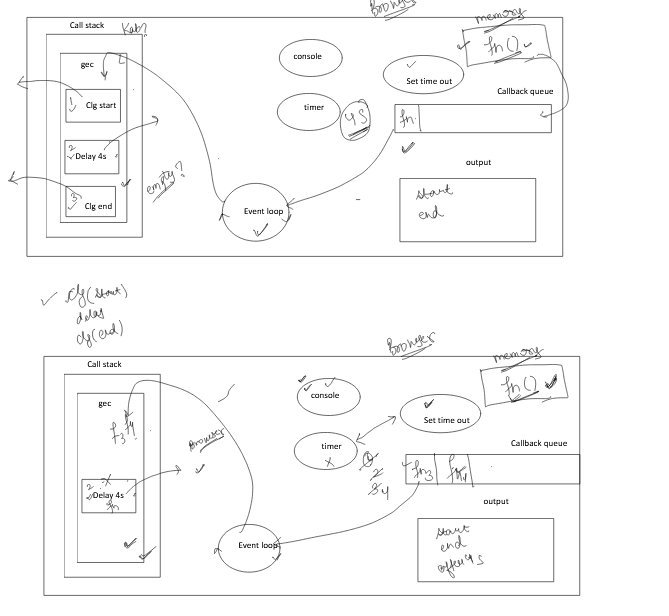
Set timeout() ye ek method hai jiski help se delay create karte hai.

For ex: set timeout(callbackfunction,delay in msec)

Q1.







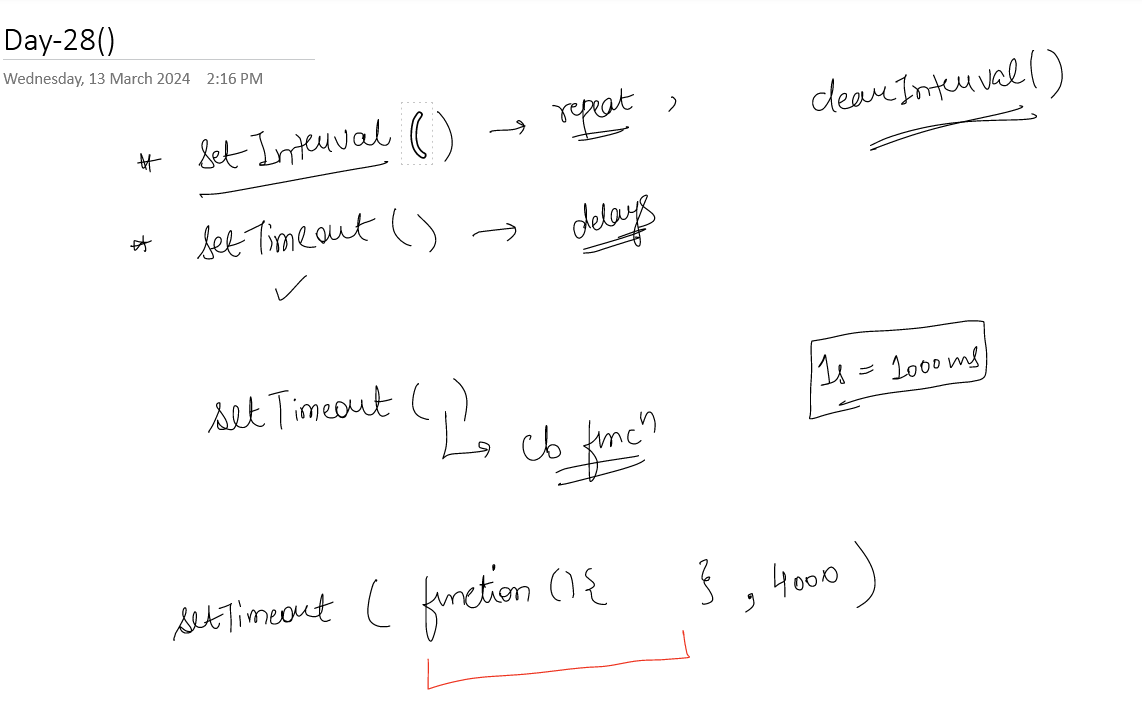
**Web APIs:**

* Set interval(function(){},time) – repeat infinite task
* Set timeout()-- delay
* clearInterval(id)—set interval ko rukhne ke liye

ex: let id =set interval(function(){console.log(mera man jo karega mai wahi karungi )},3000)

set timeout(function(){clear interval(id)},10000)

**Notes:** ye clear interval set timeout ke andar isliye kept kiya hai kyuki ye web api ke through use hota hai to id pehle mil jati hai to set interval chalne se pehle clear interval chal jata hai to clear kar deta hai if set interval ke andar use nak are to.



**Promises:** promises is introduced by avoiding call back hell but it is not a best( in this chaining is happened) way to resolve it so we can learn better later…..

