J.S Advance Crash Course

J.S version → ES5 (old) → var
→ ES6 (new) → Let, const

we use both ES5+ES6 so we can use var, Let, const

1) Let v/s var

var → function scoped ⇒ var apne function scoped main kahi bhee use ho sakta hai (nearest parent function scoped)

let → braces scoped

```
function abc(){
for(var i=0; i<=12; i++){
 console.log(i);
 }
console.log(i);
}
```

```
O/P
0
1
2
3
4
...
12
12
```

due to inside function & function scopped

```
function abc(){
for(let i=0; i<=12; i++){
 console.log(i);
 }
console.log(i);
}
```

```
O/P
0
1
2
...
12
```

i is not-defined

due to let keyword is used & i is braces scopped

★ 2) **window object**

→ ~~Let~~ var add it self to the window object

→ Let, const don't add it self to the window object

★ JS Language ke paas bohat saari features hai but kuch kuch features wo khud se use nehi kar pati hai so bo window object ka use karti hai & window hai ek box feature given by browse

→ only var(ES5) can ~~use this~~ add it self to that window box but Let & const can't

Var B=12;   Let C=13

go to Ctrl+shift+j → write window u can see there value of B is already there due to B add itself to window box

* alert, window, console, prompt → These are not J.S part these are windows box features given by browser

* but array, let, var, while, for these are part of J.S

* Remember ex of pen, paper, mobile holder
→ pen hole on paper, pen write name
→ pen give me mobile phonestand, he can go to box & bring from that

Note
~~In of~~ here comes the Q if we have var as a variable in ES5 then what is the need of Let & Const in ES6?

Ans) kyunki var window mai apne aap ko add karleta that so any can see our data using window object provided by browser (security bridge here) to overcome this problem in the latter version introduce to Let & Const which is more secure but braces scoped not function scoped

Const,Let V/S Var

Var	Let, Const
→ older version ES5	→ newer version ES6
→ function scoped	→ braces scoped
→ add itself to the window object	→ don't add itself into window object

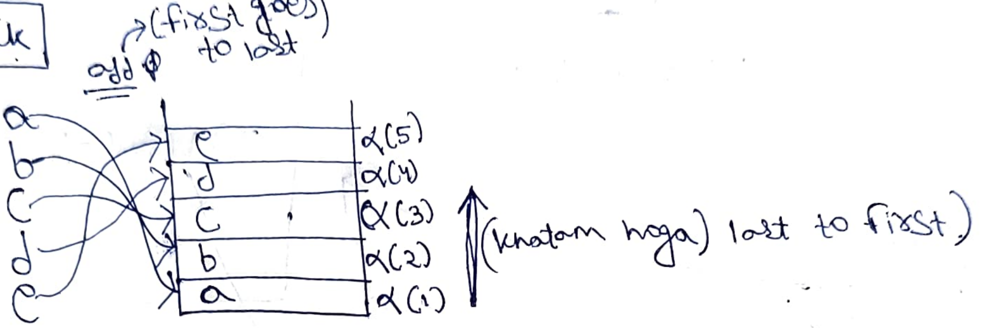
* Browser Content API mainly

browser provide 3 features which is known as browser Content api → Alert, prompt, console . . . .

1) window
2) Stack
3) Heap memory

→ Browser Content API

* Stack



Jis order mai log add hote hai usi order mai bahar Jate hai

* Heap memory

$$1+2+3+4+5$$
$\downarrow$
3

After ading 1+2=3 before going to 3+3+ . . . It is Store first 3 in the heap memory, the 6 in heap then 6+4=10 in heap 10+5=15

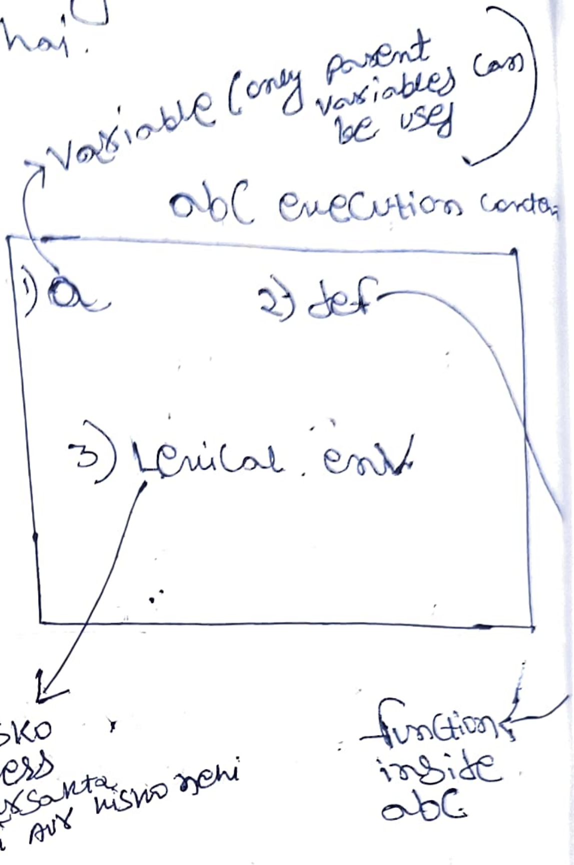
Heap memory → Jitne bhee data (or) variables hote hai unko Store karne ke liye Heap memory use hota hai

2) Execution Content

→ Jab bhi hum function banayenge tab function apne aap ek imaginary container bana lega Jisme ye tino cheez rahenge
1) Variables 2) function inside that Parent function
3) Lexical env. of that function

is container to imaginary usko hum exe-cution context bolte hai.

```
function abc(){
    var a=12;
    function def(){
        var b=12;
    }
}
->call
abc();
```



Variable (only parent variables can be used)
abc execution context
1) a
2) def
3) Lexical env.
Kisko Access Kar sakta hai aur kisko nahi
functions inside abc

☆ here we can't use var "b" due to it's nearest Parent function is def(), & var is nearest Parent scoped

Execution Context -> execution context is a container which is created whenever a funct-ion called, it contains 3 things
1) Variables of the Parent function
2) function inside " "
3) Lexical env. of that function

3) Lexical env.

-> Lexical env. is a kind of a chart jisme ye likha hota hai ki aap perticular function ke konse konse cheejo ko access kar sakte hai & kinko nahi matlab ki it holds scope & scope chain.

scope chain means only parent function it en
can access, sabse andar wale function apne
jitne bhee parent hai sabko access kar
sakta hai but child ko nahi,

4) How to Copy reference values

→ spread operator

```
var a=[1,2,3]
 var b=a;
console.log(b) // [1,2,3]
Console.log(a) // [1,2,3]
 b.pop(); // [1,2]
Console.log(a) // [1,2]
```

```
var a=[1,2,3];
var b= [...a]; //copy of a
b.pop(); // [1,2]
console.log (a); // [1,2,3]

Var Skm ={
     name: "Subham"
   };
 var c={... Skm};
Skm // {name: "Subham"}
 c // {name: "Subham"};
c.name = "Ramny";
    ↳ i can change
```

5) Truthy & Falsy

→ J.S main Tum kuch bhee likh do wo hamesa
mainly 2 types ka hota hai → Truthy / → falsy.

Falsy → '', 0, false, NaN, undefined, null, document.all

Truthy → ~(falsy), anything which is not in falsy
Ex → "subham", -7, -2, 1, true, ...

```
                              name
if(-1){              → "skm", -7,
     ← Truthy, so this will execute
}
```

```
if (null){          → falsy
}
else {
      → else will execute due to falsy
}
```

6) $\boxed{\text{forEach, forin, do-while}}$

forEach → only use if we have array

```
Var a = [1,2,3,4]     → everytime comes
                        uper function
a.forEach(function(val){
    ~~console.log(a);~~ console.log(val+2);
})
```

o/p
3
4
5
6

☆ ~~Var~~ forEach kabhi direct array mai change nehi karta wo ek copy banata hai uske upar changes karta hai So main array kuch change nehi hota

forin → only work/use upon objects

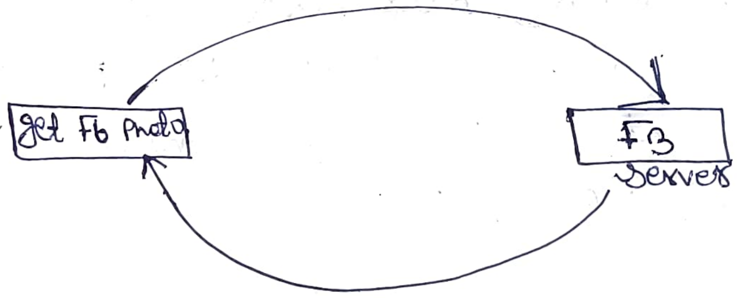
Ex→

```
Var skm={
    name:"subham",
    age:27,
    school:"RCST"
};
```

```
for (var key in obj){
    Console.log(key);              → name
                                     age
                                     School
    Console.log(obj[key]);         → Subham
                                     27
                                     RCST
    Console.log(key, obj[key]);    → name    Subham
                                     age     27
                                     School  RCST
}
```

$$\begin{tikzcd} \text{Obj[key]} \arrow[r] \arrow[r, bend left] & \text{obj[name]} \arrow[r] \arrow[r, bend left] & \text{Subham} \arrow[r] & \text{we can Access obj's name} \end{tikzcd}$$

7) Callback functions



→ Jab bhee koi esa code jo baad main chalta hai aap likhoge, kyunki ye code baad main chalta hai J.S ko ye pata nehi chalta ki wo complete hua ya nehi, aise code ke completion par js ko btaya jaata hai ke wo complete hogaya aur aap use chala sakte ho, ye batane ka kaam callback ka hai

(P.T.O)

→CamelCase

```
setTimeout (function() {
    Console.log("now run");
}, 2000);
```

→ Async. J.S

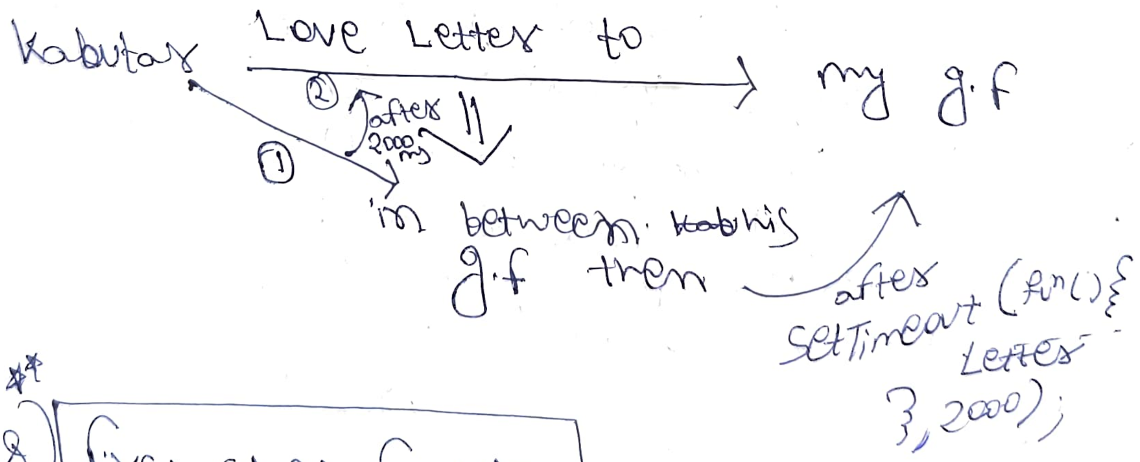
2000 → 2 sec

* Console.log will execute After 2 sec

Ex→ kabutar $\xrightarrow{\text{love letter}}$ to my g.f
⇓
in betwn kabutar gf

→ esa code jo baad me chalta hai usko hum ek function de dete hai ~~ke~~ ki bhaiya jab complete hojana to ye function chala dena, aur wo function jo hum dete hai wo ek normal function hi hota hai / aur usea hum callback function kehte hai

Ex :-


Kabutar
Love Letter to
my g.f
②
after 2000 ms
①
in between ~~kab~~ his g.f then
after setTimeout (fn(){ Letter }, 2000);

**
8) $\boxed{\text{first class function}}$

→ J.S main ek concept hota hai jisme hum function ko as a value le sakte hai

(P.T.O).

```
var a= function (){};

function skm(a) {
    a();
}

skm( function ( ){console.log("name");})
```

o/p = name

**

9) How arrays made behind the scene

```
var arr= [1,2,3,4];

arr={
  0:1,
  1:2,
  2:3,
  3:4
};
```

J.S convert to object (in back)

* TypeOf [ ] → o/p = object
* TypeOf { } → o/p = object

How can we know which one is array & which one is obj

Array.isArray([ ]) → o/p True

Array.isArray({ }) → o/p false

10) How to delete obj. props

```
Var obj = {
        name = "Skm",
        age = 23,
        School = "RCST",
        };

delete obj.name ; //here name key
                    delete from object
```