Js - Day 2

⇝ datatypes

var : It's deprecated

↳ too many issues-- , scope related--

- let ... introduced..
- is let hoisted? : Yes ✗

Yes + Explaination ✓

↳ age is defined but can't accesed earlier..

```
age=12;
let age = 11;
```

↑ Temporal dead zone

declared variables (let, const) exist in memory, but not accesible

- Const -- introduced
  - ↓
- Zyadatar iska use..
- const cant be changed
- same behavior as Let in context of Hoisting..

FUNCTIONS

- Set of reusable instructions.
- performs something
- may return something
- return → last statement.

parameters ↑

Syntax

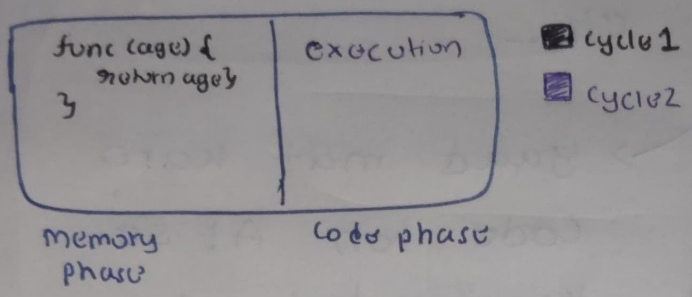
- function myfunc( )
  {
  //code
  }

function call → myfunc( );

→ function can return any thing-- function, exp, variable
(any thing ---> javascript legal)

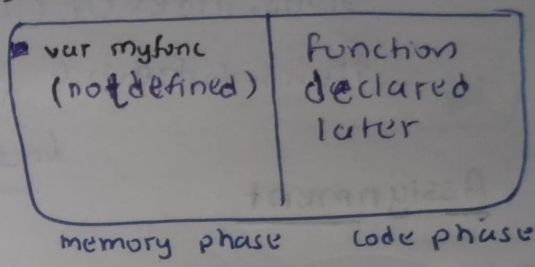
→ a variable can hold a function.

```
function func (age) {
    return age >= 18;
}
```



- function can be accessed before - (defined in memory phase)

```
var myfunc = function (age) {
    return age >= 18;
}
```



- can access function before (undefined)

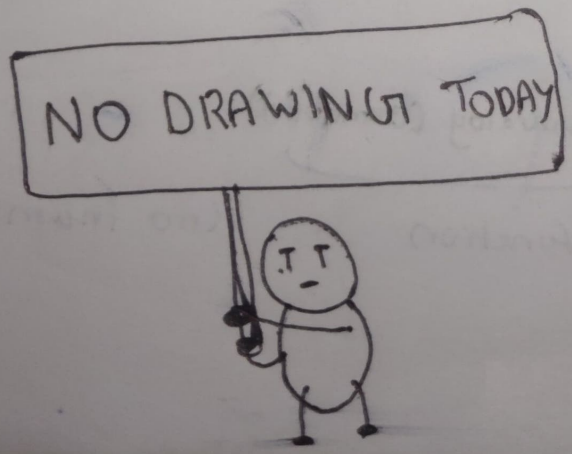
Arrow function

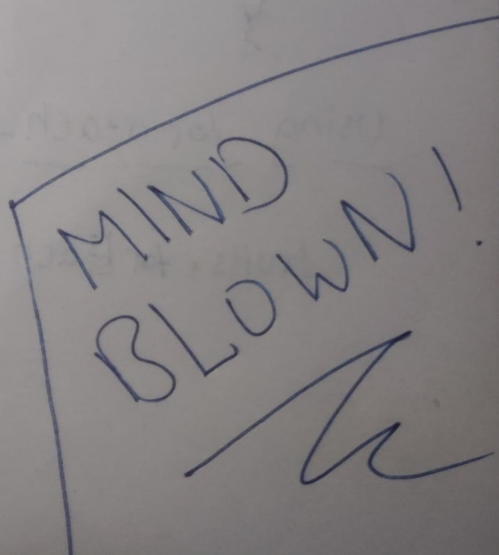
frame parameter return

- `const isAllowedToVote = (age) => age >= 18`

e.g.;

```
const isAllowedToOpenAccount = (age, minbulance) =>
    age >= 18 && minbalance >= 5000
```





#DATA STRUCTURES

- Array ⇒ let fruits = ["apple", "mango", banana"]

↳ methods--

- fruits.include("mango") ⟶ true

e.g, fruits.shif()
fruits.pop()
fruits.indexOf()
fruits.forEach(↑ function)

etc--

↳ yaad mat karo code toh AI se karwana hai-- Lmao

Assignment

→ Implement Queue & Stack -- using Array--

HIGH ORDER FUNCTION

```
function jollyFunction (udhaarKaFunction) {
    return udhaarKaFunction () + 40;
}
```

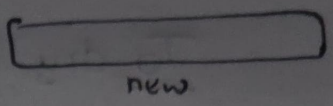
using for each--

fruits.forEach (element ⇒ console.log (element))

arrow function

(no fname)

map → internally new array create karta hai




```
const nums = [1,2,3,4,5,6];
const result = nums.map ((xyz) => xyz * 2);
```

op => 2, 4, 6, 8, 10, 12

---

me → github.com/realSUDO

SKIP

X → @sudo_core

discord → @ sudo.dis

LOL PROMOTION

---

for each

→ only iterates

map

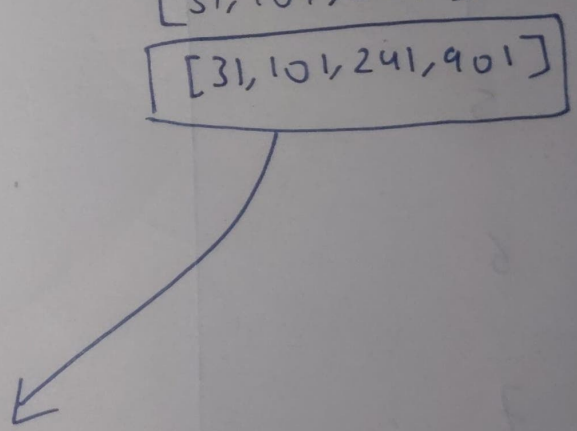
→ creates new arr

first

To dry run

```
const nums = [3,10,24,90]
const result = map(e => e*10+1)
function map(fn) {
    const result = [];
    for(let i=0 ; i < nums.length ; i++) {
        const currentElement = nums[i];
        const num = fn(currentElement);
        result.push(num);
    }
return result
}
console.log(result)
```

i	curr Element	num	result
I1 :	3	31	[31]
I2 :	10	101	[31, 101]
I3 :	24	241	[31, 101, 241]
I4 :	90	901	[31, 101, 241, 901]



Final result

31, 101, 241, 901

second

```
const nums2 = [3,10,24,90,80,34, 67]
const result2 = nums2.forEach ( function (e) {
                if (e%.2 ===0) {
                console.log (e)
                }
```

3)

```
Console.log (result2)
```

Iteration	e (parameter)	console.log (e)
1	3	—
2	10	10
3	24	24
4	90	90
5	80	80
6	34	34
7	67	—

Output =

10
24
90
80
34