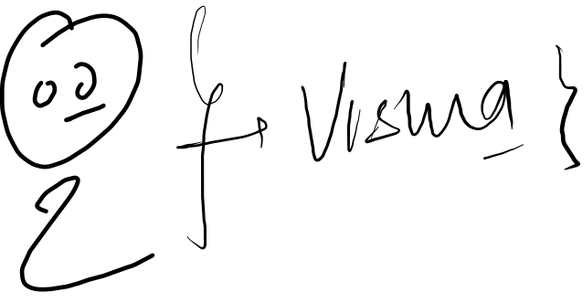
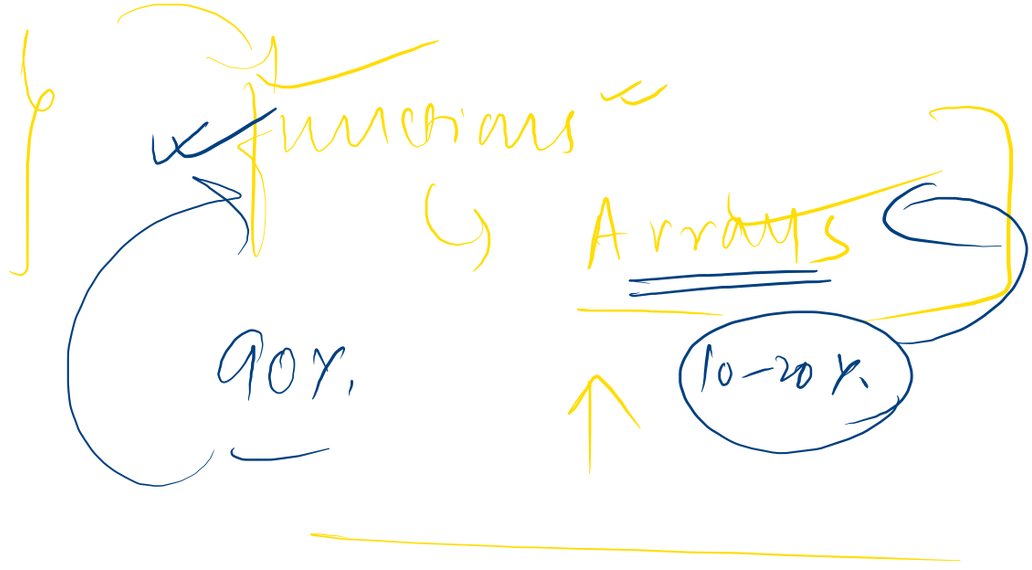
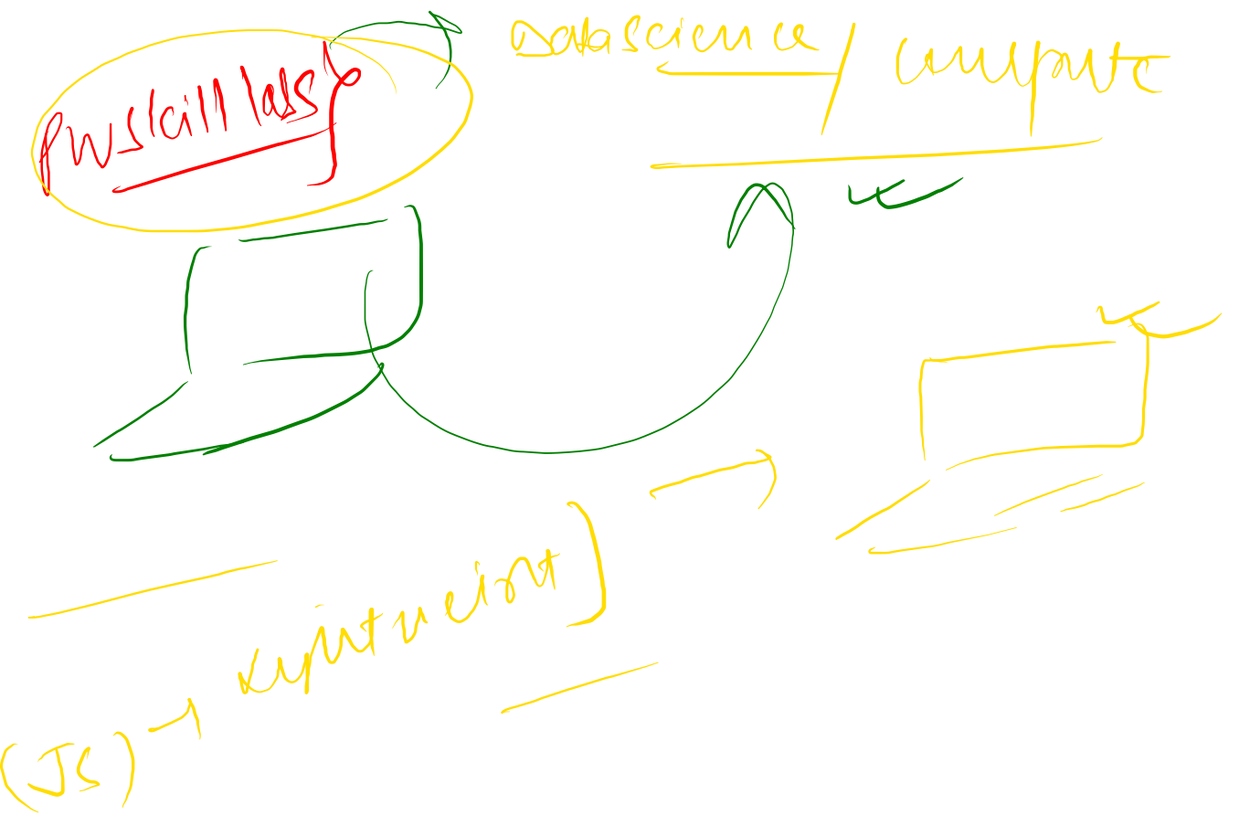


conditionals / Loops

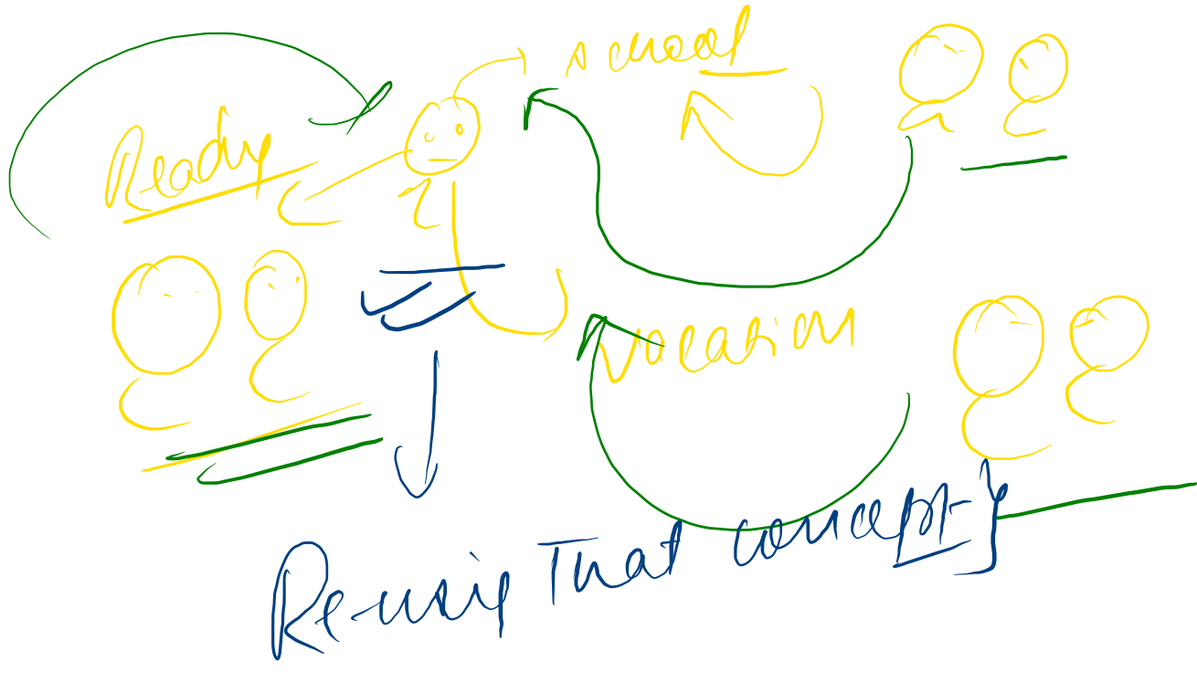


common

(important)


Data science / compute
(JS) → instruction

functions



Programming

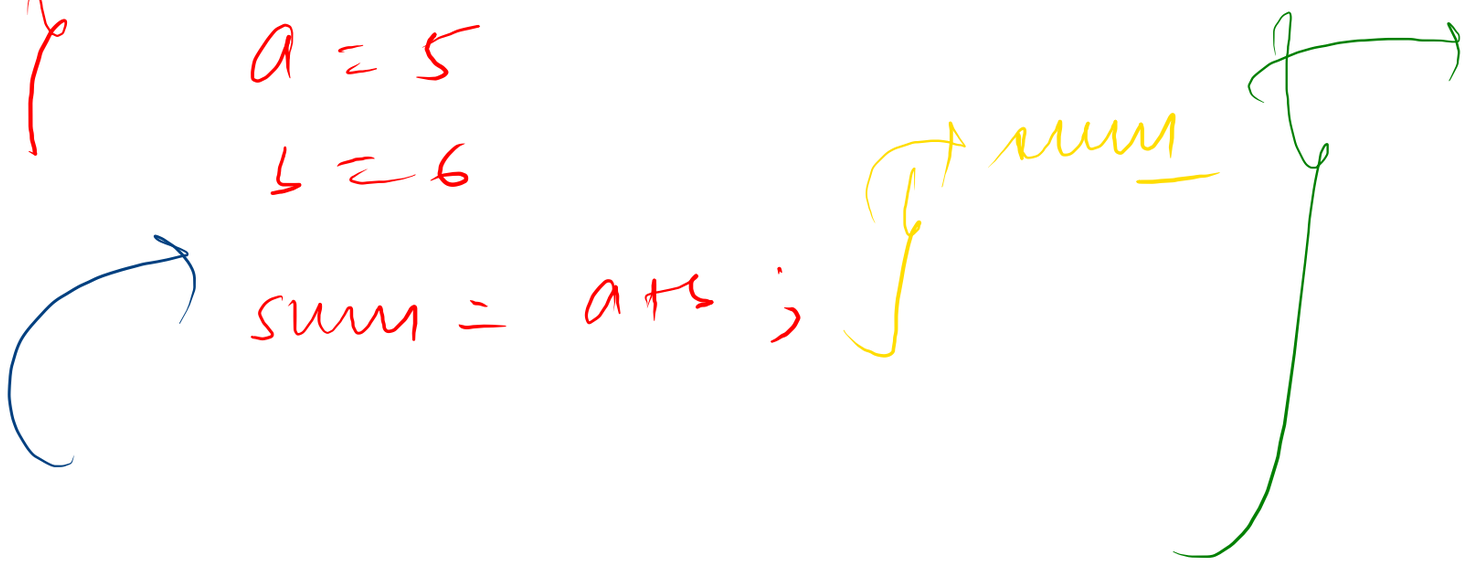
concept ⟹ multiple times

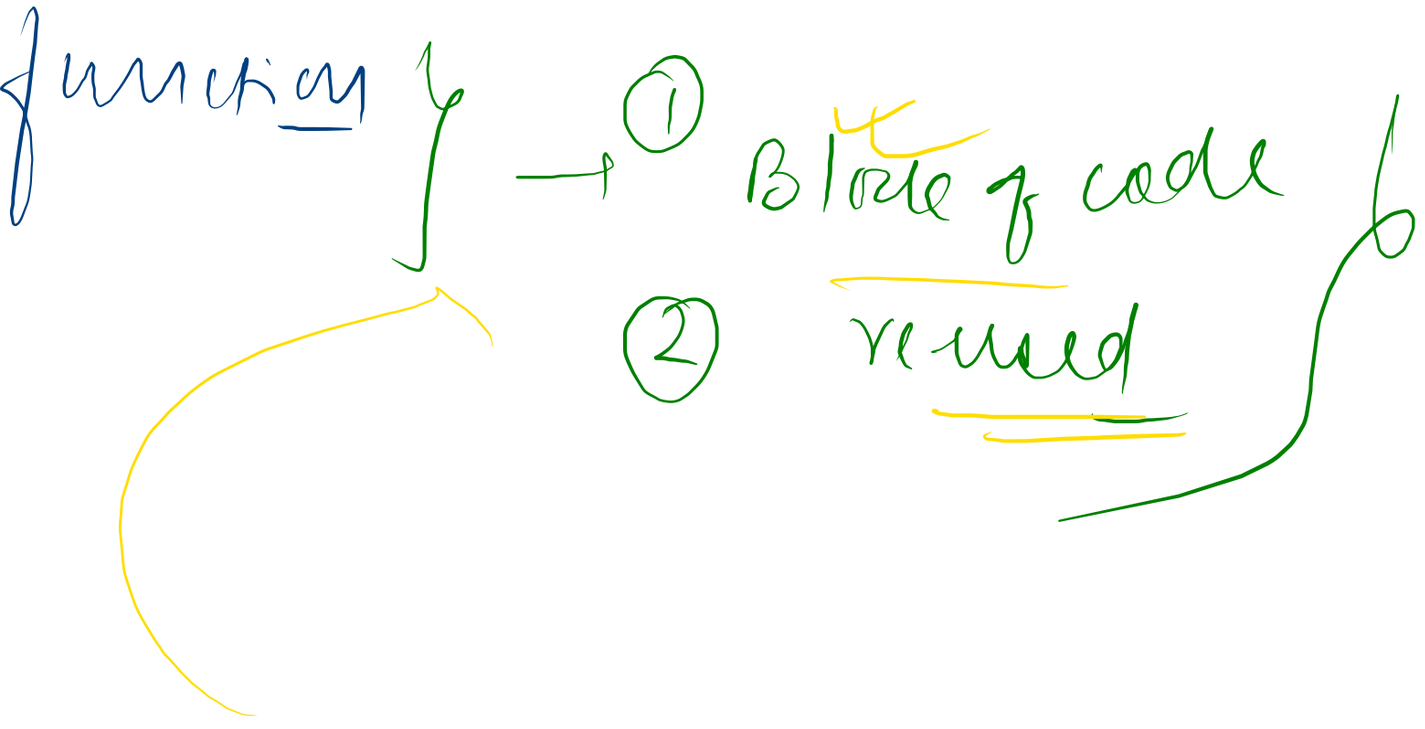
sum

a = 5

b = 6

sum = a+b ;



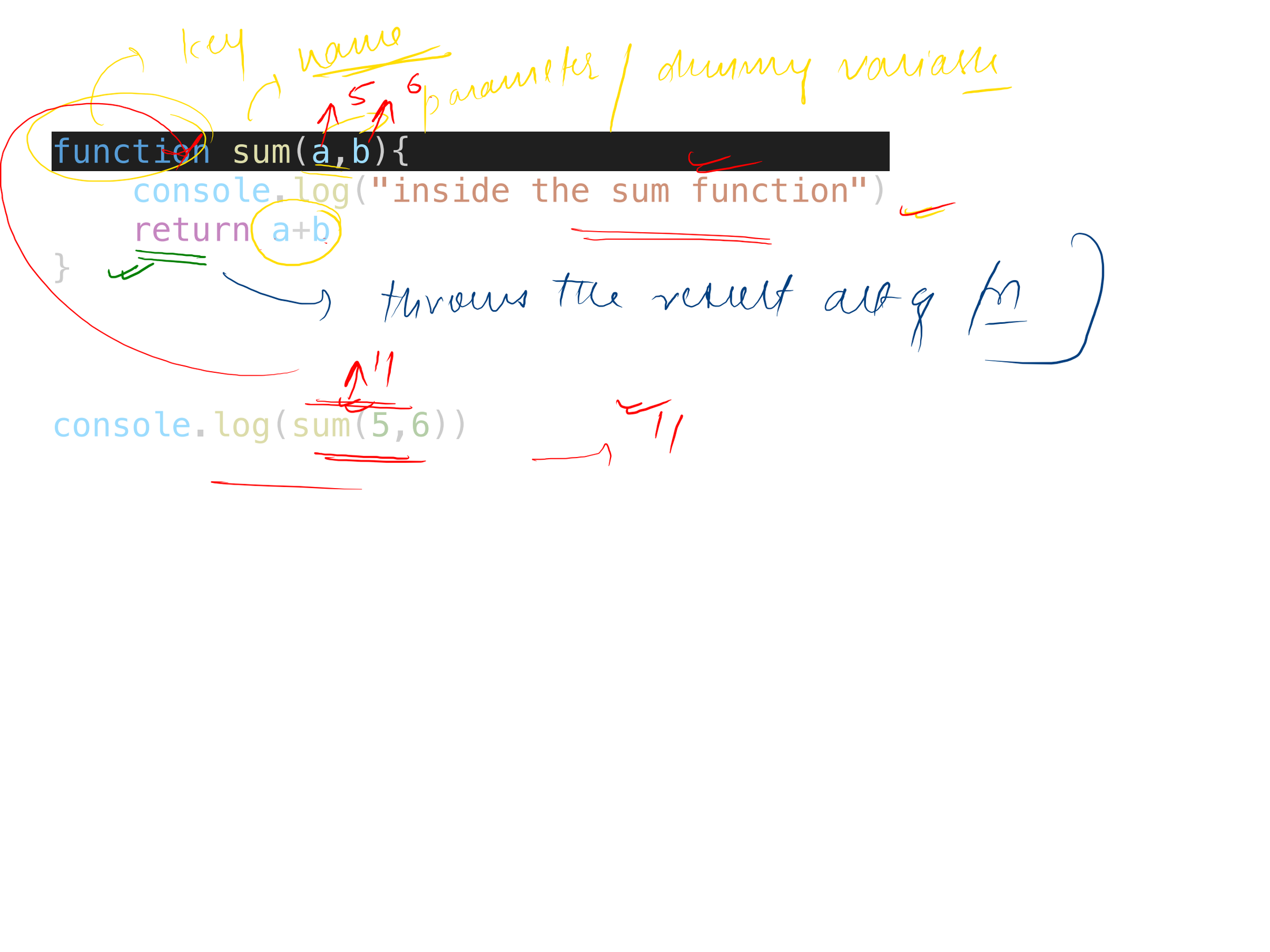

function
1
Block of code
2
reused


1
keyword
name
parameters
function greet_students(){
console.log("Hi")
console.log("Welcome to the 7 PM class")
console.log("Keep learning")
}
start of the body
function body
logic
Return value
function ending
greet_students()
use
arguments
executing function


Real world
Amazon
Cart services
[(axios) + inventory
(axios) + logging
function checkout() {
function checkout-inventory() {
}
}

= 10 Minute

DMA/ Questions


key
name
parameters / dummy variables
5
6
function sum(a,b){
console.log("inside the sum function")
return a+b
}
returns the result of fn
11
console.log(sum(5,6))
11

Parameters / Arguments

← Definition ← parameters

```
function product(a, b) {
    return a * b;
}

product(5, 6);
```

execute ← arguments

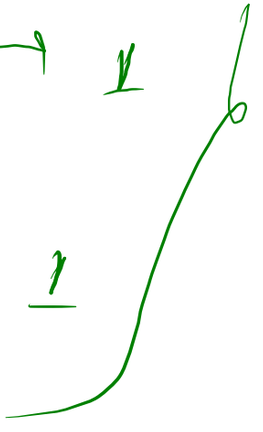
Q) write a fn for factorial of a number

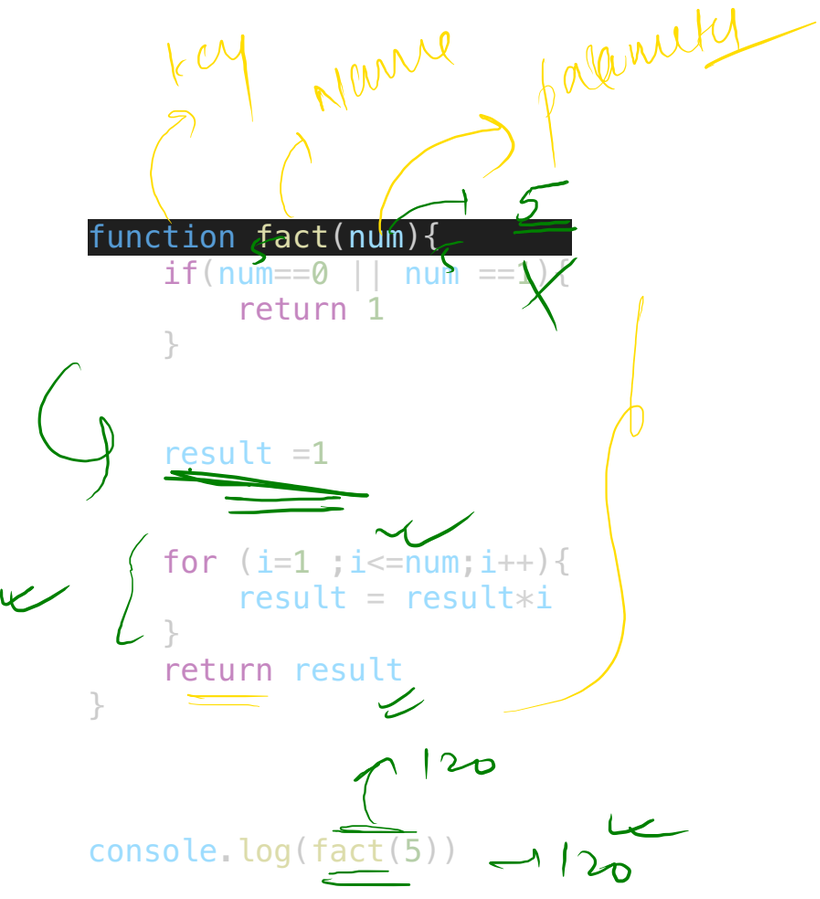
5! = 5 x 4 x 3 x 2 x 1 = 120

3! = 3 x 2

0! = 1

1! = 1





1 → i = 1

result = result × i

result = 1 × 1

2 → i = 2

result = result × 2

= 1 × 2

3 → i = 3

result = result × 3

= 2 × 3 = 6

(4) i = 4

result = result × 4

= 6 × 4

= 24

(5) i = 5

result = result × 5

= 24 × 5

= 120

fact(1) → 1

```
function fact(num){
    if(num==0 || num ==1){
        return 1
    }


    result =1


    for (i=1 ;i<=num;i++){
        result = result*i
    }
    return result
}



console.log(fact(5))
```

fact(3) ⟶ 6

result = 1

```
function fact(num){
    if(num==0 || num ==1){
        return 1
    }

    result =1

    for (i=1 ;i<=num;i++){
        result = result*i
    }
    return result
}

console.log(fact(5))
```

1) i = 1
   result = result x 1
   = 1 x 1 = 1

2) i = 2
   result = result x 2
   = 2 x 1 = 2

3) i = 3
   result = result x 3
   = 2 x 3 = 6

(JS) → functions → 1st class citizen

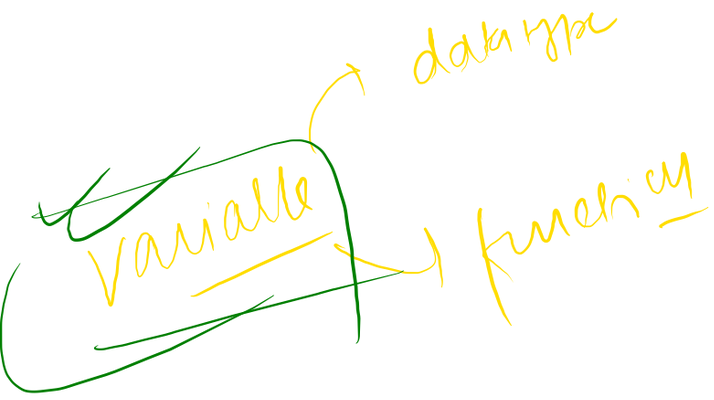
Example
How?

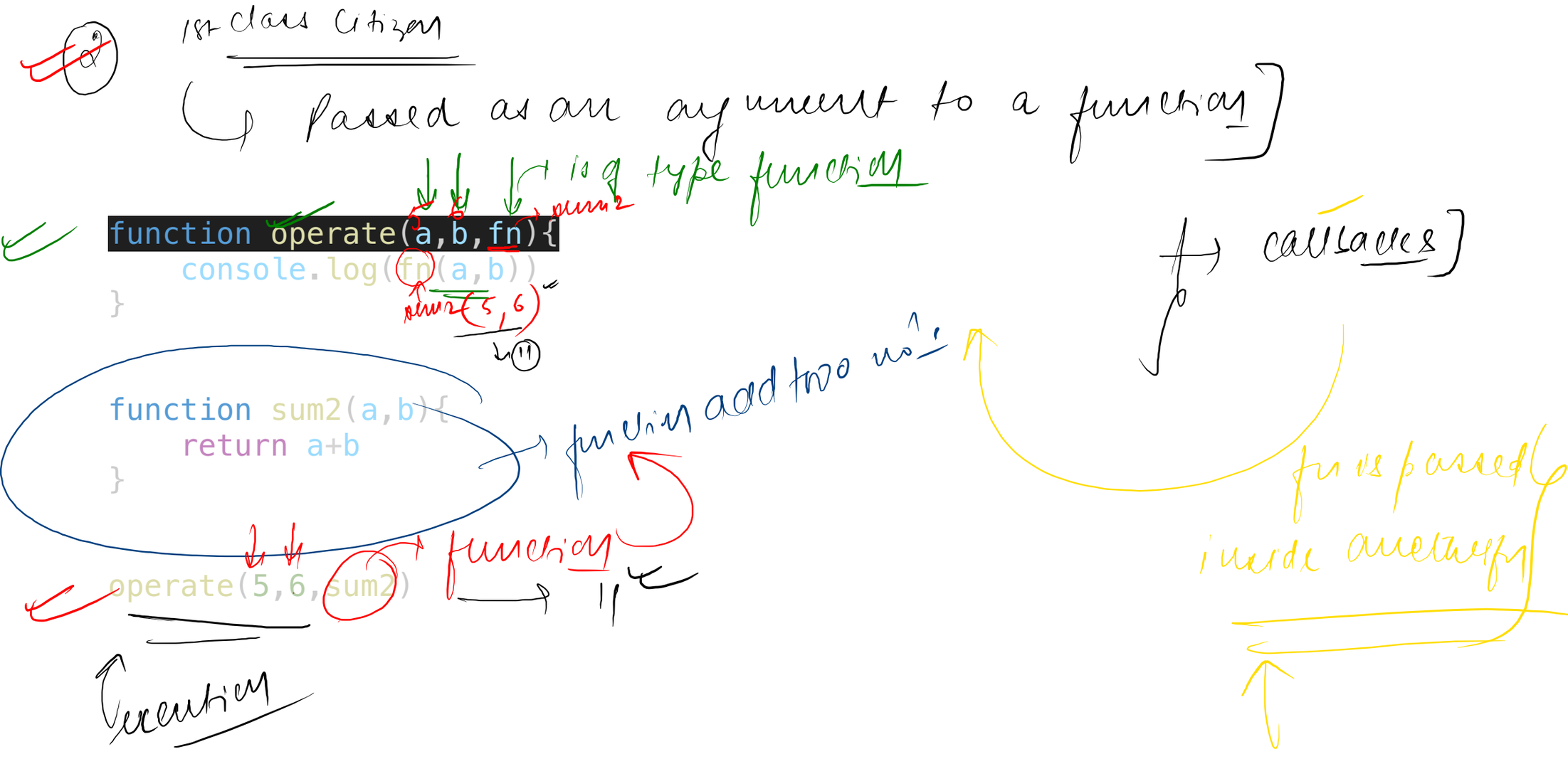
① assigned to a variable

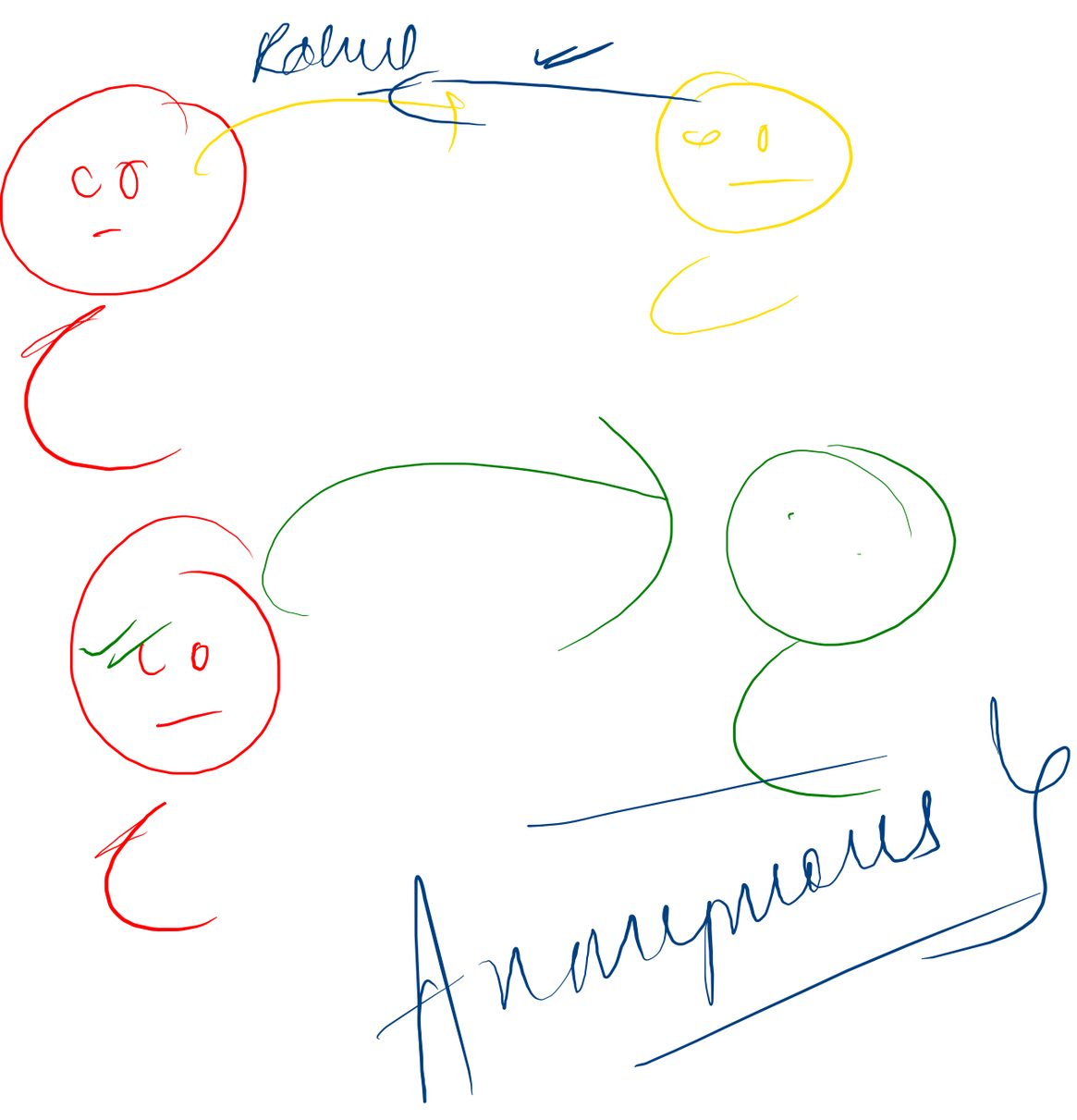
② can be passed as an argument

③ fn can be returned from a function

Interview


data type
variable
function


1st class citizen
Passed as an argument to a function
is a type function
function operate(a,b,fn){
console.log(fn(a,b))
}
sum2(5,6)
11
function sum2(a,b){
return a+b
}
function add two no's
operate(5,6,sum2)
function
Execution
callback
fn is passed inside another


Kolmio
Ympyrät

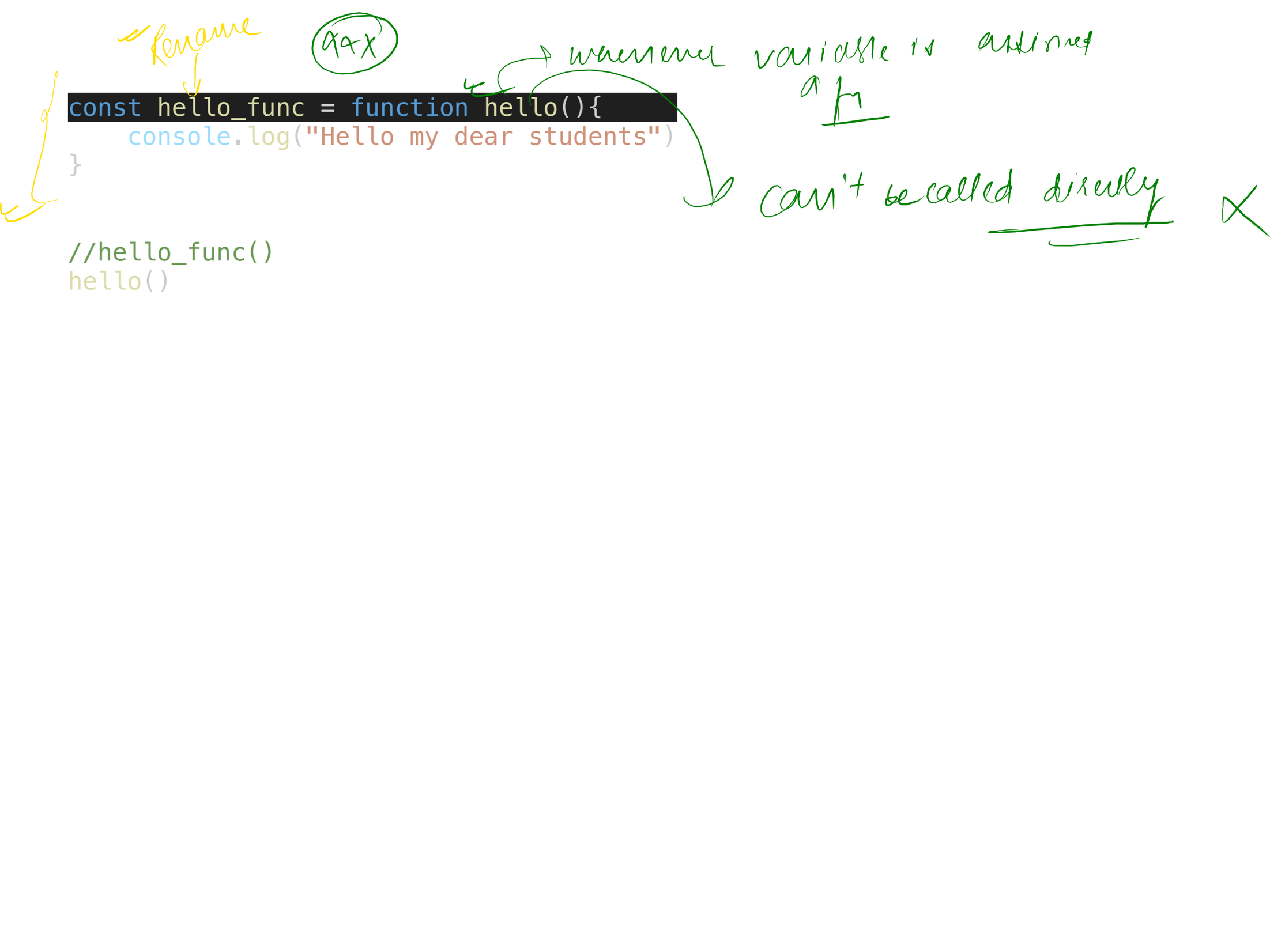
Assigned to a variable

```
const hello_func = function hello(){
    console.log("Hello my dear students")
}

hello_func()
```

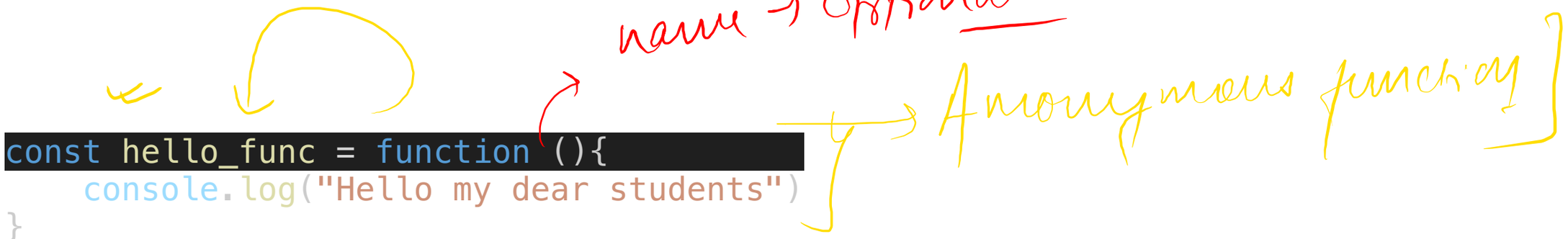
JS → No data type

Return x No type :)

```
const sum_func = function sum1(a,b){
    return a+b
}

console.log(sum_func(5,6))
```




```
const hello_func = function hello(){
    console.log("Hello my dear students")
}


//hello_func()
hello()
```




```
const hello_func = function (){
    console.log("Hello my dear students")
}


hello_func()
```

(fn) → 1st class citizen

① return function from a fn

write inside ()

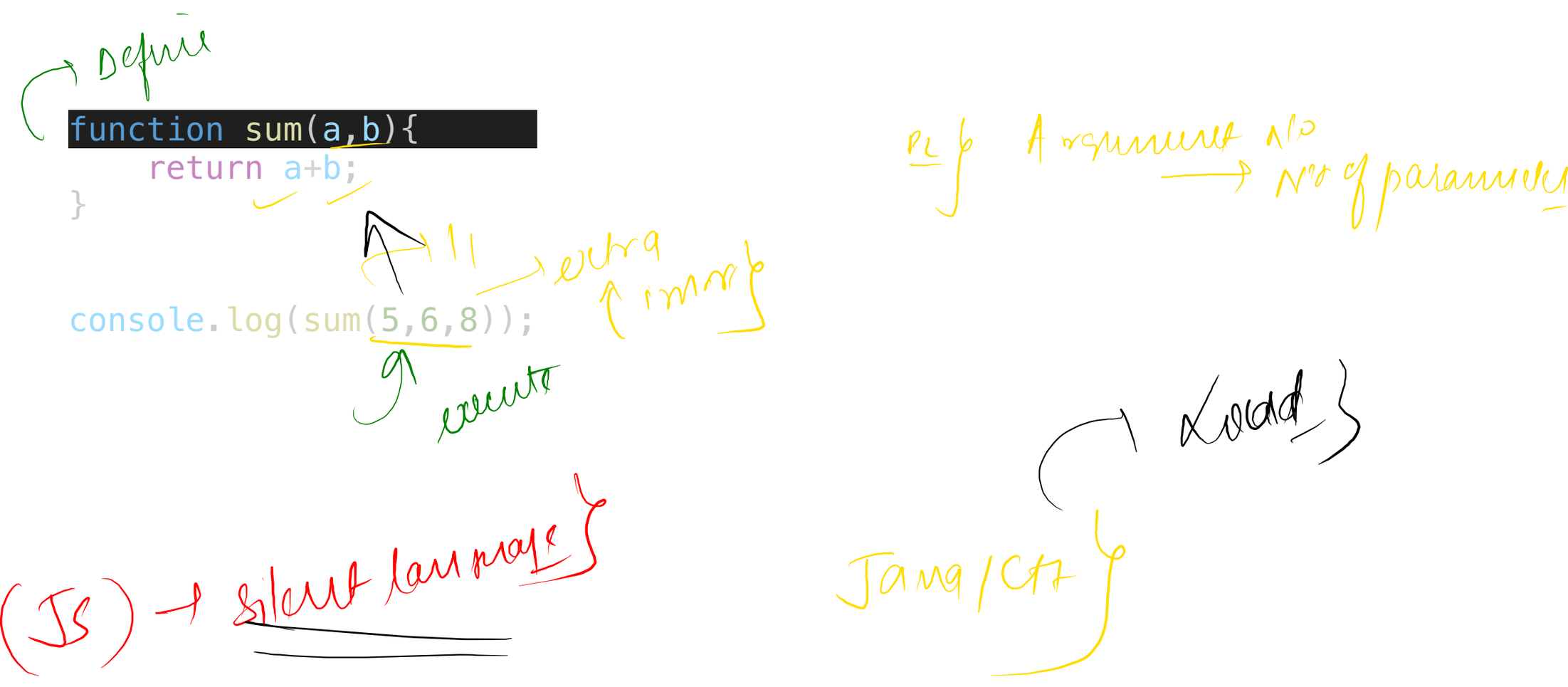
```
function return_greet_fn(){
    return function(){
        console.log("Hello Students")
    }
}
```

fn

return_greet_fn()() → Hello Students

executing

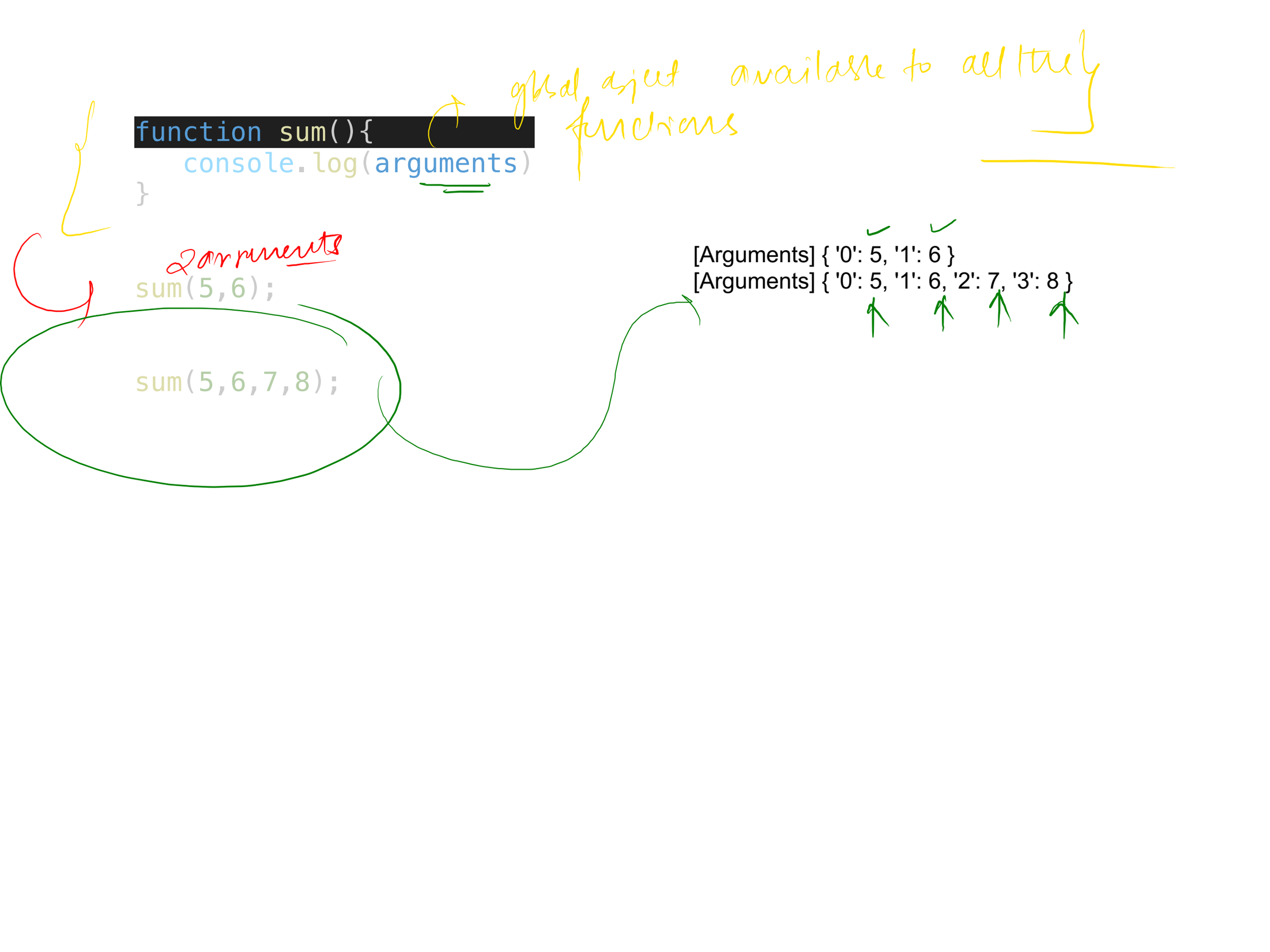
```
function sum(a,b){
    return a+b;
}


console.log(sum(5,6,8));
```


Define
function sum(a,b){
return a+b;
}
console.log(sum(5,6,8));
extra
execute
JS → silent language
Arguments → No of parameters
Java / C++
Load

(JS) → function → magical

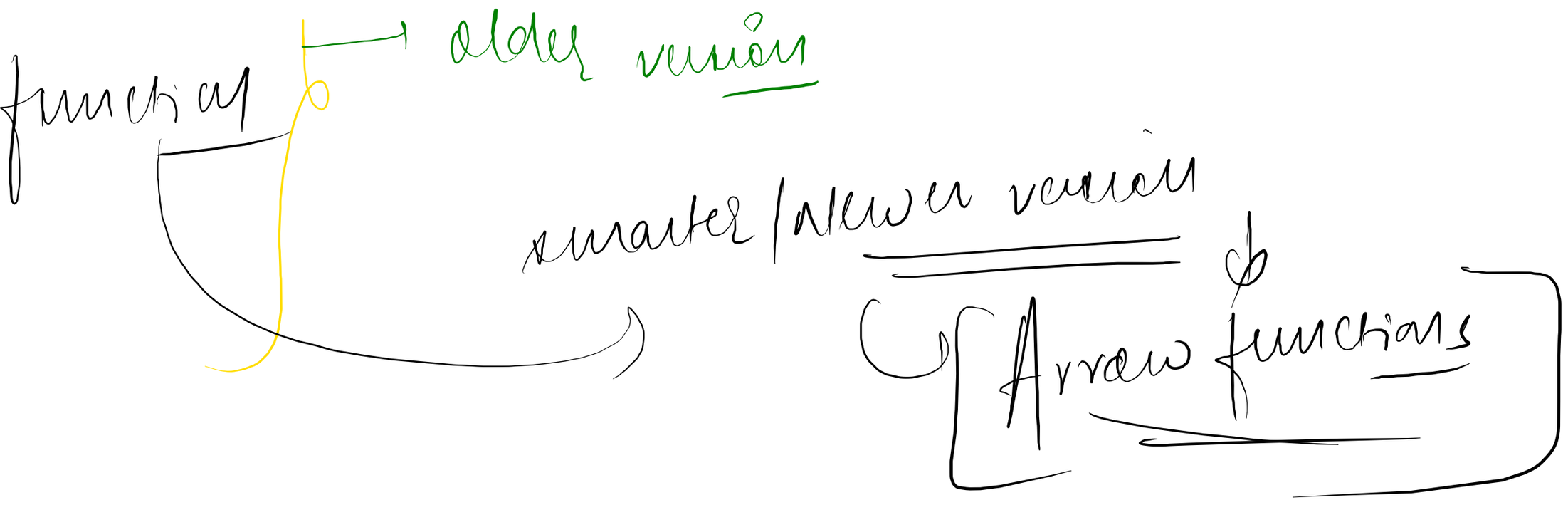
(fn) → Any number of arguments pass (without any parameter)

use that also


global object available to all the functions
function sum(){
console.log(arguments)
}
2 arguments
sum(5,6);
sum(5,6,7,8);
[Arguments] { '0': 5, '1': 6 }
[Arguments] { '0': 5, '1': 6, '2': 7, '3': 8 }

```
function sum(){
    console.log(arguments)


    let sum =0
    for(i=0;i<arguments.length;i++){
     sum = sum + arguments[i]
    }
    return sum
}


console.log(sum(5,6));


console.log(sum(5,6,7,8));
```

Iterators are objects


function
older version
smaller/newer version
Arrow functions

Arrow function 8:36 PM 8:41 PM

Arrow functions

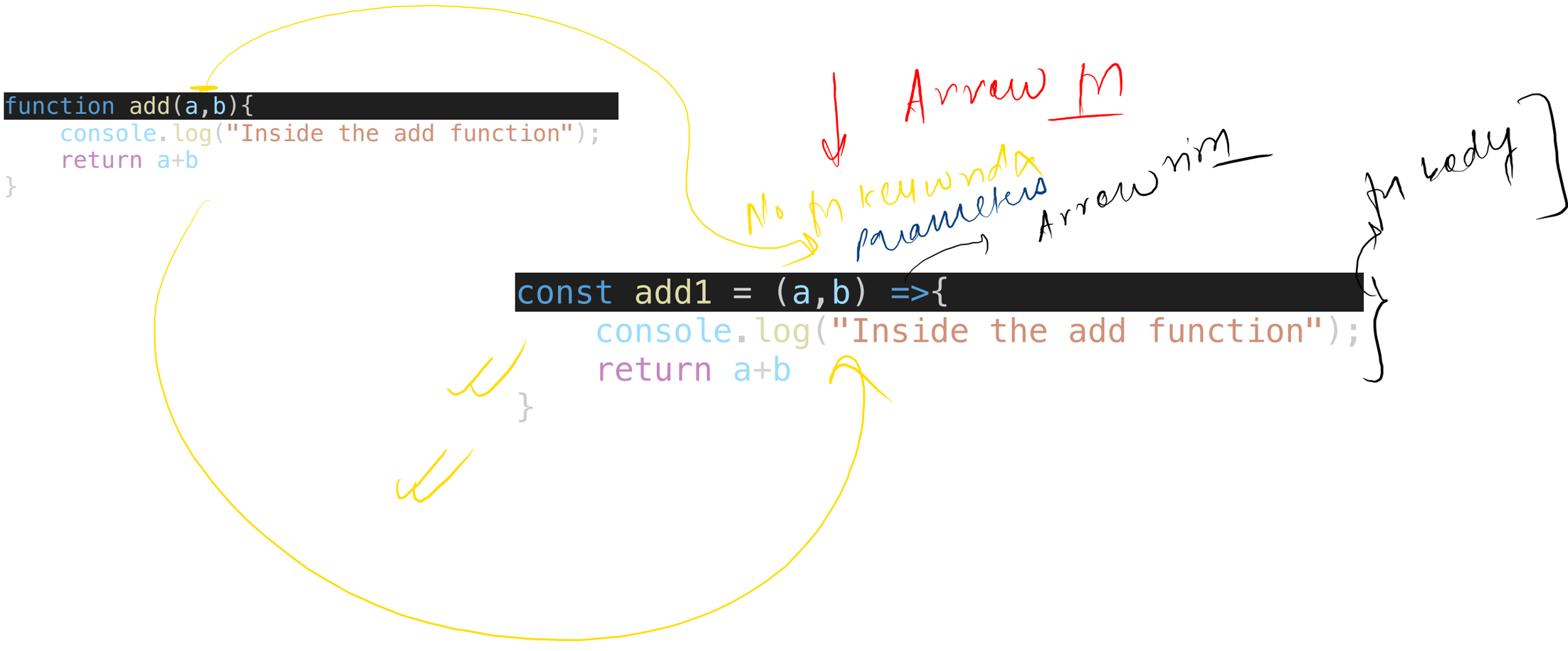
↳ Kaunsda's

↓

Syntactical sugar

f write cus

↳ get Money


function add(a,b){
console.log("Inside the add function");
return a+b
}
Arrow fn
No fn keyword
parameters
Arrow fn
fn body
const add1 = (a,b) =>{
console.log("Inside the add function");
return a+b
}

function hello ( ) {
    console.log ("Hello students")
}

to Arrow !

const hello = ( ) => { console.log ("Hello students") }

Arrow

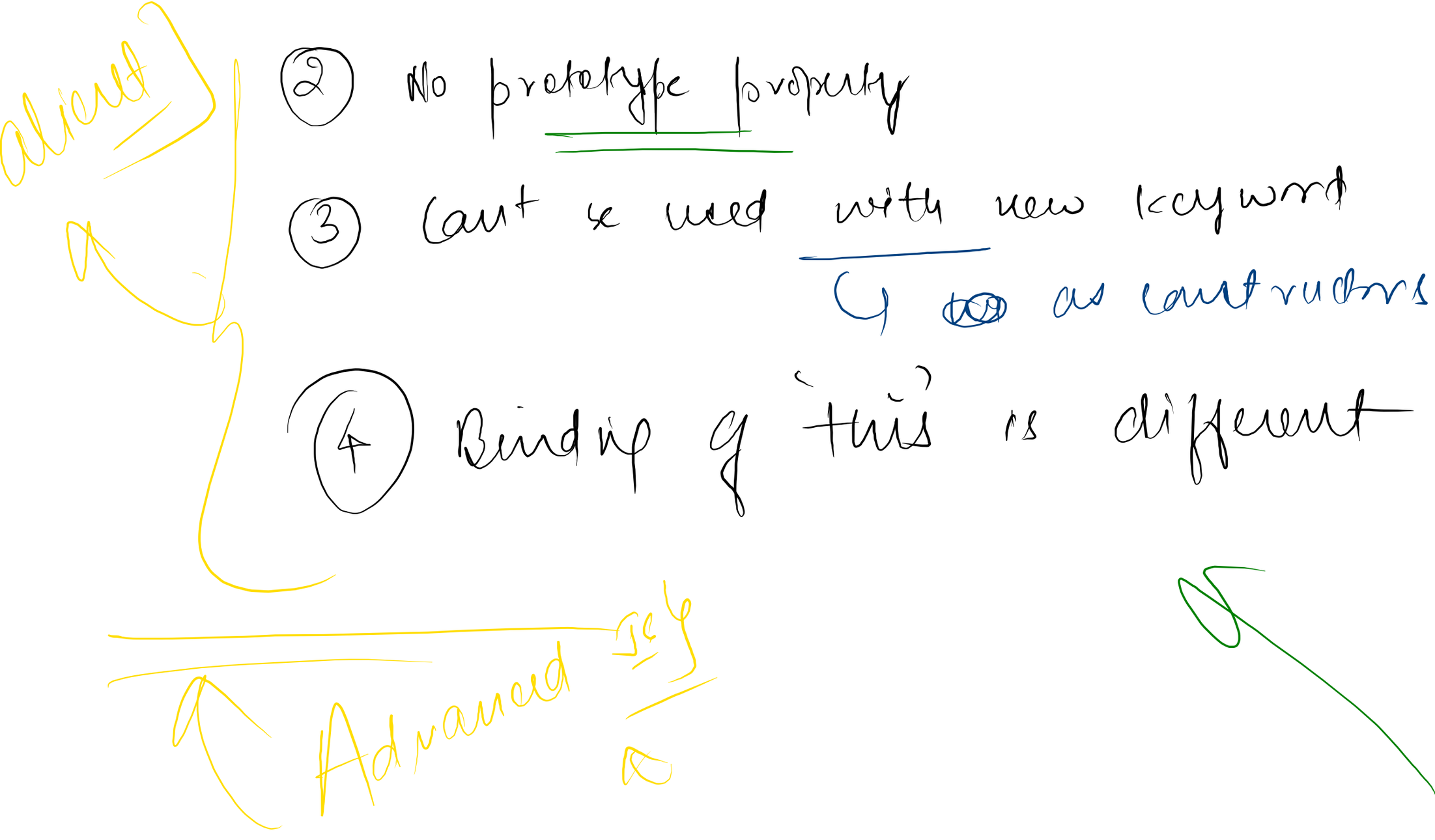
↳ Write less

↳ Do more

Limitations of arrow fn :-

① can't access the arguments object

↳ can't have the variable no of arguments

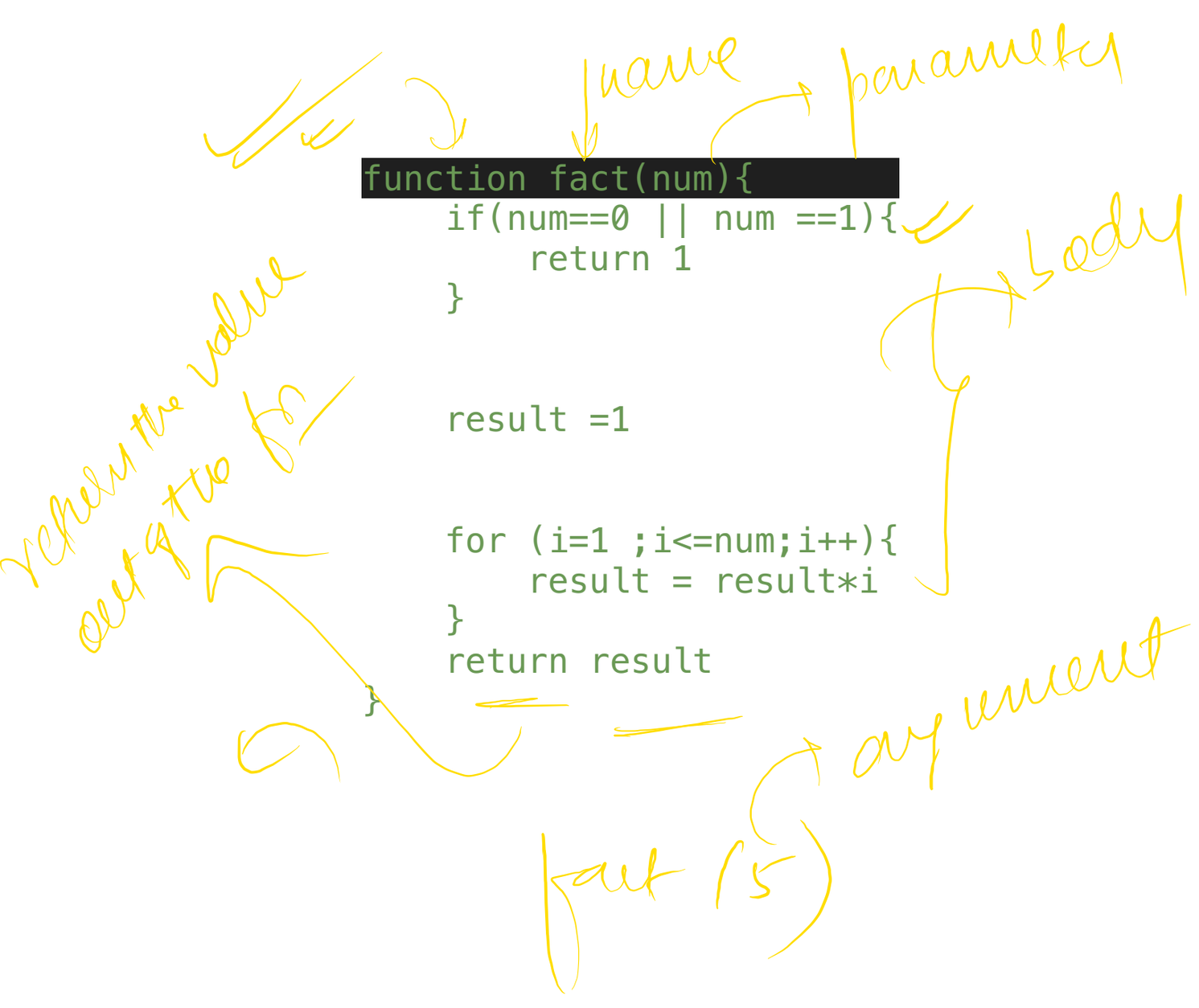

(2) No prototype property
(3) Can't be used with new keyword
as constructors
(4) Binding of this is different
Advanced JS

functions

① Block of code

② Reusability

"Write Once, Execute Many times"


function fact(num){
if(num==0 || num ==1){
return 1
}
result =1
for (i=1 ;i<=num;i++){
result = result*i
}
return result
}
name
parameter
body
return the value
argument
fact(5)

(fn) → super power

Read variable no of arguments

object → which holds all the arguments

```
function sum(){
    console.log(arguments)


    let sum =0
    for(i=0;i<arguments.length;i++){
     sum = sum + arguments[i]
    }
    return sum
}
```

Arrow function Syntactical sugar

Write less

(1) code More

```
sum = (a, b) => { return a+b }
```

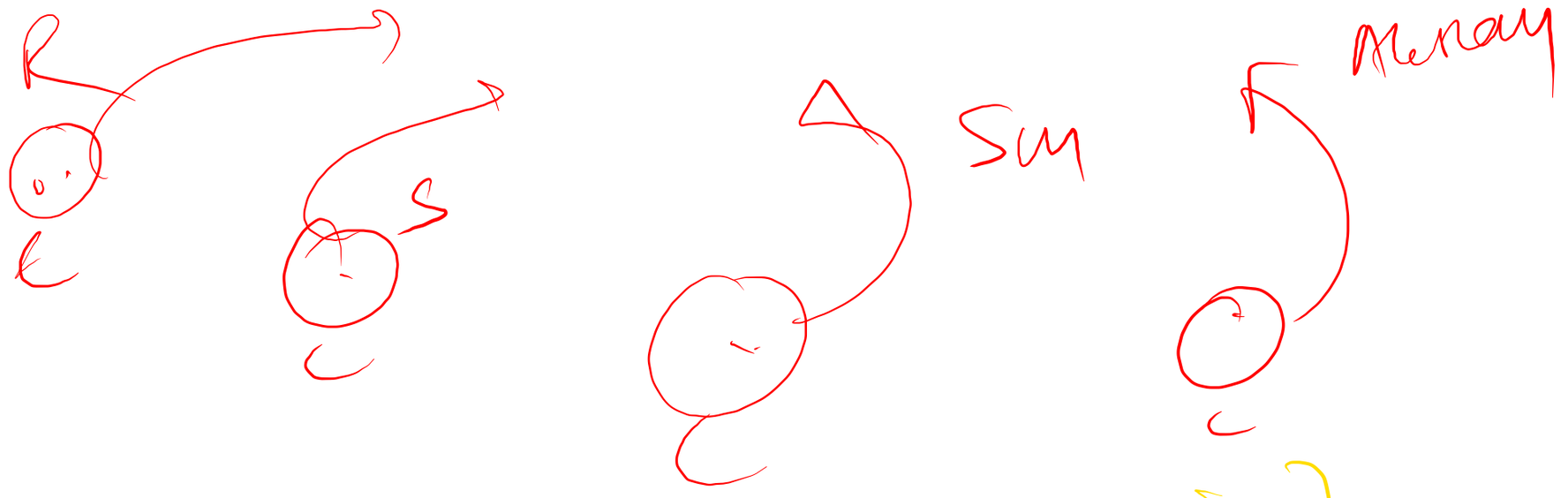
Parameters — Arrow — fn body

Give a number to a fn, determine if it is even or odd

Array

→ used to hold data in order

Class



Store data → order is maintained

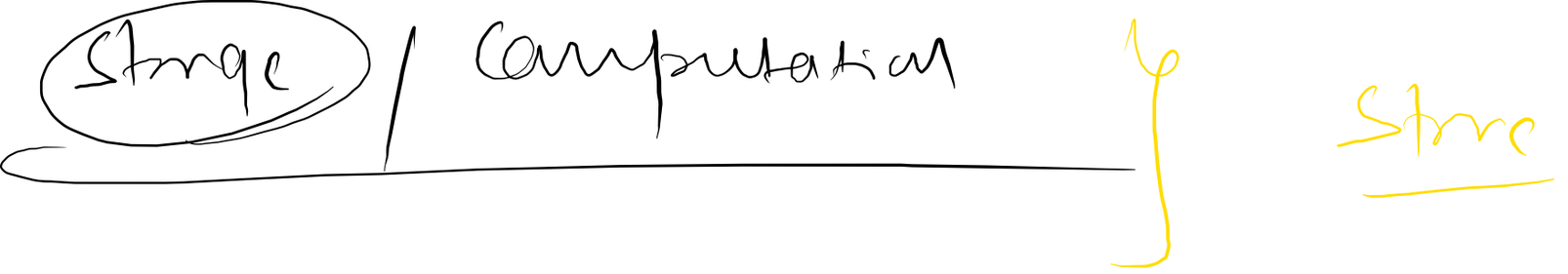
Data structure

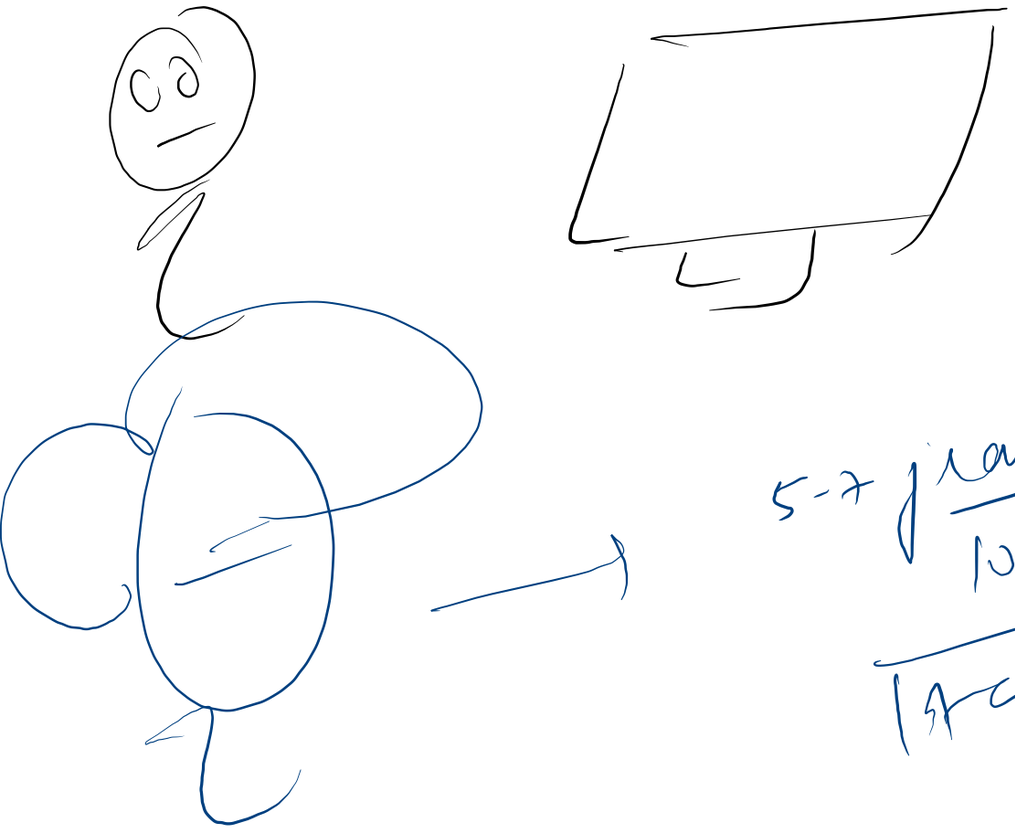
Data Structure → what & why ?

Driven Decision

company

- Data
  - Storage
  - computation


Storage
Computation
Store



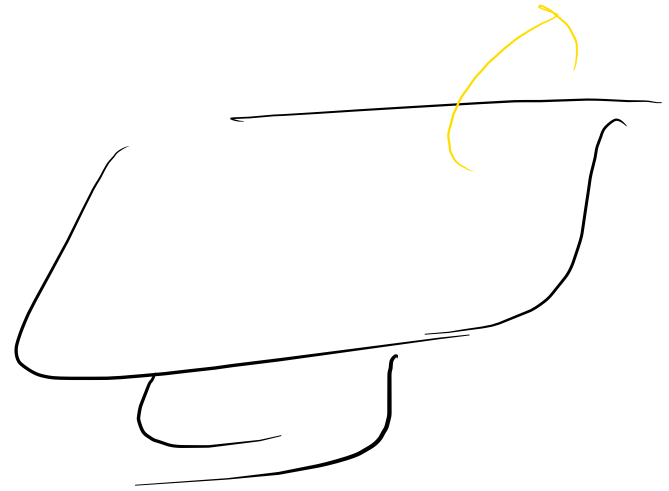
3-4 jeans | 5-6 tshirt

future

5-7 jeans
10 tshirts

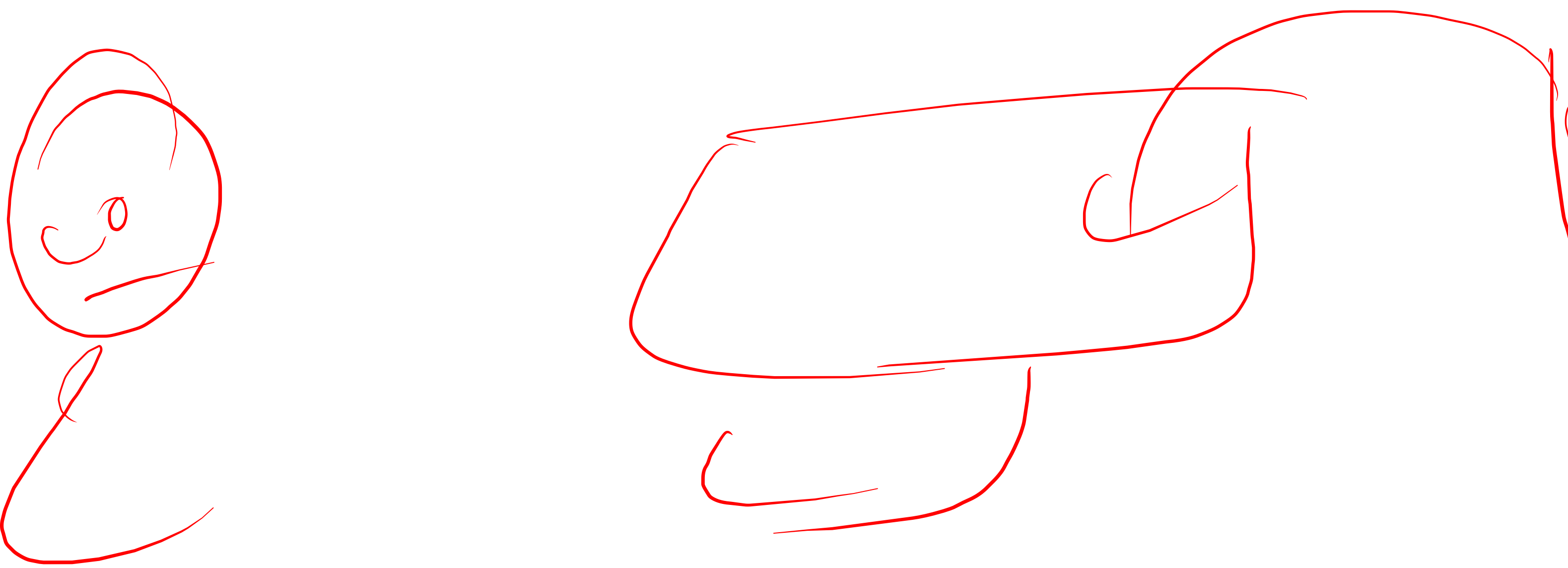
Big space left

Destination

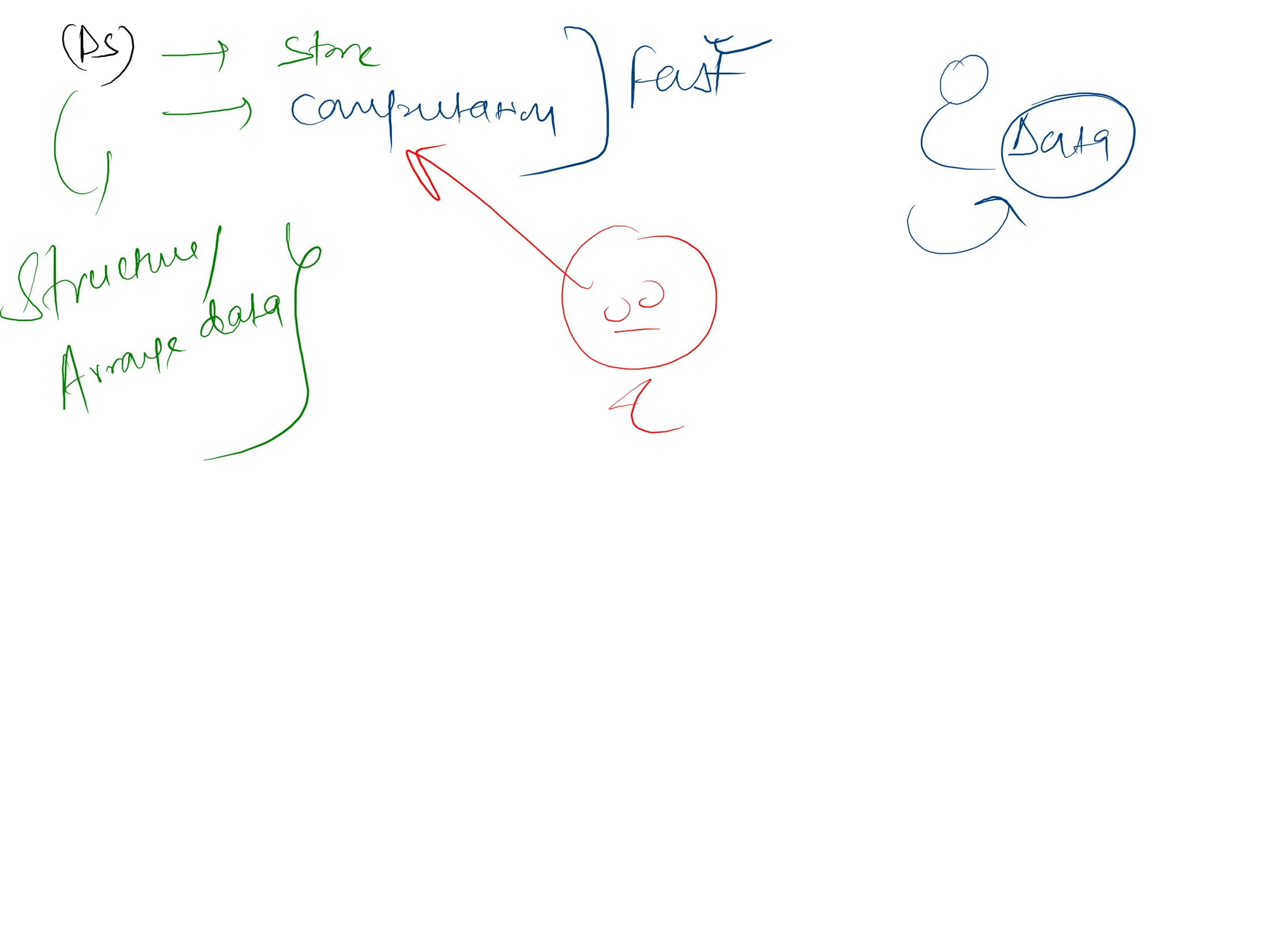


yellow signs

lots of turns



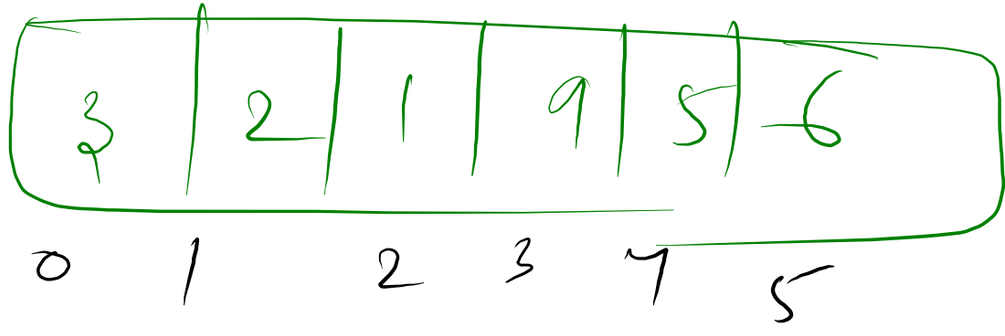
find it fast!!!


(DS)
Store
Computation
Fast
Structure/
Array data
Data

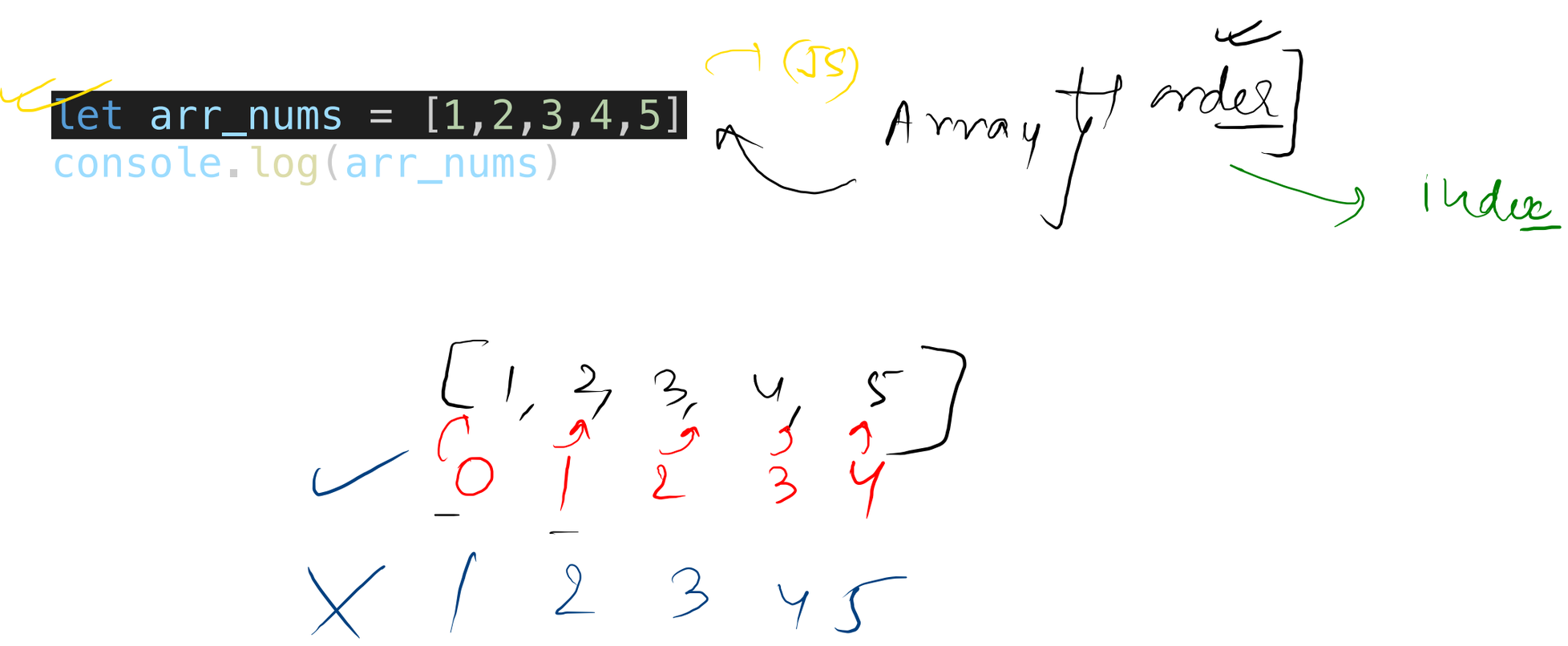
(DS) → Array → Data → ordered Manner

3, 2, 1, 9, 5, 6

Array =

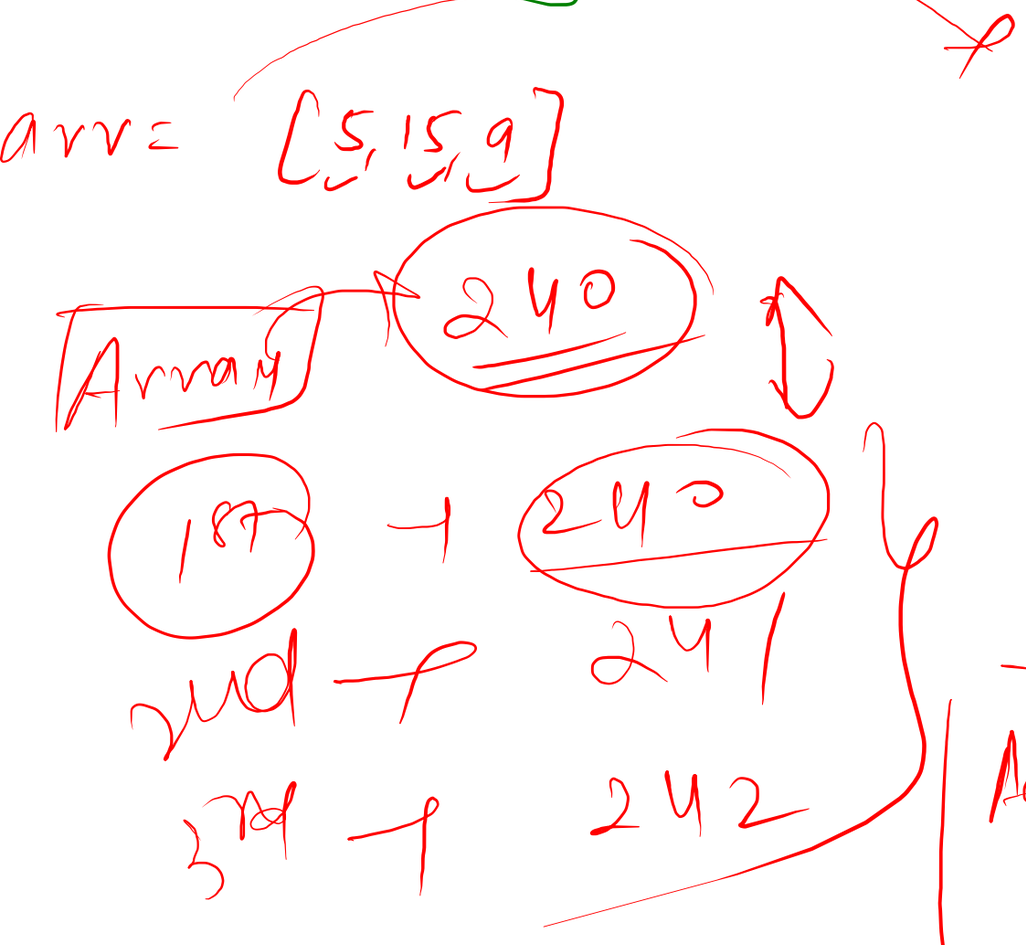


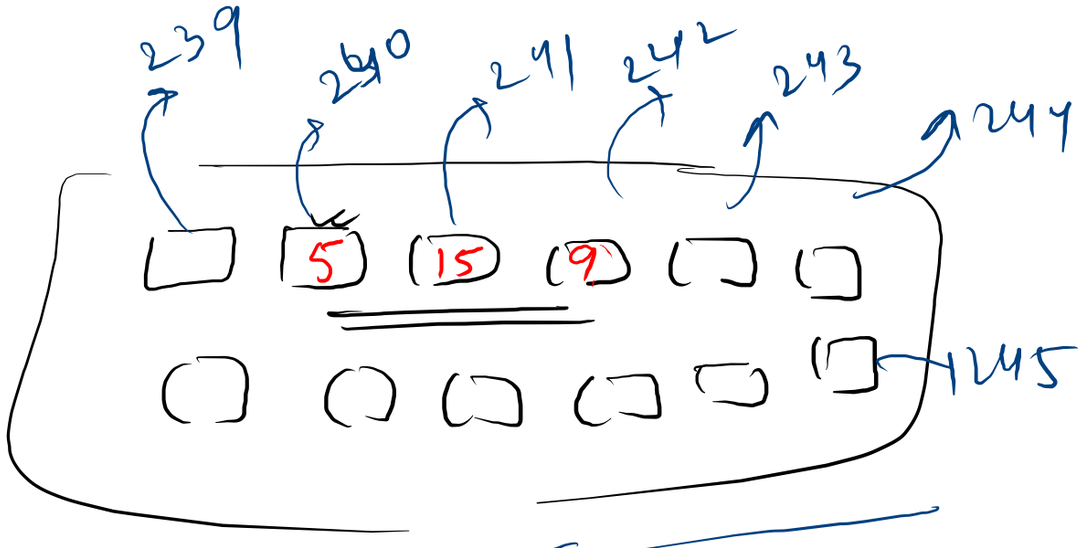
```
let arr_nums = [1,2,3,4,5]
console.log(arr_nums)
```



Array DS → Memory → continuous location

arr = [5, 15, 9]

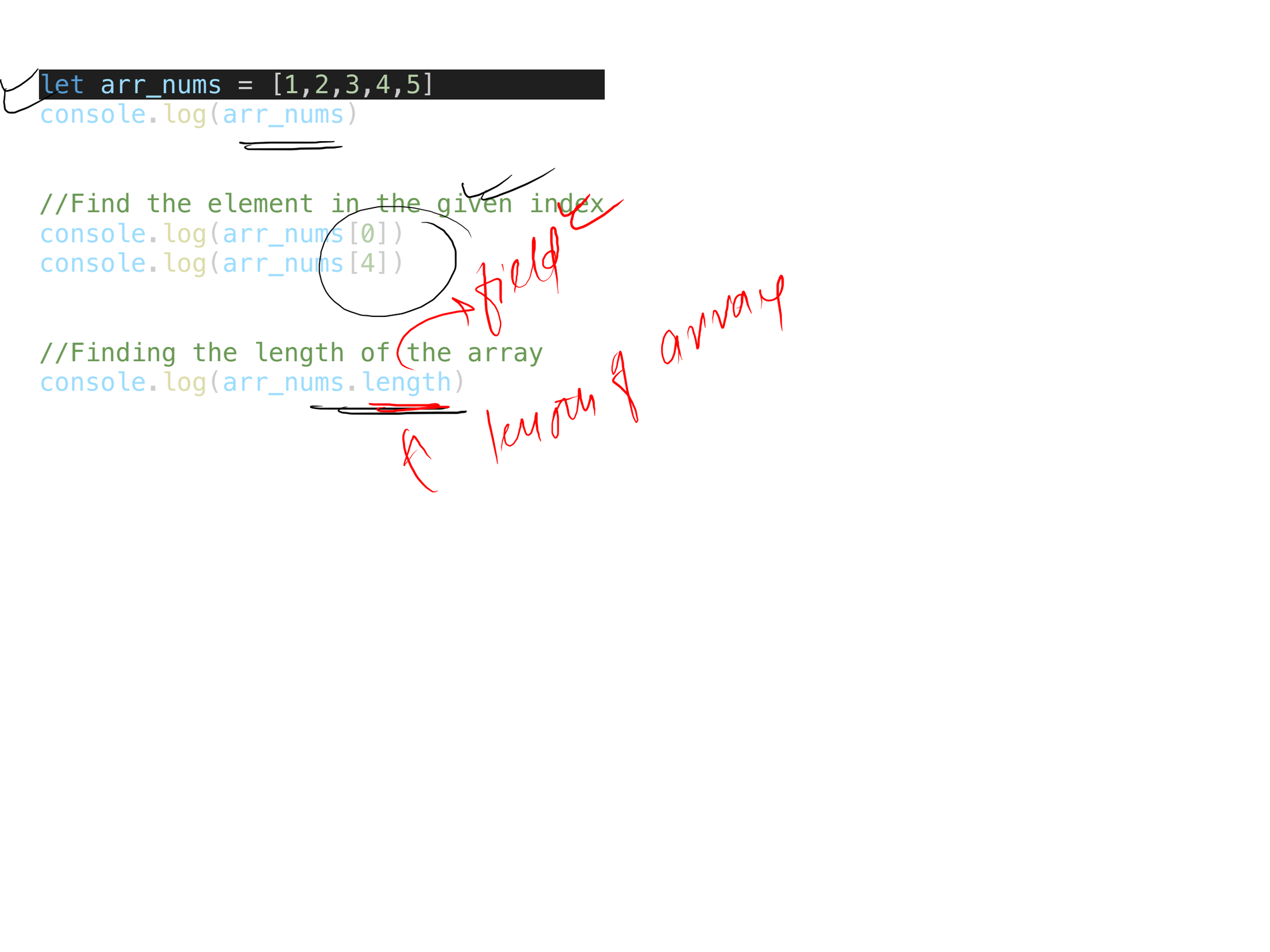




Address of any element
= address of array + i

Array → (DS) | index → index

```
let arr_nums = [1,2,3,4,5]
console.log(arr_nums)


//Find the element in the given index
console.log(arr_nums[0])
console.log(arr_nums[4])


//Finding the length of the array
console.log(arr_nums.length)
```



arr = [3, 2, 9, 1, 4, 5]

0 1 2 3 4 5

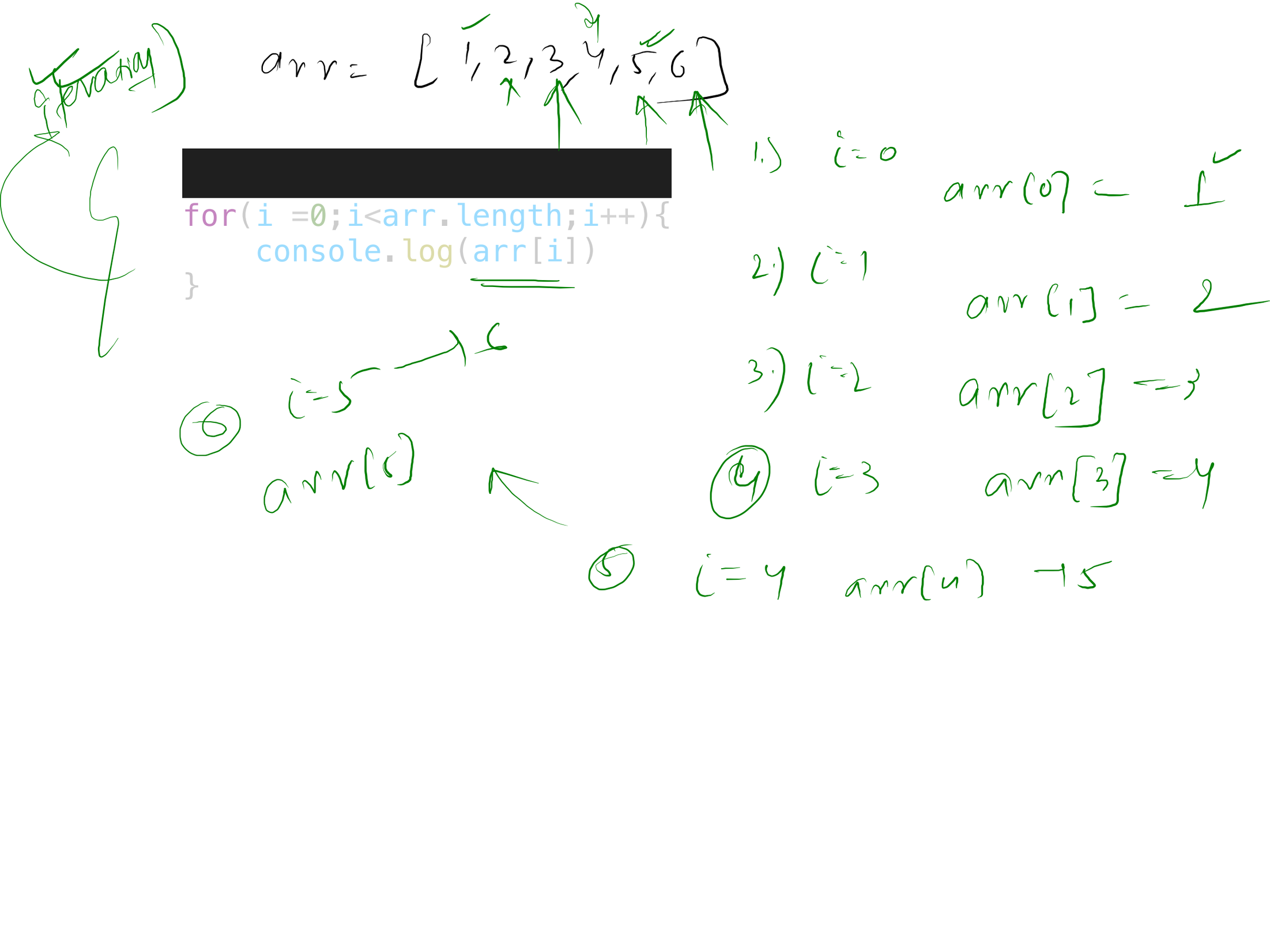
(JS) → undefined

valid values of index = [0, arr.length - 1]

= [0, 5]

Any other index → Error → index out of bound exception

Java / C++


Iteration
arr = [1,2,3,4,5,6]
for(i =0;i<arr.length;i++){
console.log(arr[i])
}
1.) i=0 arr[0] = 1
2.) i=1 arr[1] = 2
3.) i=2 arr[2] = 3
4 i=3 arr[3] = 4
5 i=4 arr[4] → 5
6 i=5 → 6
arr[6]

1 (fn) ← Array

↳ return the sum of all the elements

Iterate over array elements directly → No need of syntax

```
arr = [3,4,2,1,5]


for( num of arr){  //foreach loop
    console.log(num)
}
```

Decode → Practice Problems / Assignment

↳ few Model solutions

Doubt session

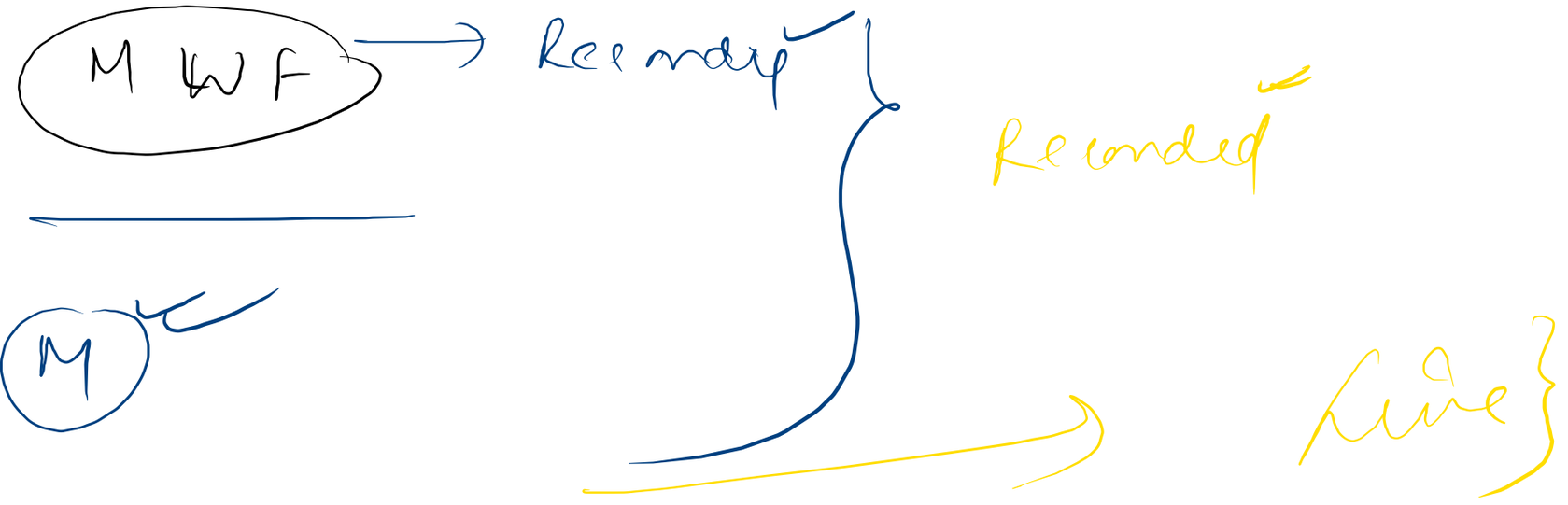
① % class nearly } ✓

② Quizzes

Assignment / Quizzes

< 3 days from the time it's uploaded!

Next session } → tomorrow 7pm ✓


M W F
Recorded
Recorded
M
Live


function return_greet_fn(){
return function(){
console.log("Hello Students")
}
}
return_greet_fn()()
"Hello students"