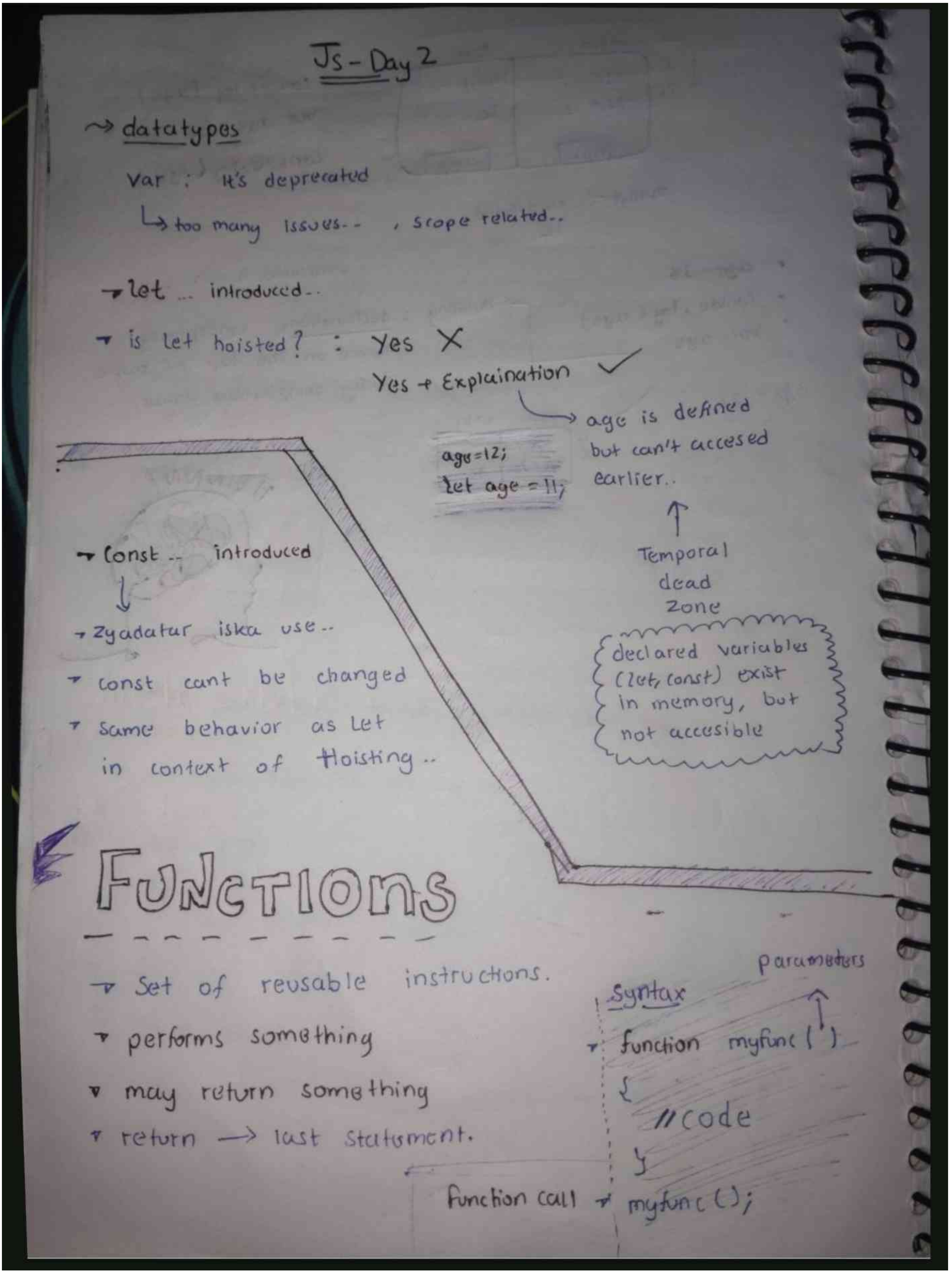


Js - Day 2

→ 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.

Syntax

parameters

```
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;
}
```

func (age) {
return age}
}
	execution

memory phase	code phase

▩ cycle1
▨ cycle2

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

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

var myfunc (not defined)	function declared later
memory phase	code phase

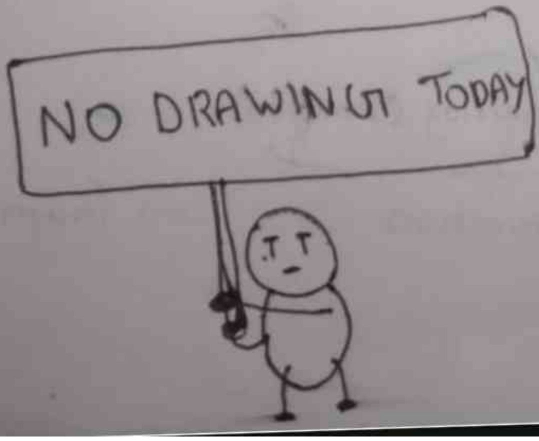
- can access function before (undefined)

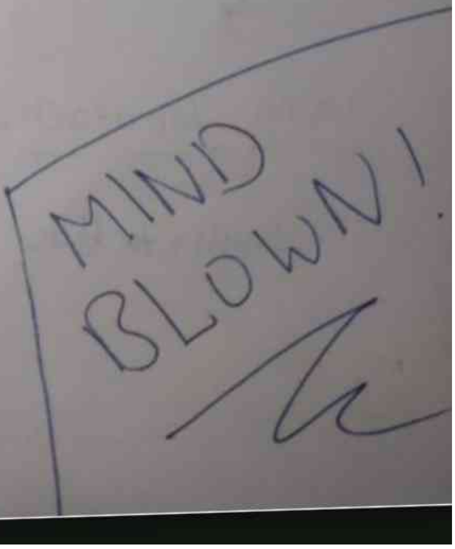
Arrow function

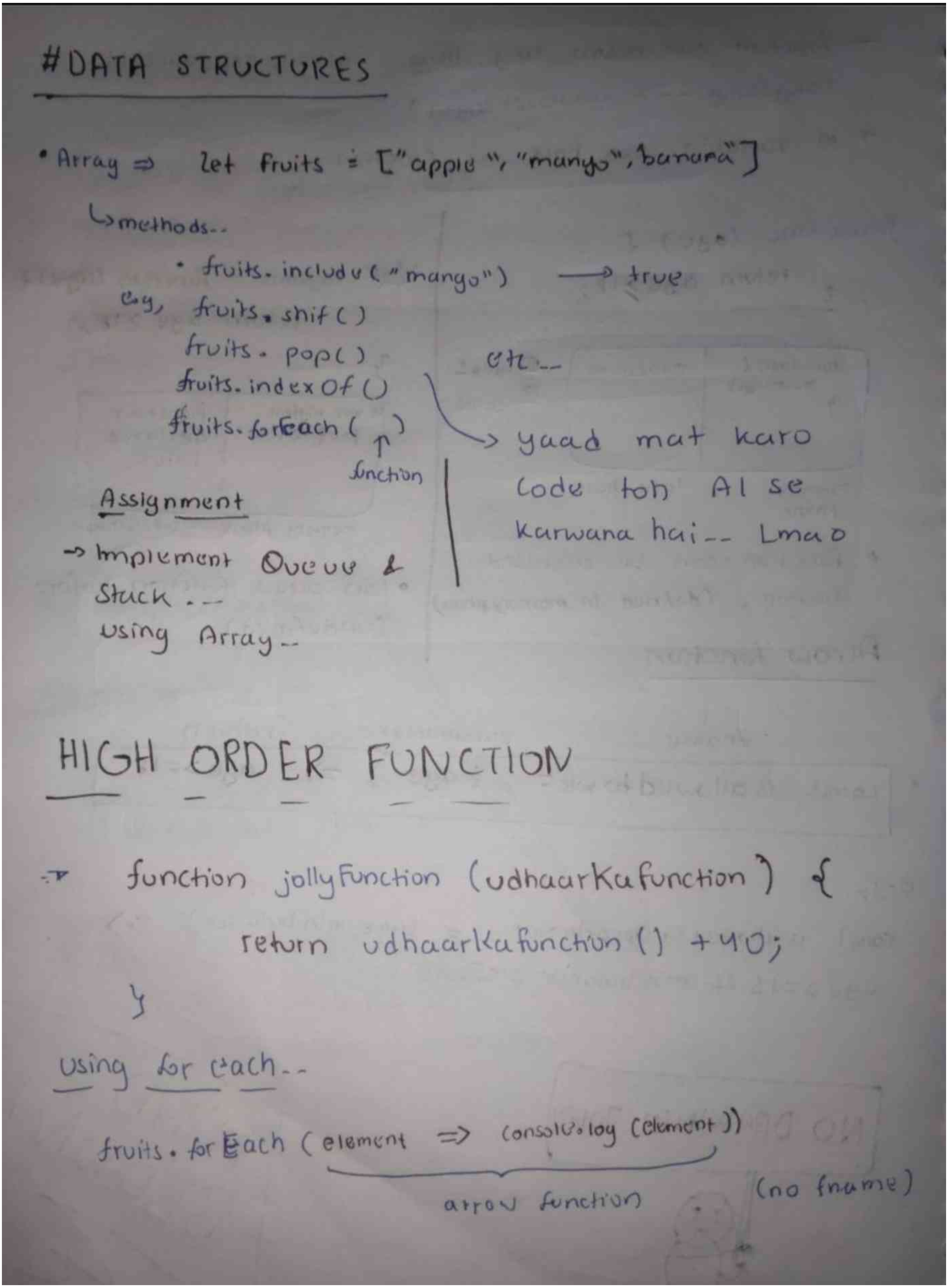
	fname	parameter	return
const	is allowed to vote =	(age) =>	age >= 18

e.g.,

const isallowed To OpenAccount = (age, min balance) =>
age >= 18 && min balance >= 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

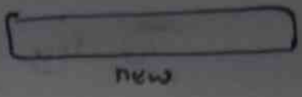
```
function jollyFunction (udhaarKafunction) {
        return udhaarKafunction () +40;
}
```

using for each..

fruits. forEach ($\underbrace{\text{element => console.log (element)}}_{\text{arrow function}}$)

(no fname)

map -> internally new array create karta hai



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

o/p => $\boxed{2, 4, 6, 8, 10, 12}$

me -> github.com/realSUDO

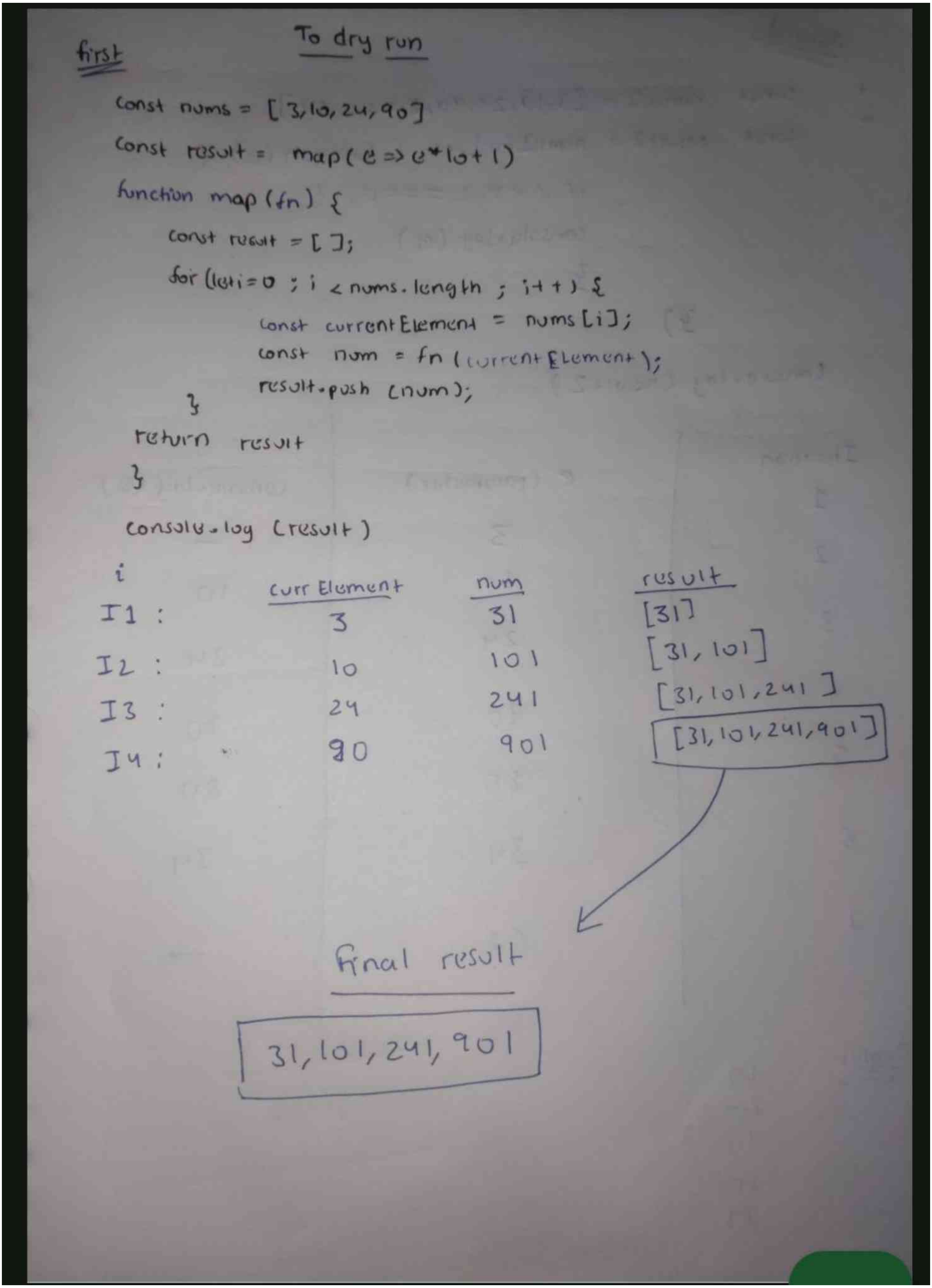
SKIP ⏭

X -> @sudo_core

discord -> @sudo.dis

LOL
PROMOTION

foreach	map
-> only iterates	-> creates new arr



first

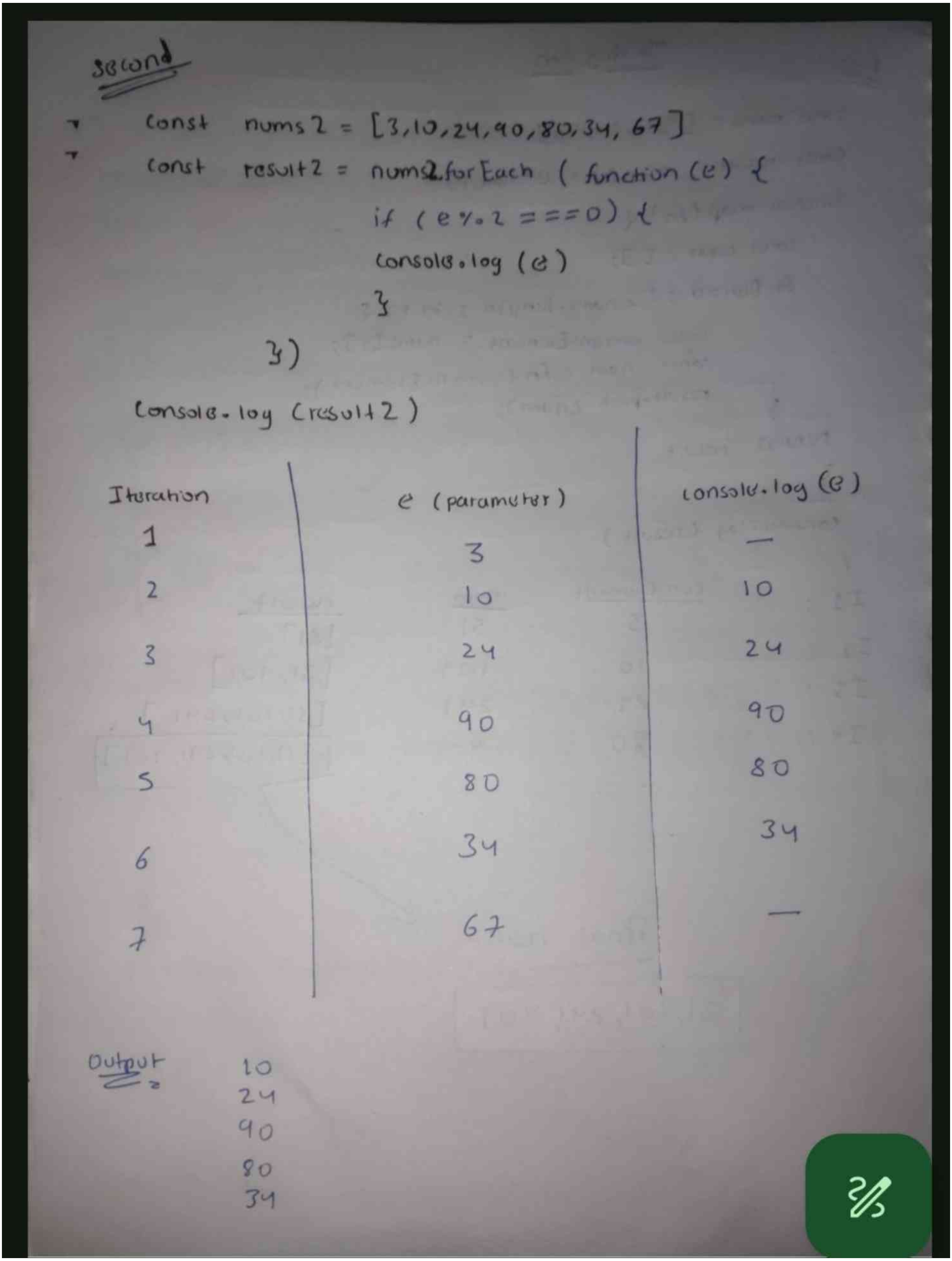
To dry run

```
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)
                  }
        })
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
```

High order function

Definition

→ A function which takes another function as an Argument or returns something from it
function

```
function x() {
    console.log("Namaste");
}

function y(x) {
    x();
}
```

high order function which takes another function as an arguments

call back function

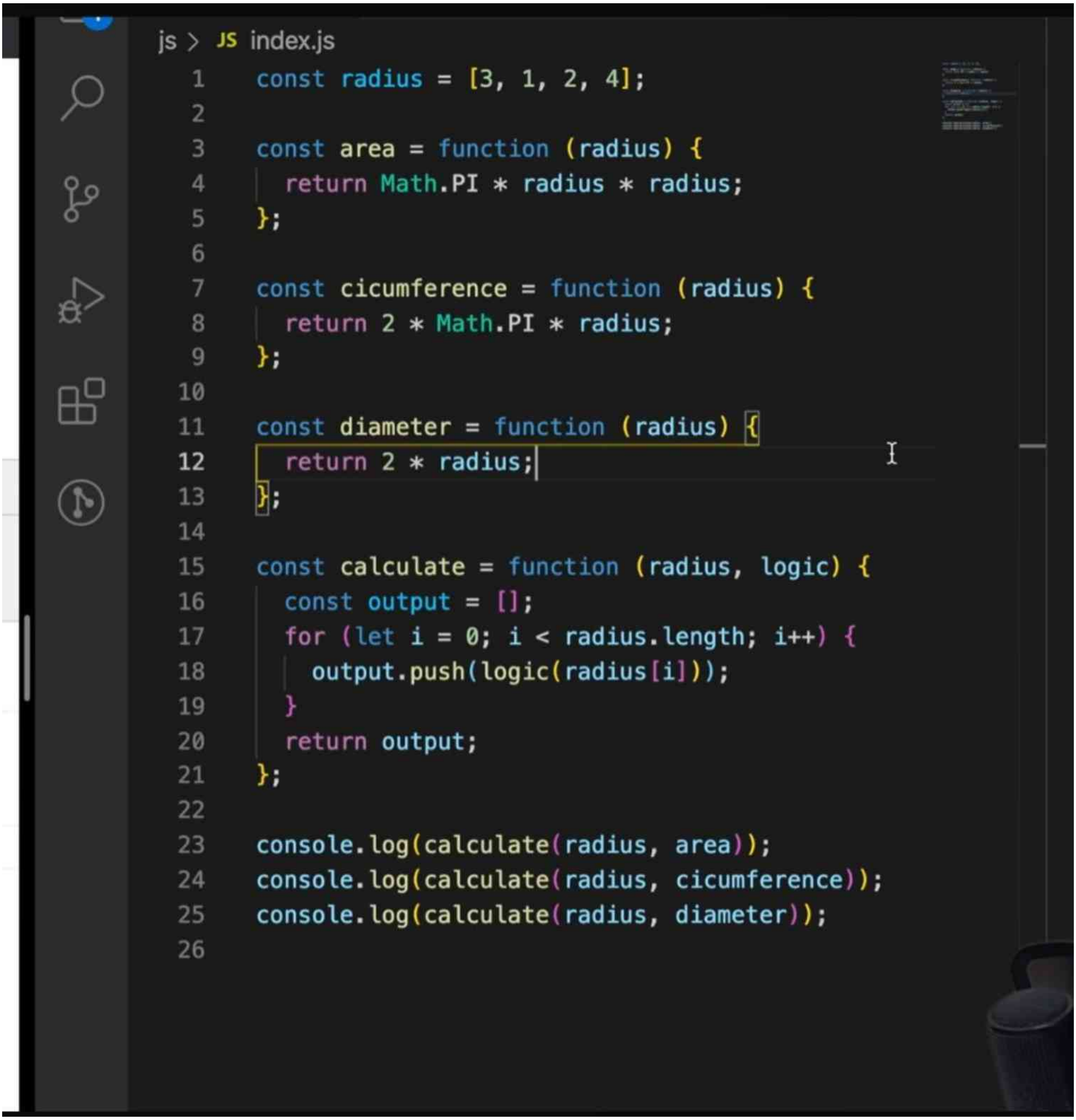
(*) Function which is Passed into higher order function is known as CallBack function

function map

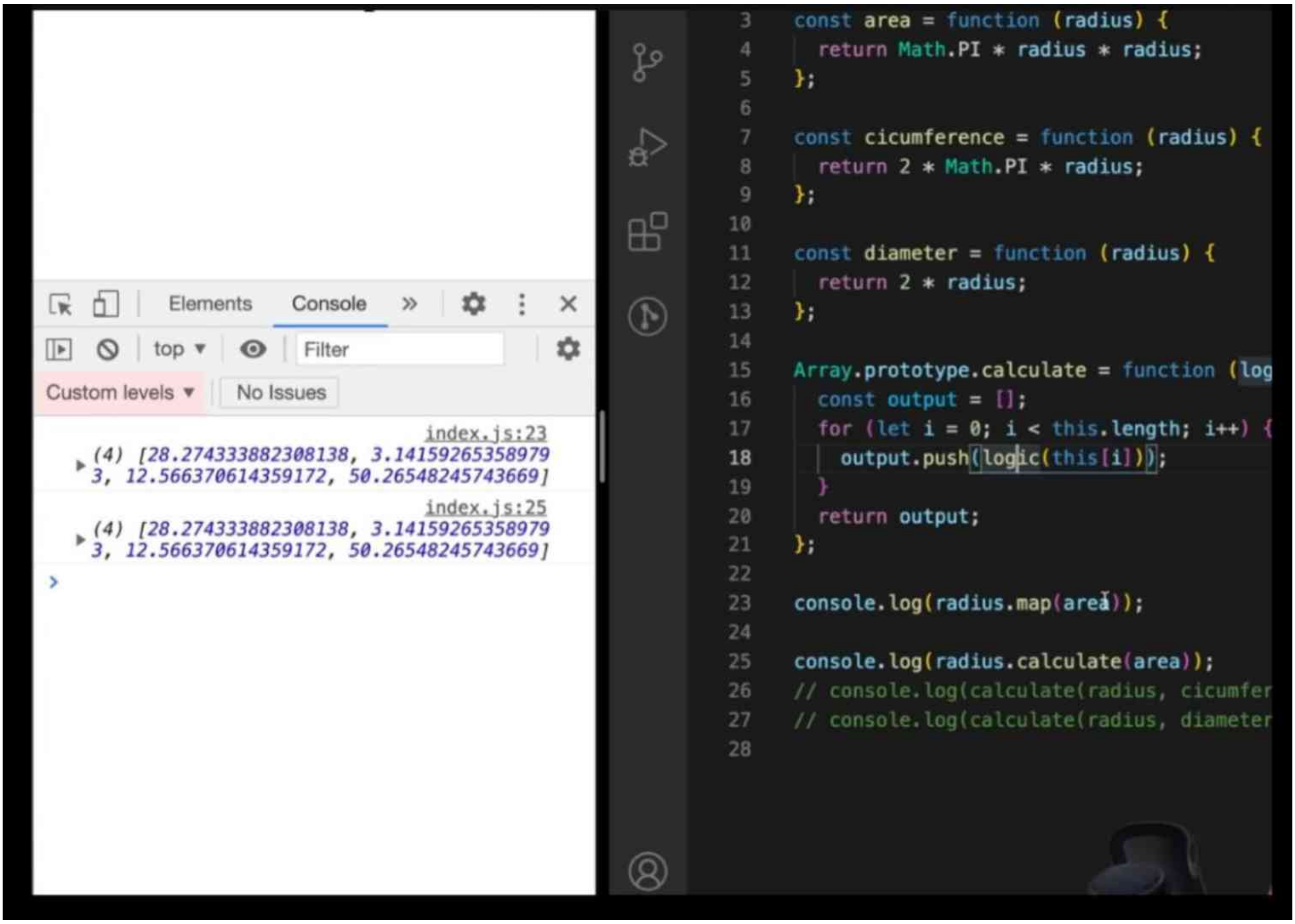
map is giving same logic

Small implementation

=> Map → is just create new Array and iterate ~~[illegible]~~ over all these of an elements in the returns ~~[illegible]~~ return the output



```javascript
const radius = [3, 1, 2, 4];

const area = function (radius) {
  return Math.PI * radius * radius;
};

const cicumference = function (radius) {
  return 2 * Math.PI * radius;
};

const diameter = function (radius) {
  return 2 * radius;
};

const calculate = function (radius, logic) {
  const output = [];
  for (let i = 0; i < radius.length; i++) {
    output.push(logic(radius[i]));
  }
  return output;
};

console.log(calculate(radius, area));
console.log(calculate(radius, cicumference));
console.log(calculate(radius, diameter));

```



```
index.js:23
▶ (4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
index.js:25
▶ (4) [28.274333882308138, 3.141592653589793, 12.566370614359172, 50.26548245743669]
>
```

```
const area = function (radius) {
  return Math.PI * radius * radius;
};

const cicumference = function (radius) {
  return 2 * Math.PI * radius;
};

const diameter = function (radius) {
  return 2 * radius;
};

Array.prototype.calculate = function (log
  const output = [];
  for (let i = 0; i < this.length; i++) {
    output.push(logic(this[i]));
  }
  return output;
};

console.log(radius.map(area));

console.log(radius.calculate(area));
// console.log(calculate(radius, cicumfer
// console.log(calculate(radius, diameter
```

~~*Prototype~~

~~When you put something onto Prototype is appear in all array~~

Prototype

When you put
Something onto Prototype
its appear in all the
array

Map() function

Transform array
while
using map() function

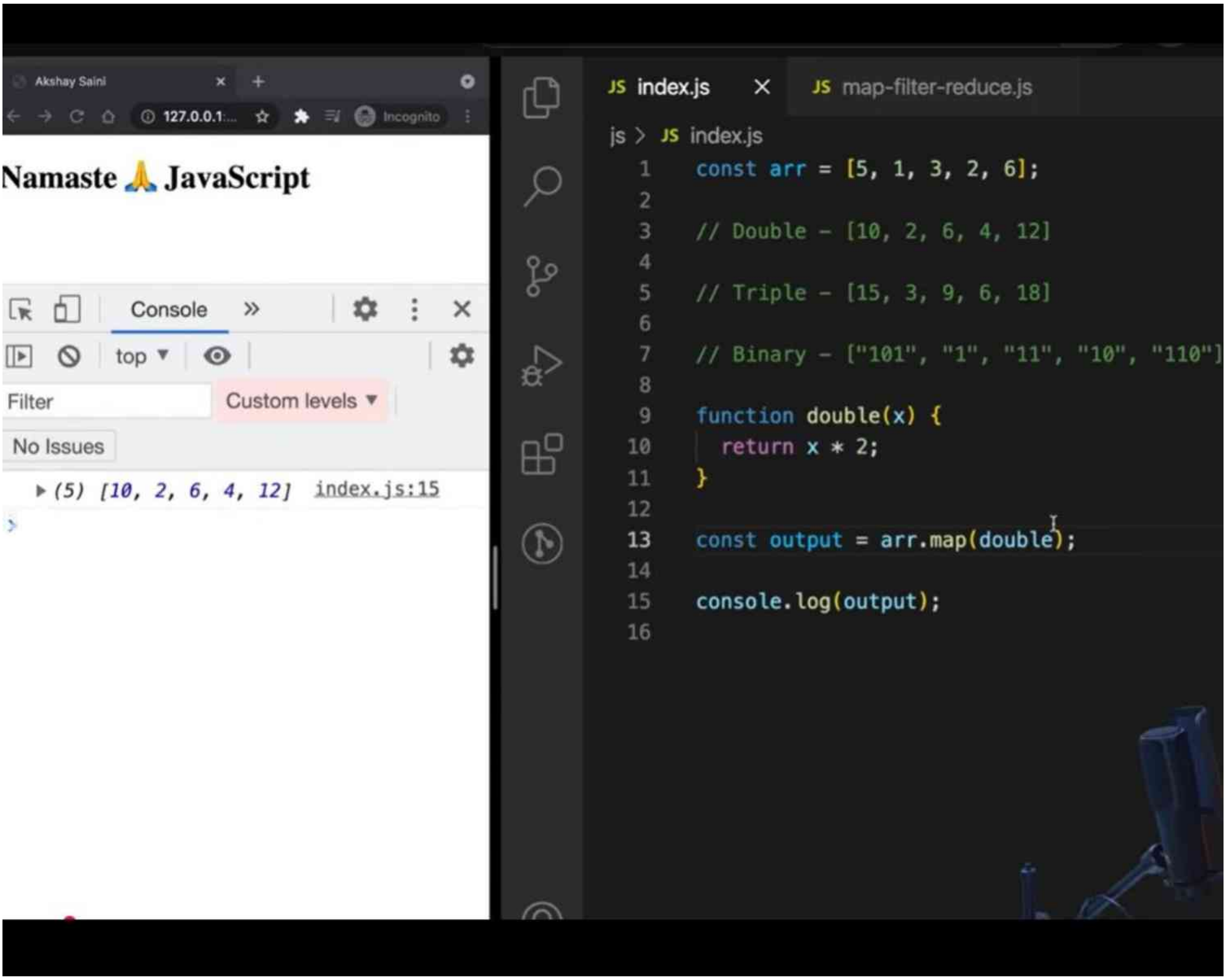
Array Transformation
↳ Transform each and every value of this array and get new Array of it

Const arr = [5, 1, 3, 2, 6)

Double :- [10, 2, 6, 4, 12]

Triple : [15, 3, 9, 6, 18]

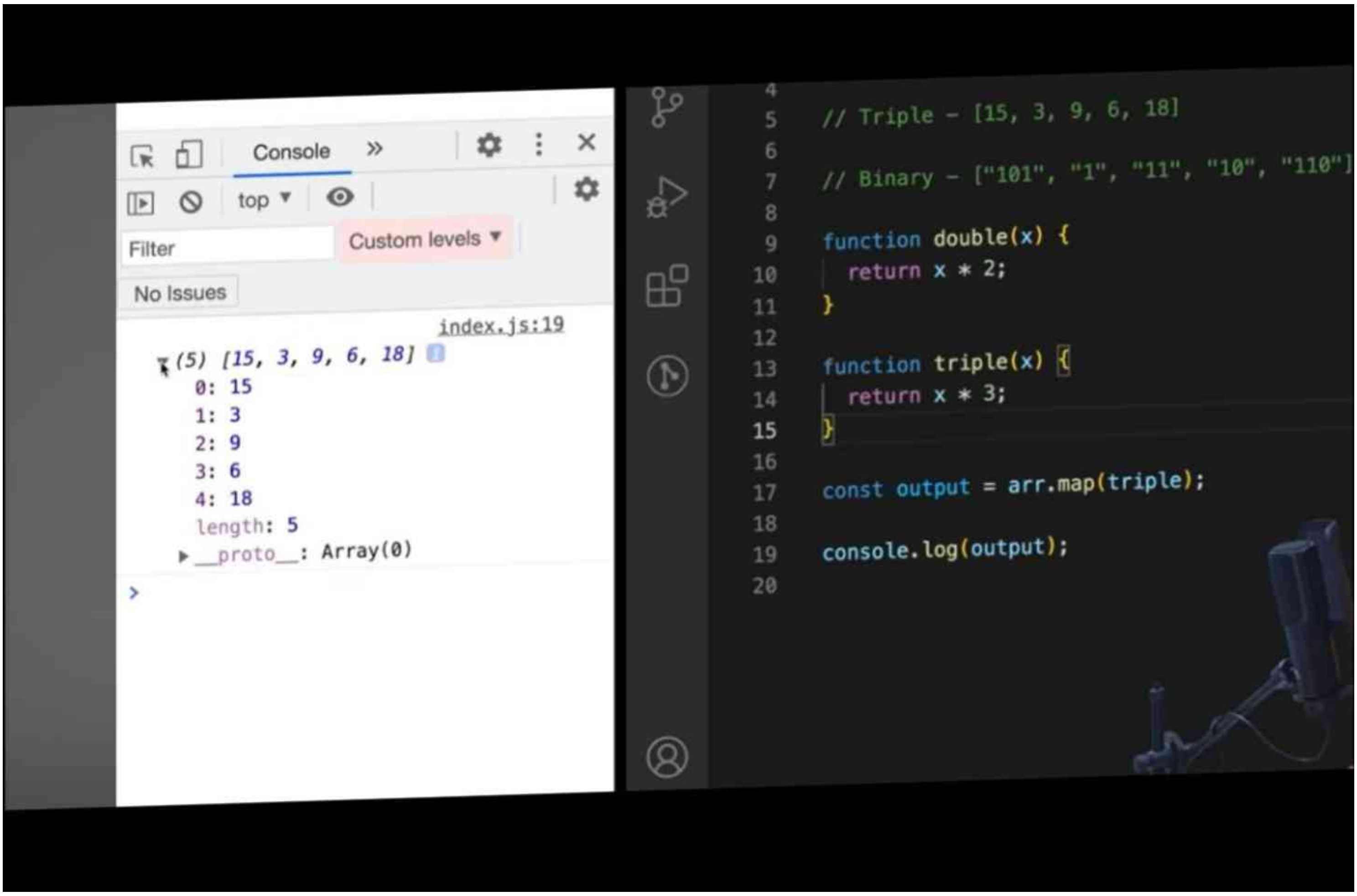
Binary : ["101", "1", "11", "10", "110"]



Namaste 🙏 JavaScript

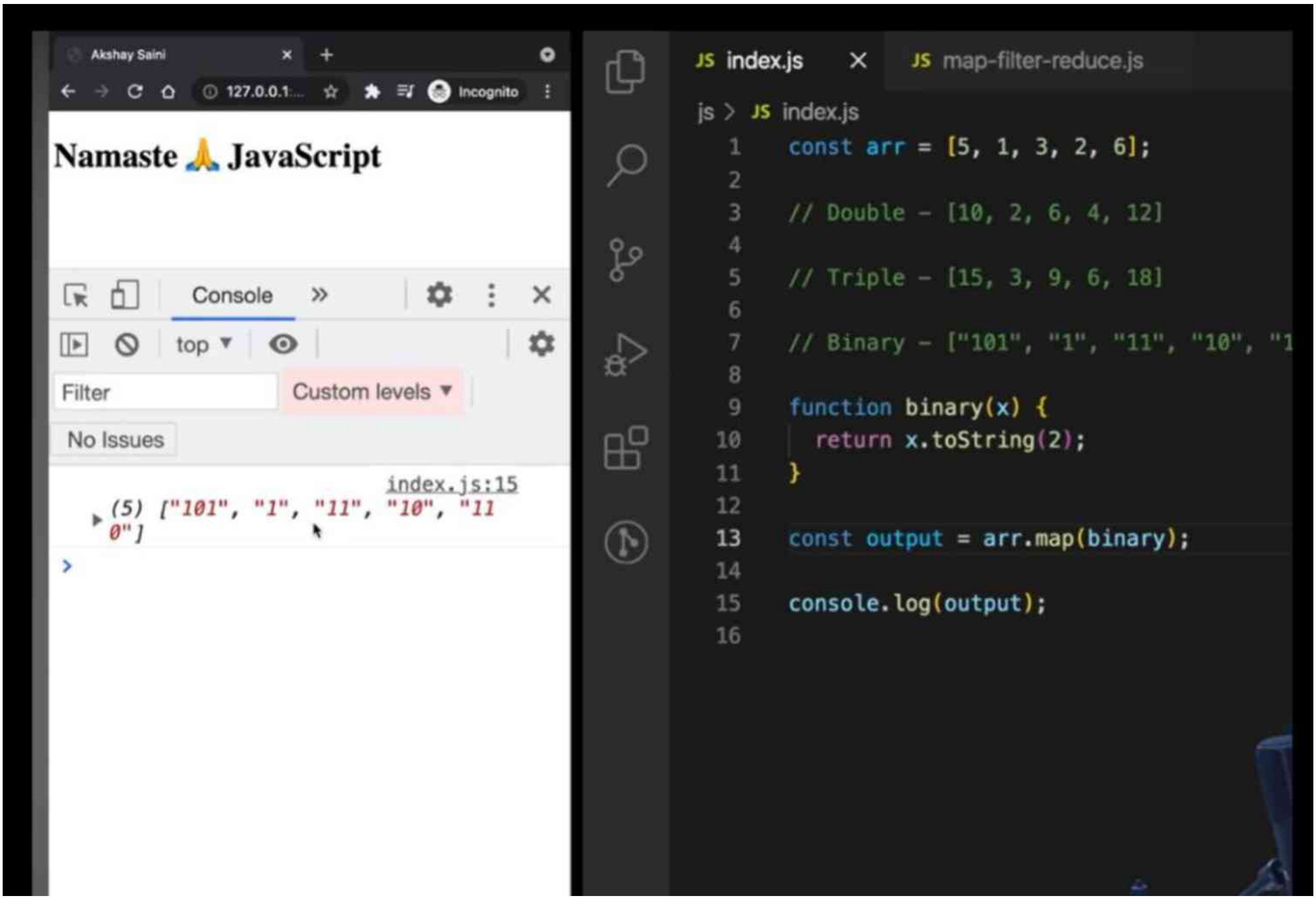
```
▶ (5) [10, 2, 6, 4, 12]   index.js:15
```

```
const arr = [5, 1, 3, 2, 6];

// Double - [10, 2, 6, 4, 12]

// Triple - [15, 3, 9, 6, 18]

// Binary - ["101", "1", "11", "10", "110"]

function double(x) {
  return x * 2;
}

const output = arr.map(double);

console.log(output);
```



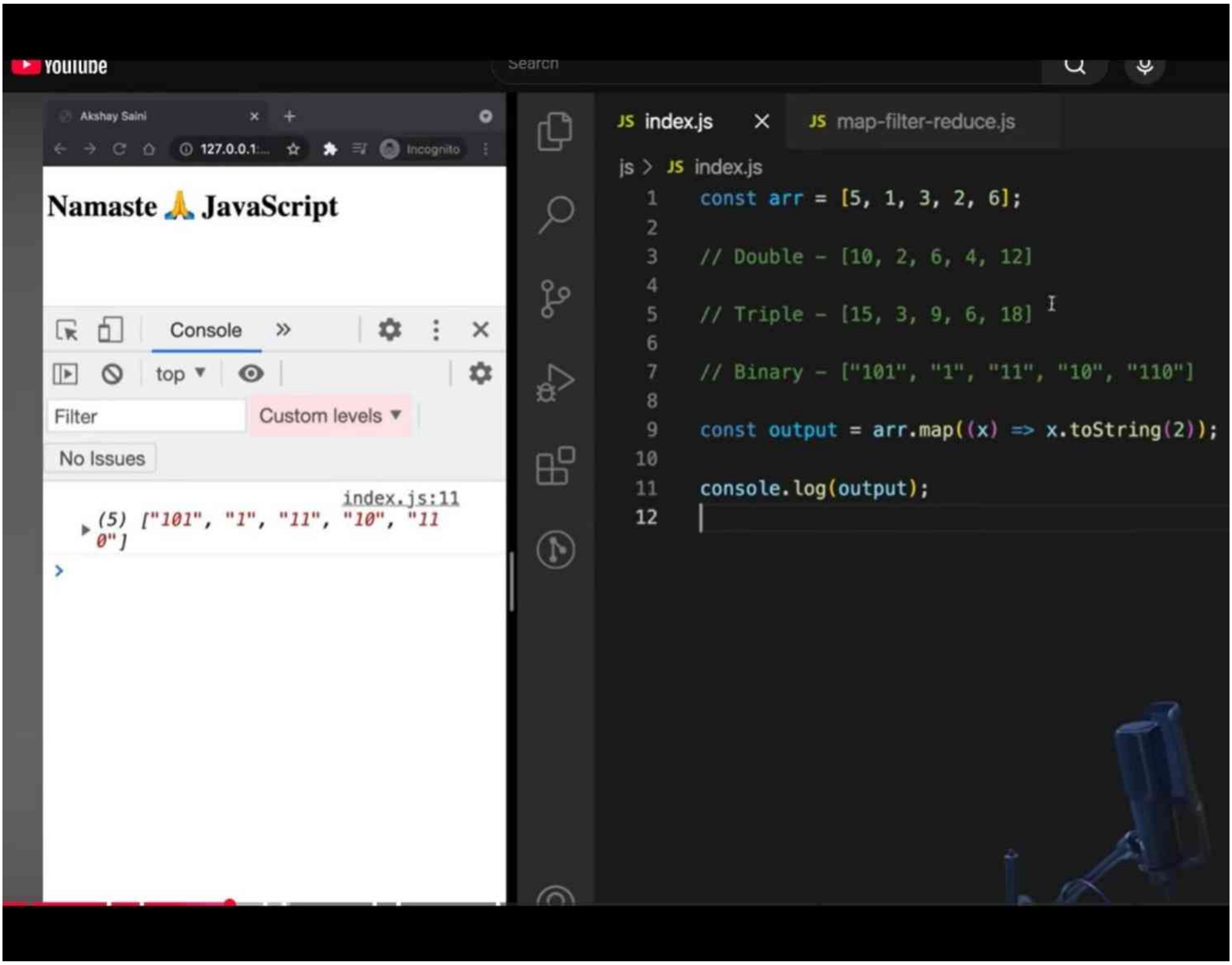
```
                                    index.js:19
▼(5) [15, 3, 9, 6, 18]
    0: 15
    1: 3
    2: 9
    3: 6
    4: 18
    length: 5
  ▶__proto__: Array(0)
>
```

```javascript
// Triple - [15, 3, 9, 6, 18]

// Binary - ["101", "1", "11", "10", "110"]

function double(x) {
  return x * 2;
}

function triple(x) {
  return x * 3;
}

const output = arr.map(triple);

console.log(output);
```



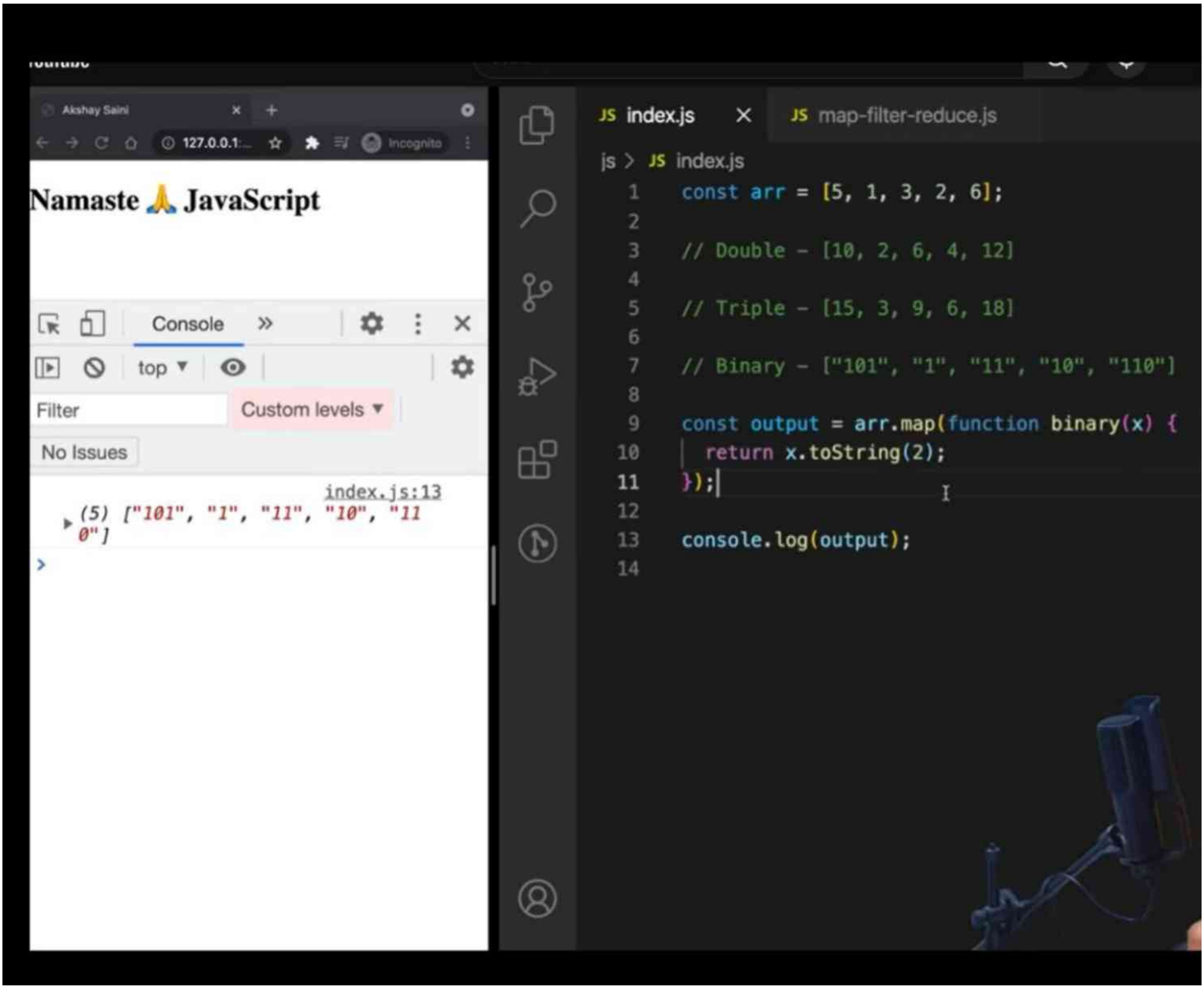
```
index.js:15
(5) ["101", "1", "11", "10", "11
0"]
```

```javascript
const arr = [5, 1, 3, 2, 6];

// Double - [10, 2, 6, 4, 12]

// Triple - [15, 3, 9, 6, 18]

// Binary - ["101", "1", "11", "10", "1

function binary(x) {
  return x.toString(2);
}

const output = arr.map(binary);

console.log(output);

```



```
index.js:11
▸ (5) ["101", "1", "11", "10", "11
0"]
>
```

```javascript
const arr = [5, 1, 3, 2, 6];

// Double - [10, 2, 6, 4, 12]

// Triple - [15, 3, 9, 6, 18]

// Binary - ["101", "1", "11", "10", "110"]

const output = arr.map((x) => x.toString(2));

console.log(output);

```



```
index.js:13
▸ (5) ["101", "1", "11", "10", "110"]
>
```

```javascript
const arr = [5, 1, 3, 2, 6];

// Double - [10, 2, 6, 4, 12]

// Triple - [15, 3, 9, 6, 18]

// Binary - ["101", "1", "11", "10", "110"]

const output = arr.map(function binary(x) {
  return x.toString(2);
});

console.log(output);
```