

JS - Day 2

→ datatypes

var : It's deprecated

↳ too many issues -- scope related..

→ let ... introduced..

→ is let hoisted? : Yes X

Yes + Explanation ✓

age=12;
let age=11;

age is defined
but can't accessed
earlier..

→ const -- introduced

→ Zyadat aur iska use..

→ const can't be changed

→ same behavior as let
in context of hoisting ..

↑
Temporal
dead
zone

{ declared variables
(let, const) exist
in memory, but
not accessible }

FUNCTIONS

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

Function call

parameters

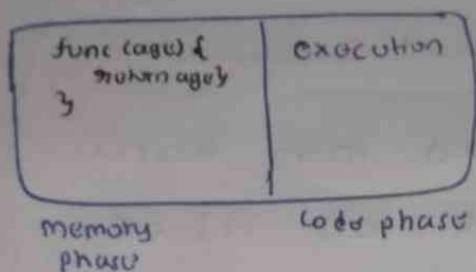
Syntax

```
function myfunc() {
    // code
}
myfunc();
```

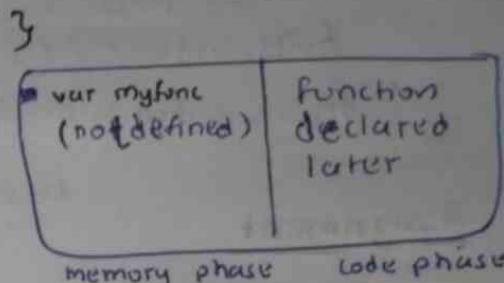
→ function can return anything - function, exp, variable
(anything → javascript legal)

→ a variable can hold a function.

```
function func (age) {  
    return age ≥ 18;  
}
```



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



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

- can access function before (undefined)

Arrow function

```
frame          parameter      return  
const isAllowedToVote = (age) => age ≥ 18
```

e.g.,
const isAllowedToOpenAccount = (age, minBalance) =>
 age ≥ 18 && minBalance ≥ 5000

NO DRAWING TODAY



MIND BLOWN!

#DATA STRUCTURES

* Array \Rightarrow Let fruits = ["apple", "mango", "banana"]

↳ methods--

• fruits.includes("mango") \rightarrow true

e.g., fruits.shift()

fruits.pop()

fruits.indexOf()

fruits.forEach(
 ↑
 function)

Assignment

etc...

yaad mat karo

code toh Al se
karwana hai -- Lmao

\rightarrow Implement Queue &
Stack --

using Array --

HIGH ORDER FUNCTION

```
function jollyFunction (udhaarkufunction) {
    return udhaarkufunction () + 10;
}
```

Using for each--

fruits.forEach(element \Rightarrow console.log(element))

arrow function

(no fname)

map → internally new array
Creates new array

```
const nums = [1, 2, 3, 4, 5, 6];
const result = nums.map ((x,y,z) => x+y+z*2);
    
$$\text{Ex} \rightarrow [2, 4, 6, 8, 10, 12]$$

```

map → github.com/realSUDO

Skip

x → @sudo_core

discord → @ sudo.dis

LOL
PROMOTION

for each

→ only iterates

map

→ creates new arr



4

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 functions

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 argument

Call
back function

3

* Function which is
Passed into higher order
function is known as
callback function

function
map

maps giving some logic

small implementation

=> Map → is just create new
array and iterate over
all those of all elements
in the range ~~selection~~
return the output

```
js > JS index.js
1  const radius = [3, 1, 2, 4];
2
3  const area = function (radius) {
4    return Math.PI * radius * radius;
5  };
6
7  const circumference = function (radius) {
8    return 2 * Math.PI * radius;
9  };
10
11 const diameter = function (radius) {
12   return 2 * radius;
13 };
14
15 const calculate = function (radius, logic) {
16   const output = [];
17   for (let i = 0; i < radius.length; i++) {
18     output.push(logic(radius[i]));
19   }
20   return output;
21 };
22
23 console.log(calculate(radius, area));
24 console.log(calculate(radius, circumference));
25 console.log(calculate(radius, diameter));
26
```

The screenshot shows a browser's developer tools interface with the 'Console' tab selected. The left pane displays two log entries from 'index.js':

```
▶ (4) [28.274333882308138, 3.14159265358979  
 3, 12.566370614359172, 50.26548245743669]  
▶ (4) [28.274333882308138, 3.14159265358979  
 3, 12.566370614359172, 50.26548245743669]
```

The right pane shows a portion of a JavaScript file with line numbers:

```
3 const area = function (radius) {  
4   return Math.PI * radius * radius;  
5 };  
6  
7 const circumference = function (radius) {  
8   return 2 * Math.PI * radius;  
9 };  
10  
11 const diameter = function (radius) {  
12   return 2 * radius;  
13 };  
14  
15 Array.prototype.calculate = function (log  
16   const output = [];  
17   for (let i = 0; i < this.length; i++) {  
18     output.push(logic(this[i]));  
19   }  
20   return output;  
21 };  
22  
23 console.log(radius.map(area));  
24  
25 console.log(radius.calculate(area));  
26 // console.log(calculate(radius, circumference));  
27 // console.log(calculate(radius, diameter));
```

*~~Prototype~~

When you ~~put~~

Put something onto

~~Prototype~~

It appears in all arrays

Prototype

When you put

Something onto Prototype

It's ~~is~~ appear in all the

array

Map() function

Array Transformation

Transform ^{what} array
using map() function

→ Transform each and every value of this array
and get new array of it

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

Doubt : [1, 2, 6, 9, 12]

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

Brown : ["101", "1", "11", "12", "110"]

A screenshot of a browser developer tools interface, likely from Google Chrome, showing the JavaScript console tab. The title bar shows "Akshay Saini" and "127.0.0.1...". The console tab is selected, and the output area displays the text "Namaste ⚒ JavaScript". Below this, the "Console" tab is active, and the "Custom levels" dropdown is open, with a red box highlighting it. The "No Issues" section shows a message "(5) [10, 2, 6, 4, 12] index.js:15". To the right of the console, the code editor shows two files: "index.js" and "map-filter-reduce.js". The "index.js" file contains the following code:

```
1 const arr = [5, 1, 3, 2, 6];
2
3 // Double - [10, 2, 6, 4, 12]
4
5 // Triple - [15, 3, 9, 6, 18]
6
7 // Binary - ["101", "1", "11", "10", "110"]
8
9 function double(x) {
10   return x * 2;
11 }
12
13 const output = arr.map(double);
14
15 console.log(output);
16
```

A screenshot of a developer's workspace. On the left is a terminal window titled "Console" with the tab "Custom levels" highlighted. It shows the output of a command: "(5) [15, 3, 9, 6, 18]" followed by its elements: 0: 15, 1: 3, 2: 9, 3: 6, 4: 18, length: 5, and __proto__: Array(0). On the right is a code editor window displaying the following JavaScript code:

```
4 // Triple - [15, 3, 9, 6, 18]
5 // Binary - ["101", "1", "11", "10", "110"]
6
7
8
9 function double(x) {
10   return x * 2;
11 }
12
13 function triple(x) {
14   return x * 3;
15 }
16
17 const output = arr.map(triple);
18
19 console.log(output);
20
```

A screenshot of a browser window showing developer tools. The top bar shows the URL as 127.0.0.1. The left side has a 'Console' tab selected, displaying the output of the JavaScript code. The right side shows the source code in a file named index.js.

Console Output:

```
(5) ["101", "1", "11", "10", "11"]
```

Source Code (index.js):

```
js > JS index.js
1 const arr = [5, 1, 3, 2, 6];
2
3 // Double - [10, 2, 6, 4, 12]
4
5 // Triple - [15, 3, 9, 6, 18]
6
7 // Binary - ["101", "1", "11", "10", "11"]
8
9 function binary(x) {
10   return x.toString(2);
11 }
12
13 const output = arr.map(binary);
14
15 console.log(output);
16
```

The screenshot shows a browser window with the URL `127.0.0.1...`. The developer tools are open, specifically the Console tab. The console output is as follows:

```
(5) ["101", "1", "11", "10", "11  
0"]
```

On the right side of the developer tools, there is a code editor window titled `index.js` with the following content:

```
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);
```

A screenshot of a developer's workspace. On the left, a browser window displays the text "Namaste 🌈 JavaScript". Below it is a terminal window showing the command "index.js:13". To the right is a code editor with two tabs: "index.js" and "map-filter-reduce.js". The "index.js" tab contains the following code:

```
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);
```