

## \* Array method

(i) for each

arr. forEach (some fun<sup>n</sup> definition or name)

let arr = [1, 2, 3, 4, 5];

let print = function (el) {  
 console.log(el);

};

arr. forEach (print);

(OR)

arr. forEach (function (el) {

console.log(el);

});

(OR)

let arr = [

{

marks : 95,

},

{

marks : 45,

},

];

arr. forEach (student) => {

console.log(student.marks);

});

// or we  
know fun<sup>n</sup>



## • Map & Filter (store value in new array)

num = [1, 2, 3, 4]

let double = num.map((el) => {  
return el \* 2

});

## • Filter

let nums = [2, 4, 1, 5, 6, 2, 7, 8, 9];

let even = nums.filter((num) => {  
return num % 2 == 0;

});

// save even no. store to array even

console.log(even);

→ Return true if every element of array gives true  
else returns false.

## • Every & or. every (some fun<sup>n</sup> definition or name)

(logical and)

[1, 2, 3, 4].every((el) => {  
return el % 2 == 0;});

## • Some & logical or

→ if any one is true then

{ } gives wrong answer.

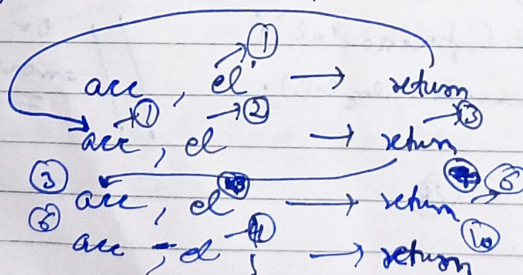
## • Reduce &

→ Reduces array to a single value (it can be any value)

→ arr.reduce (reducer fun<sup>n</sup> with 2 variable for accumulator, default)  
logic of reducing

→ [1, 2, 3, 4].reduce((acc, el) => {  
return acc + el;});

(10)



We cannot  
anything to  
place of  
acc & el

also on (multiple calls)

(0, 1) → 1

(1, 2) → 3

(3, 3) → 6

(6, 4) → 10

final val

let num = [1, 2, 3];

let finalVal = num.reduce((acc, el) => {

return acc + el

});

console.log(finalVal);



### • Maximum in array :-

```
let max = arr.reduce((max, el) => {  
  if (max < el) {  
    return el;  
  }  
  return max;  
}, arr[0]);  
console.log(max);
```

### • Default parameters :-

```
function sum(a, b=2) {  
  return(a+b);  
}
```

// we not do b=2  
then NaN

sum(1)

### • Spread :-

like array, string, object etc  
element ko array

→ Spread an iterable into multiple values

→ function fun(...arr) {

}

→ let arr = [1, 2, 3, 4, 0, 5, 5, 7, 1, 8]

Math.min(...arr);

→ arr ki saari value individually  
print !

console.log(...arr);

// saare individually print.

### • spread (object literals) :-

const data = {

email: "inew",

password: "abcd",

};

const dataCopy = {...data} // full above data.  
// for address;

const dataCopy = {...data, id: 123}

### • Rest :-

Allows a fun<sup>n</sup> to take an indefinite no. of  
arguments & bundle them in array.

1 2 3 4 5 6 7 8 9 10



function `sum (...args)` {  
 return `args.reduce((add, el) => add + el)`;  
}

31x12 21x1 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100

to args work this karta hai  
 reduce array ka method

`sum(12, 13, 14, 15, ...)`

• `fun sum(msg, ...args) {  
 console.log(msg);  
}`

• Agr aur koi parameter hai to ...args  
 ki value aayegi.

## \* Destructuring

• Store value of array into multiple variables

• `let names = ["Tony", "Bruce", "Mory"]`

`let winner = names[0];`

`let runner up = names[1];`

• It does not work

• It is destructuring

`let [winner, runnerup] = names;`

`console.log(winner) // 'Tony'`

// yehi hai jo naam log ki jata hai.

`let [winner, runnerup, ...others] = names`

## \* for objects

`const student = {`

`};`

`let {username, password} = student;`

`console.log(username) // 'aman'`  
`// (city) // undefined`

`let {username: user, password} = student;`

`console.log(username) // undefined.`

`console.log(user) // 'aman'`

`// (city) // 'Mumbai'`

// yehi student ki city hai aur user preference isyahi.