* Array method

(i) forEach()

arr.forEach(some fun^n 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 we arrow fun^n

OR

```
let arr=[
{
  marks :95,
},
{
  marks :45,
},
];
arr.forEach( (student)=>{
        console.log (student.marks);
});
```

• **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;
});
//saare even no. store ho jayenge even me
console.log(ans).
```

callback → true → el ✓ (element of new array me add) ; false → el ✗

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

• **Every**: arr.every(some fun^n definition or name.)
(logical and)

[1,2,3,4].every ((el) => (el%2 == 0)); → true or false

→ yhi aane chahiye yha, {} gives wrong answer.

• **Some**: logical or

• **Reduce**:

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

→ arr.reduce (reducer fun^n with 2 variable for (accumulator, element));
↓ logic of reducing

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

(10)

acc, el (1) → return
acc (1), el (2) → return (3)
(3) acc, el (3) → return (6)
(6) acc, el (4) → return (10)
& so on (till it ends)

We can use anything in place of acc & el.

(0, 1) => 1
(1, 2) => 3 → jo so Low Ne ahrat
(3, 3) => 6
(6, 4) => 10
→ final val

```
let num = [1,2,3];
let finalVal = num.reduce((res,el)=>{
    console.log(res)
    return res + el
});
console.log(finalVal);
```

- **Maximum in array:**

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

- **Default parameters:**

```
function sum(a, b=2){
    return(a+b);
}
sum(1)
```

//if we not do b=2 then NAN

- **Spread:**

→ Expread an iterable into multiple values (like array, string jo ek ek element ko acces)

→ function fun(...arr){
}

```
→ Let arr = [1, 2, 3, 4, 0, 5, 5, 7, 1, 8]
Math.min(...arr);
```

↳ arr ki saari value individually लोटेगी!

```
console.log(...arr);
//saare individually print.
```

- **spread (object literals):**

```
const data = {
    email : "inam",
    password : "akd",
};
const dataCopy = {...data} // full above data.
// for adding:
const dataCopy = {...data, id:123}
```

- **Rest:**

Allows a funⁿ to take an indefinite no. of arguments & bundle them in array.

```
function sum(...args){
    return args.reduce((add, el) => add + el);
}
sum(12, 13, 14, 15 --).
```

→ उधर यहाँ पर ये नहीं arguments to args work nhi karta so reduce array ka method.

- fun^n sum(msg, ... args){
  console.log(msg)

- Agr aur koi parameter हो तो change hai to ...args हे पहले रखना.

* Destructuring :-

- storing value of array into multiple variables
- let names = ["tony", "bruce", "mary"]
  let winner = names[0];
  let runner up = names[1];

  { इस chiz ko short karne को कहते Destructuring

```
let [winner, runnerup] = names;
console.log(winner)    // 'tony'
```

// अगर baki ke naam dalna हो तो .

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

* for objects

```
const student = {

};
let {username, password, city} = student;
console.log(username)    // 'Aman'
  "     "    (city)      // undefined
```

OR

```
let {username : user, password, city = "mumbai"} = student;
console.log(username)  // undefined.
console.log(user)      // 'Aman'
  "     "   (city)     // 'mumbai'
```

// if student के अंदर भी city hoti then usko preference jyada.