ARRAY REDUCE

Array.prototype.reduce()

The reduce() method executes a user-supplied "reducer" callback function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result of running the reducer across all elements of the array is a single value.

The first time that the callback is run there is no "return value of the previous calculation". If supplied, an initial value may be used in its place. Otherwise array element 0 is used as the initial value and iteration starts from the next element (index 1 instead of index 0).

Syntax

```
array.reduce(function(total, currentValue, currentIndex, arr), initialValue)
```

Parameters

Parameter	Description
function()	Required. A function to be run for each element in the array.

Reducer function parameters:

total	Required. The initialValue, or the previously returned value of the function.
currentValue	Required. The value of the current element.
currentIndex	Optional. The index of the current element.
arr	Optional. The array the current element belongs to.
initialValue	Optional. A value to be passed to the function as the initial value.

Return Value

The accumulated result from the last call of the callback function.

```
1
2   let arr = [10, 20, 30, 40, 50];
3
4   // pv -> previous value
5   // cv -> current value
6   // ci -> current index
7   // oarr -> original array
8
9   let sum = arr.reduce(function(pv, cv, ci, oarr){
10      console.log(pv + " - " + cv + " - " + ci)
11      return pv + cv;
12  });
13
14  // 10, 20, 1
15  // 30, 30, 2
16  // 60, 40, 3
17  // 100, 50, 4
18  // 150
19
20  console.log(sum);
21
22
23  let sum2 = arr.reduce(function(pv, cv, ci, oarr){
24      console.log(pv + " - " + cv + " - " + ci)
```

```
19
20  console.log(sum);
21
22
23  let sum2 = arr.reduce(function(pv, cv, ci, oarr){
24      console.log(pv + " - " + cv + " - " + ci)
25      return pv + cv;
26  }, 5); // passing intial value as 5
27
28  // 5, 10, 0
29  // 15, 20, 1
30  // 35, 30, 2
31  // 65, 40, 3
32  // 105, 50, 4
33  // 155
34
35  console.log(sum2);
36
37  let sum3 = arr.reduce((pv, cv, ci) => pv + cv, 5);
38  console.log(sum3);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE                    zsh - Lecture_37

➜  Lecture_37 git:(main) ✗ node 1_ArraysReduceDemo.js
10 - 20 - 1
30 - 30 - 2
60 - 40 - 3
100 - 50 - 4
150
5 - 10 - 0
15 - 20 - 1
35 - 30 - 2
65 - 40 - 3
105 - 50 - 4
155
155
```

CUSTOM REDUCE

```
1
2   // cb -> callback
3   // iv -> initial value
4
5   Array.prototype.myReduce = function(cb, iv){
6
7       let oarr = this;
8       let pv;
9
10      if(iv == undefined){
11
12          pv = oarr[0];
13          for(let i = 1; i < oarr.length; i++){
14              let cv = oarr[i];
15              pv = cb(pv, cv, i, oarr);
16          }
17
18      }else{
19
20          pv = iv;
21          for(let i = 0; i < oarr.length; i++){
22              let cv = oarr[i];
23              pv = cb(pv, cv, i, oarr);
24          }
25
26      }
27
28      return pv;
29  }
30
31  let arr = [10, 20, 30, 40, 50];
32
33  // pv -> previous value
34  // cv -> current value
35  // ci -> current index
36  // oarr -> original array
```

```
38  let sum = arr.myReduce(function(pv, cv, ci, oarr){
39      console.log(pv + " - " + cv + " - " + ci)
40      return pv + cv;
41  });
42
43  // 10, 20, 1
44  // 30, 30, 2
45  // 60, 40, 3
46  // 100, 50, 4
47  // 150
48
49  console.log(sum);
50
51
52  let sum2 = arr.myReduce(function(pv, cv, ci, oarr){
53      console.log(pv + " - " + cv + " - " + ci)
54      return pv + cv;
55  }, 5); // passing intial value as 5
56
57  // 5, 10, 0
58  // 15, 20, 1
59  // 35, 30, 2
60  // 65, 40, 3
61  // 105, 50, 4
62  // 155
63
64  console.log(sum2);
65
66  let sum3 = arr.myReduce((pv, cv, ci) => pv + cv, 5);
67  console.log(sum3);
```

Ques → count all primes using reduce

```
// count all primes using reduce

let arr = [51, 23, 37, 44, 73, 82, 97, 45];
let cp = arr.reduce(function(pv, cv, ci, oarr){

    let flag = true;

    console.log(pv + " - " + cv);

    for(let div = 2; div * div <= cv; div++){
        if(cv % div == 0){
            flag = false;
            break;
        }
    }
    if(flag == true){
        return pv + 1;
    } else {
        return pv;
    }

}, 0);


console.log(cp);

// 0, 51
// 0, 23
// 1, 37
// 2, 44
// 2, 73
// 3, 82
// 3, 97
// 4, 45
// 4

```

```
→  Lecture_37 git:(main) ✗ node 3_ArraysReduceQues1.js
0 - 51
0 - 23
1 - 37
2 - 44
2 - 73
3 - 82
3 - 97
4 - 45
4
```

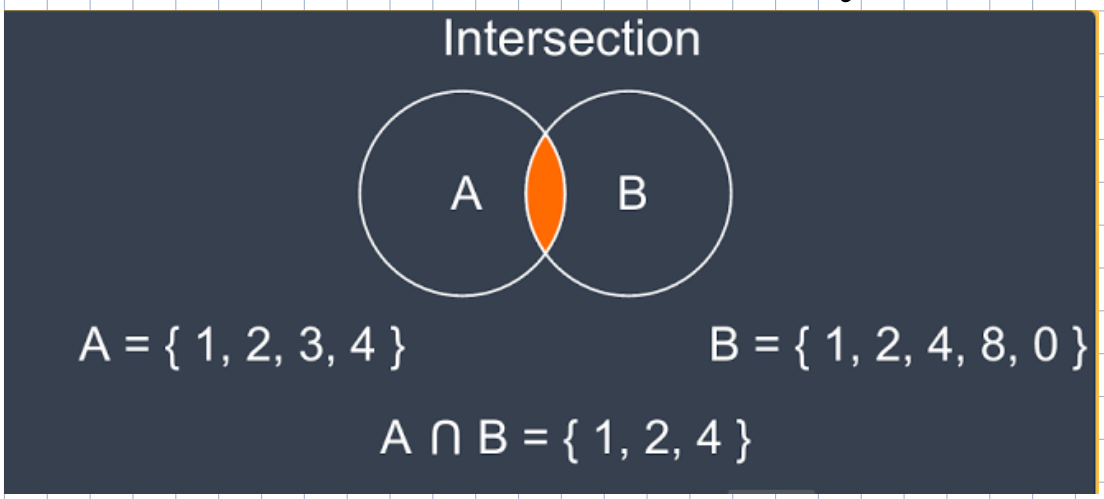
Ques → flatten 2D array using reduce

```
1
2   // flatten 2D ARRAY
3   // convert 2D Array into 1D Array
4
5   let arr2d = [
6       [10, 20, 30],
7       [22, 17],
8       [54, 58, 92, 34],
9       [61, 31, 55, 92],
10      [17]
11  ]
12
13  let arr = arr2d.reduce(function(pv, cv, ci, oarr){
14      let narr_joined = pv.concat(cv);
15      console.log("[" + pv + "]" + "-" + "[" + cv + "]" + "-" + ci);
16      return narr_joined;
17  }, [])
18
19  console.log(arr);
20  // [], [10,20,30] - 0
21  // [10,20,30], [22,17] - 1
22  // [10,20,30,22,17], [54,58,92,34] - 2
23  // [10,20,30,22,17,54,58,92,34], [61,31,55,92] - 3
24  // [10,20,30,22,17,54,58,92,34,61,31,55,92], [17] - 4
25  // [10, 20, 30, 22, 17, 54,58, 92, 34, 61, 31, 55, 92, 17]
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_37 git:(main) ✗ node 4_ArraysReduceQues2.js
[]-[10,20,30]-0
[10,20,30]-[22,17]-1
[10,20,30,22,17]-[54,58,92,34]-2
[10,20,30,22,17,54,58,92,34]-[61,31,55,92]-3
[10,20,30,22,17,54,58,92,34,61,31,55,92]-[17]-4
[
  10, 20, 30, 22, 17, 54,
  58, 92, 34, 61, 31, 55,
  92, 17
]
```

Ques → Intersection of arrays




```
// intersection of two arrays

let a1 = [10, 30, 50, 70, 90];
let a2 = [63, 34, 50, 90, 80, 10, 60];

// 10, 50, 90
console.log(a2.includes(80));
console.log(a2.includes(88));

let inter1 = a1.filter(v => a2.includes(v));
console.log(inter1);

let arr2d = [
    [10, 50, 70, 80, 90, 100, 30, 60],
    [11, 50, 75, 85, 90, 100, 34, 60], // 50, 90, 100, 60
    [10, 51, 70, 80, 90, 100, 30, 60], // 90, 100, 60
    [11, 50, 75, 85, 92, 100, 34, 60], // 100, 60
    [10, 50, 70, 80, 90, 100, 30, 60], // 100, 60
];

let inter2 = arr2d.reduce(function(pv, cv, ci, oarr){

    console.log(pv + " #### " + cv);

    let inter = pv.filter(v => cv.includes(v));
    return inter;

});

console.log(inter2);
// [10, 50, 70, 80, 90, 100, 30, 60] [11, 50, 75, 85, 90, 100, 34, 60] => [50, 90, 100, 60]
// [50, 90, 100, 60] [10, 51, 70, 80, 90, 100, 30, 60] => [90, 100, 60]
// [90, 100, 60] [11, 50, 75, 85, 92, 100, 34, 60] => [100, 60]
// [100, 60] [10, 50, 70, 80, 90, 100, 30, 61] => [100]

// [100]
```

Output

```
→  Lecture_37 git:(main) ✗ node 6_ArraysReduceQues4.js
true
false
[ 10, 50, 90 ]
10,50,70,80,90,100,30,60 #### 11,50,75,85,90,100,34,60
50,90,100,60 #### 10,51,70,80,90,100,30,60
90,100,60 #### 11,50,75,85,92,100,34,60
100,60 #### 10,50,70,80,90,100,30,60
[ 100, 60 ]
```

Ques → Union of arrays

```
1
2   //union of two array
3
4   let arr1 = [10, 50, 70, 80, 90, 100, 30, 60];
5   let arr2 = [11, 50, 75, 85, 90, 100, 34, 60];
6
7   // a2ma1 -> a2 minus a1 -> a2 - a1
8
9   let a2ma1 = arr2.filter(v => arr1.includes(v) == false); // elements in a2 which are not in a1
10
11  console.log(a2ma1);
12
13  // arr1 U a2ma1
14
15  let union = arr1.concat(a2ma1);
16  console.log(union);
17
18  // union of arrays
19
20  let arr2d = [
21      [10, 50, 70, 80, 90, 100, 30, 60],
22      [11, 50, 75, 85, 90, 100, 34, 60], // [10, 50, 70, 80, 90, 100, 30, 60, 11, 75, 85, 34]
23      [10, 51, 70, 80, 90, 100, 30, 60], // [10, 50, 70, 80, 90, 100, 30, 60, 11, 75, 85, 34, 51]
24      [11, 50, 75, 85, 92, 100, 34, 60], // [10, 50, 70, 80, 90, 100, 30, 60, 11, 75, 85, 34, 51, 92]
25      [10, 50, 70, 80, 90, 100, 30, 60], // [10, 50, 70, 80, 90, 100, 30, 60, 11, 75, 85, 34, 51, 92]
26  ];
27
28  let union1 = arr2d.reduce(function(pv, cv, ci, oarr){
29      let cvmpv = cv.filter(v => pv.includes(v) == false);
30      let union = pv.concat(cvmpv);
31      return union;
32  })
33  console.log(union1);
34
35  // [10, 50, 70, 80, 90, 100, 30, 60, 11, 75, 85, 34, 51, 92] (union of arrays)
```

```
→  Lecture_37 git:(main) ✗ node 5_ArraysReduceQues3.js
[ 11, 75, 85, 34 ]
[
  10, 50, 70, 80, 90,
  100, 30, 60, 11, 75,
  85, 34
]
[
  10, 50, 70, 80, 90, 100,
  30, 60, 11, 75, 85,  34,
  51, 92
]
```

a2 - a1

arr1 U (a2 - a1)

union1

Ques → Compound function

```
//compound functions
function f(x){
    return x * x;
}

function g(x){
    return x + 10;
}

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

let farr = [f, g, h]; // [h, g, f]
let x = 10;

let cv = farr.reverse().reduce(function(pv, cv){
    return cv(pv);
}, x);

console.log(cv);

//Ques1

// f(g(h(x))) = f(g(2x)) = f(2x + 10) = 4x^2 + 100 + 40x = 900

// 10, f
// f(10), g
// g(f(10)), h
// h(g(f(10)))


// 10, h
// h(10), g
// g(h(10)), f
// f(g(h(10)))
```

Ques :- Sum of squares of ages of all valid candidates.

```
let arr = [
    {name: "A", age: 14, gender: "M"},
    {name: "B", age: 34, gender: "M"},
    {name: "C", age: 24, gender: "F"},
    {name: "D", age: 44, gender: "F"},
    {name: "E", age: 44, gender: "M"},
    {name: "I", age: 28, gender: "F"},
    {name: "G", age: 36, gender: "M"},
    {name: "H", age: 47, gender: "F"}
]

//ques3: sum of squares of ages of all valid candidates (females age between 20 and 30)

let res1 = arr.filter(v => v.gender == 'F' && v.age >= 20 && v.age <= 30);

let res2 = res1.map(v => v.age * v.age);

let sum = res2.reduce(function(pv, cv, ci, oarr){
    return pv + cv;
});

console.log(sum);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_37 git:(main) ✗ node 10_ArraysFilterMapReduceQues.js
1360
→  Lecture_37 git:(main) ✗
```

ARRAY REDUCE RIGHT

Array.prototype.reduceRight()

The reduceRight() method applies a function against an accumulator and each value of the array (from right-to-left) to reduce it to a single value.

Syntax

```
array.reduceRight(function(total, currentValue, currentIndex, arr), initialValue)
```

Parameters

Parameter	Description
function()	Required. A function to be run for each element in the array.

Reducer function parameters:

total	Required. The initialValue, or the previously returned value of the function.
currentValue	Required. The value of the current element.
currentIndex	Optional. The index of the current element.
arr	Optional. The array the element belongs to.
initialValue	Optional. A value to be passed to the function as the initial value.

Return Value

The accumulated result from the last call of the callback function

```js
const numbers = [1, 2, 3, 4, 5, 6];

function sum_reducer(accumulator, currentValue, currentIndex) {

  console.log(accumulator + "  " + currentValue + "  " + currentIndex);
  return accumulator + currentValue;
}

let sum = numbers.reduceRight(sum_reducer);
console.log(sum); // 21

// using arrow function
let summation = numbers.reduceRight(
  (accumulator, currentValue) => accumulator + currentValue
);
console.log(summation); // 21
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
➜  Lecture_37 git:(main) ✗ node 14_ArraysReduceRightDemo.js
6  5  4
11  4  3
15  3  2
18  2  1
20  1  0
21
21
➜  Lecture_37 git:(main) ✗
```

ARRAY SEARCHING

Easy searching

Array.prototype.includes()

1) includes()

The includes() method determines whether an array includes a certain value among its enteries, returning true or false as appropriate.

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

console.log(arr.includes(2));
// expected output: true
```

Array.prototype.indexOf()

2) indexOf()

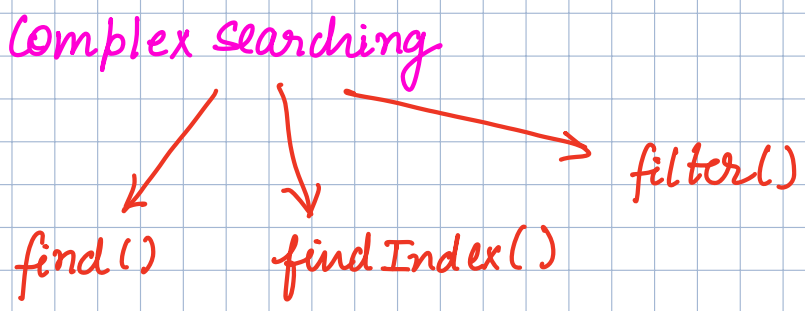
The indexOf() method returns the first index at which a given element can be found in the array, or -1 if it is not present.

```
let arr = ['a', 'b', 'c', 'd', 'b']

console.log(arr.indexOf('b'));
// expected output: 1

// start from index 2
console.log(arr.indexOf('b', 2));
// expected output: 4
```

3) lastIndexOf()

The lastIndexOf() method returns the last index at which a given element can be found in the array, or -1 if it is not present. The array is search backwards, starting at fromIndex.

```
let arr = ['a', 'b', 'c', 'd', 'b']

console.log(arr.indexOf('b'));
// expected output: 4
```




```
1   // Easy searching = indexOf, lastIndexOf, includes
2   // Complex searching = find, findIndex
3   // Complex searching and all filtered values = filter
4
5   let arr = [10, 20, 30, 40, 50, 60, 60, 50, 40, 30, 20, 10];
6
7   let ioRes = arr.indexOf(30); // firstIndex or -1
8   let lioRes = arr.lastIndexOf(30); // last index or -1
9   let iRes = arr.includes(30); // true or false
10
11  // give me the first value above 50
12  let fRes = arr.find(function(v, i, oarr){
13      return v > 50;
14  })
15
16  // give me index of first value above 50
17  let fiRes = arr.findIndex(function(v, i, oarr){
18      return v > 50;
19  })
20
21  // give me all values above 50
22  let fltrRes = arr.filter(function(v, i, oarr){
23      return v > 50;
24  })
25
26  console.log(ioRes);
27  console.log(lioRes);
28  console.log(iRes);
29  console.log(fRes);
30  console.log(fiRes);
31  console.log(fltrRes);
```

Output

```
→  Lecture_37 git:(main) ✗ node 11_ArraysSearching.js
2
9
true
60
5
[ 60, 60 ]
→  Lecture_37 git:(main) ✗
```

ARRAY SORT & REVERSE

Ques :- sort and reverse these arrays.

```
1
2   let sarr = ["hello", "bello", "bye", "there", "pep", "nados"];
3   let narr = [21, 54, 12, 33, 98, 200, 76, 100, 11, 291, 34];
4
5   // sort and reverse
6   sarr.sort();
7   console.log(sarr);
8
9   sarr.reverse();
10  console.log(sarr);
11  For sorting the numerical array
12  // narr.sort(); // does an alphabetical sort
13  // console.log(narr);
14  To sort array in numerical order
15  narr.sort((a, b) => a - b); // numerical sort
16  console.log(narr);
17
18  narr.reverse();
19  console.log(narr);
```

sort an array alphabetically

reverses the order of the elements in an array.

sorting number in alphabetical order is called Lexicographical sort.

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

→  Lecture_37 git:(main) ✗ node 12_ArrayReverseandSort.js
[ 'bello', 'bye', 'hello', 'nados', 'pep', 'there' ]
[ 'there', 'pep', 'nados', 'hello', 'bye', 'bello' ]
[
   11, 12, 21,  33,  34,
   54, 76, 98, 100, 200,
  291
]
[
  291, 200, 100, 98, 76,
   54,  34,  33, 21, 12,
   11
]
→  Lecture_37 git:(main) ✗
```

ARRAY CONCAT AND JOIN

```
let str = "My name is Hemakshi Pandey. I am a software developer. I believe in learning by doing. I need courage and patience.";

"Pandey Hemakshi is name My. developer software a am I. doing by learning in believe I. patience and courage need I.";

// without for loops you have to do it.
let res1 = str.split(".");
console.log(res1);

let res2 = res1.filter(s => s.length > 0);
console.log(res2);

let res3 = res2.map(s => s.trim());
console.log(res3);

let res4 = res3.map(s => s.split(" "));
console.log(res4);

res4.map(a => a.reverse()); // it is more relevant to use foreach
console.log(res4);

let res5 = res4.map(a => a.join(" "));
console.log(res5);

let res6 = res5.map(s => s + ".");
console.log(res6);

let res7 = res6.reduce(function(pv, cv){
    return pv + " " + cv;
})

console.log(res7);
// let res = str.split(".").filter(s => s.length > 0).map(s => s.trim().split(" ").reverse().join(" ") + ".").reduce(function(pv, cv){
//     return pv + " " + cv;
// });

// console.log(res);
```

output

```
→  Lecture_37 git:(main) ✗ node 13_ArraysConcatandJoin.js
[
  'My name is Hemakshi Pandey',
  ' I am a software developer',
  ' I believe in learning by doing',
  ' I need courage and patience',
  ''
]
[
  'My name is Hemakshi Pandey',
  ' I am a software developer',
  ' I believe in learning by doing',
  ' I need courage and patience'
]
[
  'My name is Hemakshi Pandey',
  'I am a software developer',
  'I believe in learning by doing',
  'I need courage and patience'
]
[
  [ 'My', 'name', 'is', 'Hemakshi', 'Pandey' ],
  [ 'I', 'am', 'a', 'software', 'developer' ],
  [ 'I', 'believe', 'in', 'learning', 'by', 'doing' ],
  [ 'I', 'need', 'courage', 'and', 'patience' ]
]
[
  [ 'Pandey', 'Hemakshi', 'is', 'name', 'My' ],
  [ 'developer', 'software', 'a', 'am', 'I' ],
  [ 'doing', 'by', 'learning', 'in', 'believe', 'I' ],
  [ 'patience', 'and', 'courage', 'need', 'I' ]
]
[
  'Pandey Hemakshi is name My',
  'developer software a am I',
  'doing by learning in believe I',
  'patience and courage need I'
]
[
  'Pandey Hemakshi is name My.',
  'developer software a am I.',
  'doing by learning in believe I.',
  'patience and courage need I.'
]
Pandey Hemakshi is name My. developer software a am I. doing by learning in believe I. patience and courage need I.
→  Lecture_37 git:(main) ✗
```