ARRAY FILTER

```
3  // Filter is itself a function
4  // Filter takes as input a callback function
5  // The callback function takes 3 parameter (v, i, arr)
6  // Filter will call the callback multiple times (once for each value)
7  // For each run of callback, filter will pass v, i and original array to callback
8  // Callback will process the value and index and return a single boolean value
9  // Single value returned by each run of callback will be used by filter
10 // Whenever a true is received by filter (returned by callback) then filter adds the v to a new array
11 // Filter returns that new array
12 // Length of returned array is equal to number of trues returned by callback
13
14
15 let arr = [2, 5, 9, 8, 15, 11, 6];
16 let odd_arr = arr.filter(function(v, i, oarr){
17     console.log(v + " @ " + i + " [" + oarr + "]");
18     if(v % 2 == 1){
19         return true;
20     } else {
21         return false;
22     }
23 });
24
25 console.log(odd_arr);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

➜  Lecture_35 git:(main) ✗ node 1_ArraysFilter.js
2 @ 0 [2,5,9,8,15,11,6]
5 @ 1 [2,5,9,8,15,11,6]
9 @ 2 [2,5,9,8,15,11,6]
8 @ 3 [2,5,9,8,15,11,6]
15 @ 4 [2,5,9,8,15,11,6]
11 @ 5 [2,5,9,8,15,11,6]
6 @ 6 [2,5,9,8,15,11,6]
[ 5, 9, 15, 11 ]
➜  Lecture_35 git:(main) ✗
```

Array.prototype.filter()

The filter() method creates a new array with all elements that pass the test implemented by the provided function.

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/filter#syntax

Syntax

```
// Arrow function
filter((element) => { /* ... */ } )
filter((element, index) => { /* ... */ } )
filter((element, index, array) => { /* ... */ } )

// Callback function
filter(callbackFn)
filter(callbackFn, thisArg)

// Inline callback function
filter(function(element) { /* ... */ })
filter(function(element, index) { /* ... */ })
filter(function(element, index, array){ /* ... */ })
filter(function(element, index, array) { /* ... */ }, thisArg)
```

Parameters

callbackFn

Function is a predicate, to test each element of the array. Return a value that coerces to true to keep the element, or to false otherwise.

It accepts three arguments:

element

The current element being processed in the array.

indexOptional

The index of the current element being processed in the array.

arrayOptional

The array on which filter() was called.

thisArgOptional

Value to use as this when executing callbackFn.

Return value

A new array with the elements that pass the test. If no elements pass the test, an empty array will be returned.

CUSTOM FILTER

```js
Array.prototype.myFilter  = function(cb){
    let oarr = this;
    let res = [];
    for(let i = 0; i < oarr.length; i++){
        let v = oarr[i];
        let rbv = cb(v, i, oarr);

        if(rbv == true){
            res.push(v);
        }
    }
    return res;
}

let arr = [2, 5, 9, 8, 15, 11, 6];
let odd_arr = arr.myFilter(function(v, i, oarr){
    console.log(v + " @ " + i + " [" + oarr + "]");
    if(v % 2 == 1){
        return true;
    } else {
        return false;
    }
});
console.log(odd_arr);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_35 git:(main) ✗ node 2_CustomFilter.js
2 @ 0 [2,5,9,8,15,11,6]
5 @ 1 [2,5,9,8,15,11,6]
9 @ 2 [2,5,9,8,15,11,6]
8 @ 3 [2,5,9,8,15,11,6]
15 @ 4 [2,5,9,8,15,11,6]
11 @ 5 [2,5,9,8,15,11,6]
6 @ 6 [2,5,9,8,15,11,6]
[ 5, 9, 15, 11 ]
→  Lecture_35 git:(main) ✗
```

Ques 1 – Array Filter and Map

```javascript
let arr = [
    {
        name: "Harris Pierce",
        age: 50,
        gender: "M"
    },
    {
        name: "Lea Clearwater",
        age: 25,
        gender: "F"
    },
    {
        name: "Jacob Black",
        age: 18,
        gender: "M"
    },
    {
        name: "Edward Cullen",
        age: 17,
        gender: "M"
    },
    {
        name: "Isabella Swanson",
        age: 68,
        gender: "F"
    },
    {
        name: "Damon Salvatore",
        age: 36,
        gender: "M"
    },
    {
        name: "Elena Gilbert",
        age: 43,
        gender: "F"
    },
    {
        name: "Caroline Forbes",
        age: 23,
        gender: "F"
    }
]

// age of all ladies

// approach 1
let ladies = arr.filter( p => p.gender == "F");
console.log(ladies);
let lages = ladies.map(l => l.age);
console.log(lages);

// approach 2
let ladiesages = arr.filter( p => p.gender == "F").map(l => l.age);
console.log(ladiesages);

// approach 3
let ladiesages3 = arr.filter( (v, i, oarr) => v.gender == "F").map((v, i, oarr)=> v.age)
console.log(ladiesages3);

// approach 3
let ladies_ages = arr.filter(function(v, i){
    if(v.gender == 'F'){
        return true;
    }else{
        return false;
    }
}).map(function(v, i){
    return v.age;
});
console.log(ladies_ages);
```

```
[
  { name: 'Lea Clearwater', age: 25, gender: 'F' },
  { name: 'Isabella Swanson', age: 68, gender: 'F' },
  { name: 'Elena Gilbert', age: 43, gender: 'F' },
  { name: 'Caroline Forbes', age: 23, gender: 'F' }
]
[ 25, 68, 43, 23 ]
[ 25, 68, 43, 23 ]
[ 25, 68, 43, 23 ]
[ 25, 68, 43, 23 ]
```

map() v/s filter()

The difference between map() and filter() ?

- They both return a new array.
- map returns a new array of elements where you have applied some function on the element so that it changes the original element (typically). filter returns a new array of the elements of the original array (with no change to the original values). filter will only return elements where the function you specify returns a value of true for each element passed to the function.
- So map returns the same number of elements as the original, but the element value will be transformed in some way and filter will return the same or less number of elements than the original but not change the original elements' values.

```
1
2   let arr = [2, 9, 18, 32, 74, 57, 63, 58];
3   let even = arr.filter(function(v, i){
4       if(v % 2 == 0){
5           return true;
6       }else{
7           return false;
8       }
9   })
10
11  console.log(even);
12
13  let odd = arr.filter(v => v % 2 == 1);
14
15  console.log(odd);
16
17  let v1 = arr.filter(v => v % 2 == 1);
18  let v2 = arr.map(v => v % 2 == 1);
19  console.log(v1);
20  console.log(v2);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE

→  Lecture_35 git:(main) ✗ node 4_Arrays_Map_VS_FilterDemo.js
[ 2, 18, 32, 74, 58 ]
[ 9, 57, 63 ]
[ 9, 57, 63 ]
[
  false, true,
  false, false,
  false, true,
  true,  false
]
```

Ques 2 - Array map and filter

```
1   // square of even numbers
2
3   let arr = [2, 5, 9, 18, 7, 34, 42, 45, 32];
4   let evenSq = arr.filter(v => v % 2 == 0).map(v => v * v);
5   console.log(evenSq);
6
7
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_35 git:(main) ✗ node 5_Arrays_Map_and_FilterDemo.js
[ 4, 324, 1156, 1764, 1024 ]
→  Lecture_35 git:(main) ✗
```

Ques 3- Array map and filter

```
2   let products = [
3       { name: "T-Shirt" , price: 25},
4       { name: "Headphones" , price: 125},
5       { name: "Keyboard" , price: 75},
6       { name: "Monitor" , price: 200},
7   ];
8
9   //return names of all products in uppercase who has price >= 100
10
11
12  console.log("names of all products in uppercase who has price >= 100");
13
14  let fprds = products.filter(p => p.price >= 100).map(p => p.name.toUpperCase());
15  console.log(fprds);
16
17  let fprds2 = products.filter( function(v, i, oarr){
18      if(v.price >= 100){
19          return true;
20      }else{
21          return false;
22      }
23  }).map(function(v, i, oarr){
24      return v.name.toUpperCase();
25  });
26
27  console.log(fprds2);
28
```

```
console.log("names of all products >= 100 in uppercase and < 100 lowercase");

// all >= 100 name uppercase and < 100 lowercase

let mprds = products.map(function(v){
    if(v.price >= 100){
        return v.name.toUpperCase();
    }else{
        return v.name.toLowerCase();
    }
});

console.log(mprds);


let mprds2 = products.map(p => p.price >=100 ? p.name.toUpperCase() : p.name.toLowerCase());

console.log(mprds2);
```

```
➜  Lecture_35 git:(main) ✗ node 6_Arrays_Map_and_FilterQues1.js
names of all products in uppercase who has price >= 100
[ 'HEADPHONES', 'MONITOR' ]
[ 'HEADPHONES', 'MONITOR' ]
names of all products >= 100 in uppercase and < 100 lowercase
[ 't-shirt', 'HEADPHONES', 'keyboard', 'MONITOR' ]
[ 't-shirt', 'HEADPHONES', 'keyboard', 'MONITOR' ]
➜  Lecture_35 git:(main) ✗
```

Ques 4- Array map and filter

```js
let arr = [5, 83, 24, 67, 71, 12, 24, 7];

// return cubes of values whose square <= 1000
// [5, 24, 12, 24, 7];

let cubes = arr.filter(v => v * v <= 1000).map(v => v * v * v);
console.log(cubes);

// Q1 gives me cubes of number whose cubes are less than 10000

let cubes2 = arr.filter(v => v * v * v <= 10000).map(v => v * v * v);
console.log(cubes2);

let arrnew=arr.filter((v,i)=>(v*v*v)<10000).map((v,i)=>v*v*v);
console.log(arrnew);

let ans1 = arr.map(v => v * v * v).filter(v3 => v3 <= 10000);

// Q2 what is this returning = V6 for elements whose square <= 1000

let ans2 = arr.map(val => val * val).filter(v2 => v2 <= 1000).map(v2 => v2 * v2 * v2);
console.log(ans2);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_35 git:(main) ✗ node 7_Arrays_Map_and_FilterQues2.js
[ 125, 13824, 1728, 13824, 343 ]
[ 125, 1728, 343 ]
[ 125, 1728, 343 ]
[ 15625, 191102976, 2985984, 191102976, 117649 ]
```