ARRAYS MAP

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

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

```
// node 1_ArraysMapDemo

// Map is itself a function
// Map takes as input a callback function
// The callback function takes 3 parameter (v, i, arr)
// Map will call the callback multiple times (once for each value)
// For each run of callback, map will pass v, i and original array to callback
// Callback will process the value and index and return a single value
// Single value returned by each run of callback will be collected in a new array
// Map returns that new array equal in length to original array.
// Map returns that new array


let arr = [2, 5, 9, 8, 15, 11, 6];
let sqarr = arr.map(function (v, i, oarr){
    console.log(v + " @ " + i + " = [" + oarr + "]");
    return v * v;
});
console.log(sqarr);

// let sqarr = arr.map(v => v * v);
// console.log(sqarr);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_34 git:(main) ✗ node 1_ArraysMapDemo.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]
[
    4,  25, 81, 64,
  225, 121, 36
]
```

Map calls a provided callbackFn function once for each element in an array, in order, and constructs a new array from the results. callbackFn is invoked only for indexes of the array which have assigned values (including undefined).

https://www.freecodecamp.org/news/javascript-map-how-to-use-the-js-map-function-array-method/

Sometimes you may need to take an array and apply some procedure to its elements so that you get a new array with modified elements.

Instead of manually iterating over the array using a loop, you can simply use the built-in Array.map() method.

The Array.map() method allows you to iterate over an array and modify its elements using a callback function. The callback function will then be executed on each of the array's elements.

For example, suppose you have the following array element:

```
let arr = [3, 4, 5, 6];
```

Now imagine you are required to multiply each of the array's elements by 3. You might consider using a for loop as follows:

```
let arr = [3, 4, 5, 6];

for (let i = 0; i < arr.length; i++){
  arr[i] = arr[i] * 3;
}

console.log(arr); // [9, 12, 15, 18]
```

But you can actually use the Array.map() method to achieve the same result. Here's an example:

```
let arr = [3, 4, 5, 6];

let modifiedArr = arr.map(function(element){
    return element *3;
});

console.log(modifiedArr); // [9, 12, 15, 18]
```

The Array.map() method is commonly used to apply some changes to the elements, whether multiplying by a specific number as in the code above, or doing any other operations that you might require for your application.

How to use map() over an array of objects

For example, you may have an array of objects that stores firstName and lastName values of your friends as follows:

```
let users = [
  {firstName : "Susan", lastName: "Steward"},
  {firstName : "Daniel", lastName: "Longbottom"},
  {firstName : "Jacob", lastName: "Black"}
] ;
```

An array of objects

You can use the map() method to iterate over the array and join the values of firstName and lastName as follows:

```
let users = [
  {firstName : "Susan", lastName: "Steward"},
  {firstName : "Daniel", lastName: "Longbottom"},
  {firstName : "Jacob", lastName: "Black"}
];
```

```
let userFullnames = users.map(function(element){
    return `${element.firstName} ${element.lastName}`;
})

console.log(userFullnames);
// ["Susan Steward", "Daniel Longbottom", "Jacob Black"]
```

Use map() method to iterate over an array of objects

The map() method passes more than just an element. Let's see all arguments passed by map() to the callback function.

The complete map() method syntax

The syntax for the map() method is as follows:

```
arr.map(function(element, index, array){  }, this);
```

The callback function() is called on each array element, and the map() method always passes the current element, the index of the current element, and the whole array object to it.

The this argument will be used inside the callback function. By default, its value is undefined . For example, here's how to change the this value to the number 80:

```
let arr = [2, 3, 5, 7]

arr.map(function(element, index, array){
    console.log(this) // 80
}, 80);
```

Assigning number value to map() method this argument

```js
// Use the map function to produce the below output
// ["H.P.", "S.S.", "L.E.", "R.W.", "A.D.", "D.T.", "D.M."]
let arr = [
    "Harry Potter",
    "Severus Snape",
    "Lily Evans",
    "Ronald Weasely",
    "Albus Dumbledore",
    "Dean Thomas",
    "Draco Malfoy"
]
let iarr = arr.map(function(v, i){
    console.log(v + " @ " + i);
    let names = v.split(" ");
    console.log(names);
    let ans = names[0][0] + "." + names[1][0] + ".";
    return ans;
});
console.log(iarr);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
[ 'Harry', 'Potter' ]
Severus Snape @ 1
[ 'Severus', 'Snape' ]
Lily Evans @ 2
[ 'Lily', 'Evans' ]
Ronald Weasely @ 3
[ 'Ronald', 'Weasely' ]
Albus Dumbledore @ 4
[ 'Albus', 'Dumbledore' ]
Dean Thomas @ 5
[ 'Dean', 'Thomas' ]
Draco Malfoy @ 6
[ 'Draco', 'Malfoy' ]
[
  'H.P.', 'S.S.',
  'L.E.', 'R.W.',
  'A.D.', 'D.T.',
  'D.M.'
]
```

```
let arr = [
    {
        gender: 'M',
        age: 24
    },
    {
        gender: 'F',
        age: 34
    },
    {
        gender: 'F',
        age: 28
    },
    {
        gender: 'M',
        age: 74
    },
    {
        gender: 'F',
        age: 31
    },
    {
        gender: 'M',
        age: 47
    },
    {
        gender: 'F',
        age: 26
    },
    {
        gender: 'M',
        age: 47
    },
    {
        gender: 'F',
        age: 47
    },
    {
        gender: 'F',
        age: 19
    },
    {
        gender: 'M',
        age: 20
    }
];

// Use the map function to produce the below output
// return an array with true and false for females between 20 and 30
// let us say xyz corp wants to hire females between age >= 20 and <=30

let shortlist = arr.map(function(v, i, oarr){
    if(v.gender == 'F' && v.age >= 20 && v.age <= 30){
        return true;
    }else{
    return false;
    }
});

console.log(shortlist);

// Using arrow function

let shortlist2 = arr.map((v, i, oarr) => v.gender == 'F' && v.age >= 20 && v.age <= 30);
console.log(shortlist2);

let shortlist3 = arr.map((v, i, oarr) => {
    return v.gender == 'F' && v.age >= 20 && v.age <= 30
});
console.log(shortlist3);
```

```
→  Lecture_34 git:(main) ✗ node 3_Ques2_ArraysMap.js
[
  false, false, true,
  false, false, false,
  true,  false, false,
  false, false
]
[
  false, false, true,
  false, false, false,
  true,  false, false,
  false, false
]
[
  false, false, true,
  false, false, false,
  true,  false, false,
  false, false
]
```

Javascript > Lecture_34 > JS ArraysMapDemo_3.js > ...

```
1   let arr = [2, 19, 34, 72, 11, 64, 55, 98];
2
3   let barr = arr.map(function(v, i){
4       if(v % 2 == 0){
5           return true;
6       }else{
7           return false;
8       }
9   });
10
11  console.log(barr);
```

using square function

```
let barr = arr.map(
    v => v%2==0);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_34 git:(main) ✗ node ArraysMapDemo_3.js
[
  true,  false,
  true,  true,
  false, true,
  false, true
]
→  Lecture_34 git:(main) ✗
```

Real life example of map function in hackerrank automation

```
75
76  async function handlePage(browser, page) {
77      // do some code
78      await page.waitForSelector("a.backbone.block-center");
79      let curls = await page.$$eval("a.backbone.block-center", function (atags) {
80          // let urls = [];
81
82          // for (let i = 0; i < atags.length; i++) {
83          //     let url = atags[i].getAttribute("href");
84          //     urls.push(url);
85          // }
86
87          let urls = atags.map(function(atag, i){
88              return atag.getAttribute("href");
89          })
90          return urls;
91      });
92
```

CUSTOM MAP

```
1
2   Array.prototype.myMap = function(callback){
3       let res = [];
4       for(let i = 0; i < this.length; i++){
5           let val = this[i];
6           let rv = callback(val, i, this);
7           res.push(rv);
8       }
9       return res;
10  }
11  // Map is itself a function
12  // Map takes as input a callback function
13  // The callback function takes 3 parameter (v, i, arr)
14  // Map will call the callback multiple times (once for each value)
15  // For each run of callback, map will pass v, i and original array to callback
16  // Callback will process the value and index and return a single value
17  // Single value returned by each run of callback will be collected in a new array
18  // Map returns that new array equal in length to original array.
19  // Map returns that new array
20  let arr = [2, 5, 9, 8, 15, 11, 6];
21
22  let sqarr1 = arr.map(function (v, i){
23      return v * v;
24  });
25  console.log(sqarr1);
26
27  let sqarr2 = arr.myMap(function (v, i){
28      return v * v;
29  });
30  console.log(sqarr2);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
→  Lecture_34 git:(main) ✗ node 4_ArraysCustomMap.js
[
    4,  25, 81, 64,
  225, 121, 36
]
[
    4,  25, 81, 64,
  225, 121, 36
]
→  Lecture_34 git:(main) ✗
```