CS 1520: Recitation 11

Demystifying JS map(), reduce(), filter()

map()

- The map() method calls the provided function once for each element in an array, in order.

Syntax

```
var new_array = arr.map(function callback(currentValue[, index[, array]]) {
    // Return element for new_array
}[, thisArg])
```

Parameters

callback

Function that produces an element of the new Array, taking three arguments:

currentvalue

The current element being processed in the array.

index|Optional

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

array|Optional

The array map was called upon.

thisArg|Optional

Optional. Value to use as this when executing callback.

Example 1

```
// What you have
var officers = [
  { id: 20, name: 'Captain Piett' },
  { id: 24, name: 'General Veers' },
  { id: 56, name: 'Admiral Ozzel' },
  { id: 88, name: 'Commander Jerjerrod' }
];
// What you need
[20, 24, 56, 88]
```

Using .foreach

```
var officersIds = [];
officers.forEach(function (officer) {
  officersIds.push(officer.id);
});
```

Using map()

```
var officersIds = officers.map(function
(officer) {
  return officer.id
});
```

Also, using map()

```
const officersIds = officers.map(officer
=> officer.id);
```

Exercise 1

Output of this code snippet

```
let vals = [4,8,1,2,9];
console.log(vals);
function doubler(x){
        return x*2;
}
vals.map(doubler)
console.log(vals)
```

Output in the console

```
[4,8,1,2,9]
[4,8,1,2,9]
```

Reason: map() returns a new array and `vals.map(doubler)` does not modify the existing array.

Exercise 2

Output of this code snippet

```
let vals = [4,8,1,2,9];
console.log(vals);
function doubler(x){
        return x*2;
}
vals = vals.map(doubler)
console.log(vals)
```

Output in the console

```
[4,8,1,2,9]
[8,16,2,4,18]
```

Exercise 3

- Write the most concise version of the map function in the previous example (slide 5)

Concise version

```
let vals = [4,8,1,2,9];
console.log(vals);
vals = vals.map(x=>x*2)
console.log(vals)
```

reduce()

- Reduces the array to a single value

Syntax

```
arr.reduce(callback[, initialValue])
```

Parameters

```
callback
    Function to execute on each element in the array, taking four arguments:
    The two mandatory arguments are:
    accumulator
        accumulates the callback's return values; it is the accumulated value
        previously returned in the last invocation of the callback, or
        initialValue, if supplied
    currentValue
        The current element being processed in the array
initialValue|Optional
    Initial value of the accumulator.
```

Example: summation using reduce()

Summation of array elements using reduce()

```
let vals = [5,4,1,2,9];
function sum(acc, curr){
        return acc + curr
}
let answer = vals.reduce(sum, 0);
console.log(answer)
```

Output in the console

```
21
```

Exercise

Give Output of the following

```
let vals = [5,4,1,2,9];
function sum(acc, curr){
        console.log(acc)
        return acc + curr
}
let answer = vals.reduce(sum);
console.log(answer)
```

Output in the console

```
5
9
10
12
21
```

Reason: acc is not initialized to any value, so the initial value of acc is 5 (value of array's first element)

Exercise

- Write the most concise version of the reduce() function in slide 8

Concise Version

```
let vals = [5,4,1,2,9];
let answer = vals.reduce((acc,curr)=>acc + curr, 0);
console.log(answer)
```

Example – finding max value using reduce()

Finding max value using reduce()

```
let vals = [5,4,1,2,9];
function findmax(acc, curr){
	if (curr > acc) {
		acc = curr;
	}
	return acc
}
let biggest = vals.reduce(findmax, 0);
console.log(biggest);
```

Output in the console

```
9
```

Exercise

- Condense the reduce() in the previous example (slide 11) to the most concise form

Concise Version

```
let vals = [5,4,1,2,9];
let biggest = vals.reduce((a, b)=> a > b ? a:b, 0);
console.log(biggest)
```

Exercise

Give output of the following:

`let vals = [5,4,1,2,9];`
`let biggest = vals.reduce((a, b)=> a > b ? a:b, 20);`
`console.log(biggest)`

Output in the console
20

filter()

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

Syntax

```
var newArray = arr.filter(callback(element[, index[, array]])[,
thisArg])
```

Parameters

callback
    Function is a predicate, to test each element of the array. Return true to keep the element, false otherwise. It accepts three arguments:
    element
        The current element being processed in the array.
    index|Optional
        The index of the current element being processed in the array.
    array|Optional
        The array filter was called upon.
thisArg|Optional
    Optional. Value to use as this when executing callback.

Example

Finding even numbers in an array

```
let vals = [5, 4, 1, 2, 9];
function isEven(curr){
        return (curr%2 == 0)
}
vals = vals.filter(isEven);
console.log(vals);
```

Output in the console

```
[4, 2]
```

Exercise

Filter out **undefined** from the given array

```
let vals = [5,4,undefined,2,1];
```

Solution Code:

```
let vals = [5,4,undefined,2,1];
vals = vals.filter(x=>x);
console.log(vals);
```

Output in the console

```
[5, 4, 2, 1]
```

Filters out the ***falsy*** value (undefined)

'Falsy' Values in JS

- A falsy value is a value that translates to false when evaluated in a Boolean context.
  - false (boolean false)
  - null
  - undefined
  - 0 (number zero)
  - '' or "" (empty string)
  - NaN

Example

Filtering out Falsy values

```
const noFalsyAndZero = [
  null,
  0,
  1,
  -2,
  50,
  undefined,
  true,
  false,
  '',
  NaN
].filter(Boolean);
console.log(noFalsyAndZero);
```

Output in the console

```
[ 1, -2, 50, true ]
```

Exercise

Give output of the following:

`let vals = [5,4,4,2,1];`
`vals = vals.filter(x => x%2);`
`console.log(vals)`

Output in the console
`[5,1]`

Example – Filtering Strings

- Consider the following example:

Splitting Words In a Sentence
let s = “This is CS 1520”; Let words = s.split(/\W+/); console.log(words)

Output in the console
[“This”, “is”, “”, “CS”, “1520”]

Reason: Since there are two spaces between “is” and “CS”, so it shows an empty string in the console

Example – Filtering Strings

- Filtering Out the '*falsy*' empty string

Filtering out '*falsy*' empty string

`let s = "This is  CS 1520";`
`Let words = s.split(/\W+/).filter(word=> word.length);`
`console.log(words)`

Output in the console
`["This", "is", "CS", "1520"]`

Combining map(), reduce() and filter()

Get the total score of force users only

```
var personnel = [
  {
    id: 5,
    name: "Luke Skywalker",
    pilotingScore: 98,
    shootingScore: 56,
    isForceUser: true,
  },
  {
    id: 22,
    name: "Zeb Orellios",
    pilotingScore: 20,
    shootingScore: 59,
    isForceUser: false,
  },
{
    id: 15,
    name: "Ezra Bridger",
    pilotingScore: 43,
    shootingScore: 67,
    isForceUser: true,
},
{
    id: 11,
    name: "Caleb Dume",
    pilotingScore: 71,
    shootingScore: 85,
    isForceUser: true,
  },
];);
```

Combining map(), reduce() and filter()

Using filter(), map() and reduce() to get the total score of force users

```
const totalJediScore = personnel
  .filter(person => person.isForceUser)
  .map(jedi => jedi.pilotingScore + jedi.shootingScore)
  .reduce((acc, score) => acc + score, 0);
```

References

- https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
- https://www.youtube.com/playlist?list=PLRqwX-V7Uu6YgpA3Oht-7B4NBQwFVe3pr
- https://medium.com/poka-techblog/simplify-your-javascript-use-map-reduce-and-filter-bd02c593cc2d
- https://medium.com/@sgobinda007/dealing-with-javascript-falsy-values-d75a2f1b1c90
- https://www.w3schools.com/jsref/jsref_obj_array.asp