map-filter-reduce

Map

The `map()` method is used for creating a new array from an existing one, applying a function to each one of the elements of the first array.

Example

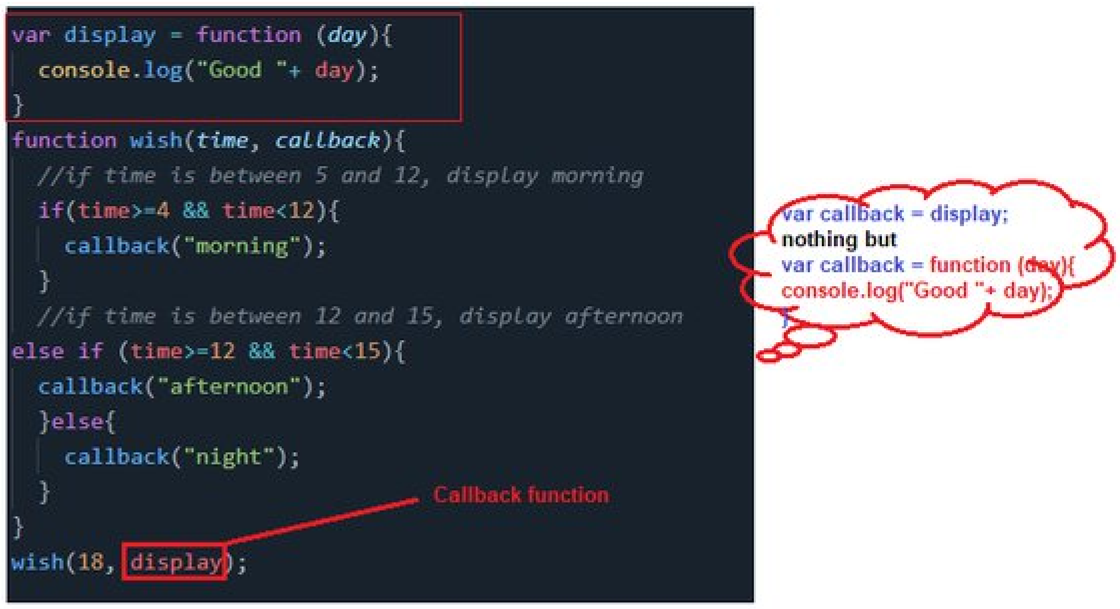
```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(item => item * 2);
console.log(doubled); // [2, 4, 6, 8]
```

Polyfill for map()

```
Array.prototype.myMap = function(cb) {

let temp = [];
for(let i = 0;i < this.length; i++) {
    temp.push(cb(this[i],i,this); //here this indicates the nums array
}
return temp;
};

const nums = [1,2,3,4];

const multiplyThree = nums.myMap((num,i,arr)=>{
   return num * 3;
});

console.log(multiplyThree);
```

If you forgot how callback works follow the link video and take a look at the image:

https://youtu.be/kz_vwAF4NHI



```
var display = function (day){
  console.log("Good "+ day);
}
function wish(time, callback){
  //if time is between 5 and 12, display morning
  if(time>=4 && time<12){
    callback("morning");
  }
  //if time is between 12 and 15, display afternoon
else if (time>=12 && time<15){
  callback("afternoon");
  }else{
    callback("night");
  }
}
wish(18, display);
```

```
var callback = display;
nothing but
var callback = function (day){
console.log("Good "+ day);
}
```

Filter

The `filter()` method takes each element in an array and it applies a conditional statement against it. If this conditional returns true, the element gets pushed to the output array. If the condition returns false, the element does not get pushed to the output array.

Examples

```
const numbers = [1, 2, 3, 4];
const evens = numbers.filter(item => item % 2 === 0);
console.log(evens); // [2, 4]
```

Polyfill for filter()

```
Array.prototype.myFilter = function (cb) {

    let temp = [];
    for (let i = 0; i < this.length; i++) {
        if (cb(this[i], i, this)) {
            temp.push(this[i]);
        } //here this indicates the nums array
    }
    return temp;
};
```

```
const nums = [1, 2, 3, 4];

const moreThanTwo = nums.myFilter((num, i, arr) => {
    return num > 2;
});

console.log(moreThanTwo);
```

Reduce

The `reduce()` method reduces an array of values down to just one value. To get the output value, it runs a reducer function on each element of the array.

```
const nums = [1,2,3,4];

const sum = nums.reduce((acc,curr,i,arr) => {
  return acc + curr;
},0);

console.log(sum);
```

- *accumulator* - the returned value of the previous iteration
- *currentValue* - the current item in the array
- *index* - the index of the current item
- *array* - the original array on which reduce was called
- The `initialValue` argument is optional. If provided, it will be used as the initial accumulator value in the first call to the callback function.

Polyfill for reduce()

```
Array.prototype.myReduce = function (cb, initialValue) {

    let accmulator = initialValue;
    for (let i = 0; i < this.length; i++) {
        accmulator = accmulator ? cb(accmulator, this[i], i, this) : this[i];
    }
    return accmulator;
};

const nums = [1, 2, 3, 4];
const sum = nums.myReduce((acc, curr, i, arr) => {
    return acc + curr;
}, 0);

console.log(sum);
```

map vs forEach

- `map` creates a new array by transforming each element using a provided function.
- `forEach` iterates through the array, applying a function for each element, without creating a new array.
- If you need to execute some code for each element in an array without producing a new array, `forEach` is a good choice.

Question

Return total marks for students with marks greater than 60 after 20 marks have been added to those who scored less than 60

```
let students = [
    { name: 'Hayat', rollNumber: 15, marks: 85 },
    { name: 'Hossain', rollNumber: 31, marks: 75 },
    { name: 'Nakib', rollNumber: 16, marks: 35 },
    { name: 'Chowdhury', rollNumber: 7, marks: 55 }
]


const totalMarks = students.map(stu => {
    if (stu.marks < 60) {
        stu.marks += 20;
    }

    return stu;
}).filter(stu => stu.marks > 60).reduce((acc, curr) => acc + curr.marks, 0)

console.log(totalMarks)
```