Api 배열요소에 반복방식 적용(https://velog.io/@undefined/%EB%B0%B0%EC%97%B4%EC%9D%98-API)

설명:(value)는 arr의 각 요소들이 배치하게 됨.
모든 arr의 요소에
->“ ”+요소+“
”값을 추가해라

```
console.log("JavaScript Repeat Array test")

let arr = [1, 3, 8, 10, 7, 11, 19, 77, 33]
let txt = ""

arr.forEach(testFunc)

function testFunc (value) {
    txt = txt + value + "<br>"
}

console.log(txt)
```

```
JavaScript Repeat Array test
1<br>3<br>8<br>10<br>7<br>11<br>19<br>77<br>33<br>
```

모든 API의 기본형태

1.let 변수명 = 배열.API(function) //function에는 내부 조건이 들어옴
function function(value){
return 조건식}

2.let 변수명 = 배열.API((element)=>조건식)

.find() 조건에 맞는 첫 번째 요소를 찾아줌

조건에 맞는 첫번째로 찾아진 요소를 찾아줌

기본형태

1.let 변수명 = 배열.find(function) //function에는 내부 조건이 들어옴
function function(value){
return 조건식}

2.let 변수명= 배열.find((element)=>조건식) //arrow function방식

```
let sequence = [1, 2, 3, 4, 5]

let firstFind = sequence.find(firstFindFunc)

function firstFindFunc (value) {
    return value > 3
}
```

```
class student{
    constructor(name,age,enrolled,score){
this.name=name;
this.score=score;
this.age=age;
this.enrolled=enrolled;
    }
}
const students=[
    new student('A',23,true,45),
    new student('B',24,true,80),
    new student('C',25,true,90),
    new student('D',26,true,88),
];
const result = students.find(function(student){
return student.score===90})
console.log(result)
```

결과 값:4 결과 값: student “C” function

arrow function방식

```
const resultScore88=students.find((student)=>student.score===88)
console.log(resultScore88)
```

결과 값: student “D” function

.map() 요소를 뽑아와서 재구성 할 때 주로 사용.

사칙연산에 주로 사용되며 foreach기능도 포함하고 있음

arrow방식

let squareseq=sequence.map((element)=>element*element)

```
let sequence = [1, 2, 3, 4, 5]
let squareSeq = sequence.map(squareFunc)

function squareFunc (value) {
    return value * value
}

console.log("sequence: " + sequence)
console.log("squareSeq: " + squareSeq)
```

```
sequence: 1,2,3,4,5
squareSeq: 1,4,9,16,25
```

.filter() 조건에 맞는 요소 전부

arrow방식

let over3=sequence.filter((element)=>element>3)

```
let over3 = sequence.filter(filterFunc)

function filterFunc (value) {
    return value > 3
}

console.log("3 보다 큰 것: " + over3)
```

```
3 보다 큰 것: 4,5
```

.reduce() 우리가원하는 배열의 시작 점 부터 돌면서 어떤값을 누적할 때

arrow방식

let reduceNum=sequence.reduce((total,element)=>total+element)

```
let reduceNum = sequence.reduce(reduceFunc)

function reduceFunc (total, value) {
    console.log("total: " + total + ", value: " + value)
    return total + value
}

console.log("reduceNum: " + reduceNum)
```

```
total: 1, value: 2
total: 3, value: 3
total: 6, value: 4
total: 10, value: 5
reduceNum: 15
```

결국 전체의 합산 값을 알기 위해선 마지막 value값도 더해야한다.
평균이 구하고싶다면 console.log(reduceFunc/sequence.length)

boolean

.every() 전부가 조건에 만족하냐

.some() 중에 하나라도 조건에 만족하냐

.lastIndexof(배열요소) 몇 번째에 위치하는지 뒤에서부터 하나 찾아줌

.findIndex() 요소의 인덱스를 찾아줌

```
let arr = [1, 3, 8, 10, 7, 11, 19, 77, 33]
let txt = arr.forEach(testFunc)

function testFunc(value){
return value+"<br>"}

console.log(txt)
```

이건 안됨?