배열 메소드의 종류

목 차

- map() 메소드
- filter() 메소드
- reduce() 메소드
- find() 메소드
- some() 메소드
- every() 메소드
- sort() 메소드
- reverse() 메소드

1. PUSH() 메소드

CODE

```
let number = [1, 2, 3, 4]

number.push(5)

console.log(number) // [1, 2, 3, 4, 5]
```

CODE

```
1 let number = [1, 2, 3, 4]
2
3 number.push(5)
4
5 console.log(number) // [1, 2, 3, 4, 5]
```

배열 마지막 자리에 요소를 추가

2. POP() 메소드

CODE

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

number.pop()

console.log(number) // [1, 2, 3, 4]
```

CODE

```
1 let number = [1, 2, 3, 4, 5]
2
3 number.pop()
4
5 console.log(number) // [1, 2, 3, 4]
```

배열의 마지막 요소를 삭제

3. SHIFT() 메소드

CODE

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

number.shift()

console.log(number) // [2, 3, 4, 5]
```

CODE

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

number.shift()

console.log(number) // [2, 3, 4, 5]
```

배열 첫번째 요소를 삭제

4. UNSHIFT() 메소드

CODE

```
let number = [1, 2, 3, 4]

number.unshift(5)

console.log(number) // [5, 1, 2, 3, 4]
```

CODE

```
let number = [1, 2, 3, 4]

number.unshift(5)

console.log(number) // [5, 1, 2, 3, 4]
```

배열 첫번째 자리에 요소를 추가

5. SPLICE() 메소드

CODE

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

number.splice(1, 1, 6)

console.log(number) // [1, 6, 3, 4, 5]
```

CODE

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

number.splice(1, 1, 6)

console.log(number) // [1, 6, 3, 4, 5]
```

n번째부터 n번째 요소까지 삭제하고 바꾸기

CODE

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

number.splice(n, 0, 6, ...)

console.log(number)
```

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

number.splice(0, n, 6, ...)

console.log(number)
```

CODE

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

number.splice(n, 0, 6, ...)

console.log(number)
```

n번째자리에 요소를 추가

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

number.splice(0, n, 6, ...)

console.log(number)
```

CODE

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

number.splice(n, 0, 6, ...)

console.log(number)
```

n번째자리에 요소를 추가

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

number.splice(0, n, 6, ...)

console.log(number)
```

0번부터 n번째 요소까지 삭제하고 바꾸기

CODE

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

number.splice(-n, n)

console.log(number)
```

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

number.splice(n) // .splice(-n)

console.log(number)
```

CODE

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

number.splice(-n, n)

console.log(number)
```

-n번째 자리에서 n번째 요소까지 삭제

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

number.splice(n) // .splice(-n)

console.log(number)
```

CODE

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

number.splice(-n, n)

console.log(number)
```

-n번째 자리에서 n번째 요소까지 삭제

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

number.splice(n) // .splice(-n)

console.log(number)
```

n/-n번째 요소부터 끝까지 삭제

6. SLICE() 메소드

CODE

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

let SlicedNumber = number.slice(1, 2)

console.log(SlicedNumber) // [2]
```

CODE

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

let SlicedNumber = number.slice(1, 2)

console.log(SlicedNumber) // [2]
```

n번째 자리에서 n번째 요소까지를 새로운 배열로 반환

CODE

```
1 let number = [1, 2, 3, 4, 5]
2
3 let SlicedNumber = number.slice(n) // .slice(-n)
4
5 console.log(SlicedNumber)
```

```
1 let number = [1, 2, 3, 4, 5]
2
3 let SlicedNumber = number.slice(n, -n)
4
5 console.log(SlicedNumber)
```

CODE

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

let SlicedNumber = number.slice(n) // .slice(-n)

console.log(SlicedNumber)
```

n/-n번째 자리 요소부터 마지막 요소를 새로운 배열로 반환

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

let SlicedNumber = number.slice(n, -n)

console.log(SlicedNumber)
```

CODE

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

let SlicedNumber = number.slice(n) // .slice(-n)

console.log(SlicedNumber)
```

n/-n번째 자리 요소부터 마지막 요소를 새로운 배열로 반환

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

let SlicedNumber = number.slice(n, -n)

console.log(SlicedNumber)
```

n번째 자리부터 -n번째를 뺀 나머지 요소를
새로운 배열로 반환

7. FOREACH() 메소드

CODE

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

number.forEach(function(Item) {

  console.log(Item) // 1, 2, 3, 4, 5
})
```

CODE

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

number.forEach(function(Item) {

  console.log(Item) // 1, 2, 3, 4, 5
})
```

for문처럼 반복적인 작업을 할때 사용

8. MAP() 메소드

CODE

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

let newNumbers = number.map(function(Item) {

  return item * 2
})

console.log(newNumbers) // [ 2, 4, 6, 8, 10 ]
```

CODE

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

let newNumbers = number.map(function(Item) {

  return item * 2
})

console.log(newNumbers) // [ 2, 4, 6, 8, 10 ]
```

기존에 있는 배열을 가공해서 새로운 배열을 생성

9. FILTER() 메소드

CODE

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

let newNumbers = number.filter(function(Item) {

  return item === 5 // (조건)
})

console.log(newNumbers) // [ 5 ]
```

CODE

```
1 let number = [1, 2, 3, 4, 5]
2
3 let newNumbers = number.filter(function(Item) {
4
5   return item === 5 // (조건)
6 })
7
8 console.log(newNumbers) // [ 5 ]
```

조건에 해당하는만큼 값을 반환, 새로운 배열 생성

10. REDUCE() 메소드

CODE

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

let sum = number.reduce(function(acc, value) {

  return acc + value
} , 0) // 초기값 : 10

console.log(sum) // 15 // 25
```

CODE

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

let sum = number.reduce(function(acc, value) {

  return acc + value
} , 0) // 초기값 : 10

console.log(sum) // 15 // 25
```

함수를 실행해 하나의 결과값을 반환

11. FIND() 메소드

CODE

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

let findNumber = number.find(function(item) {

  return item > 3
})

console.log(findNumber) // 4
```

CODE

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

let findNumber = number.find(function(item) {

  return item > 3
})

console.log(findNumber) // 4
```

조건에 맞는 첫번째 요소를 반환

12. SOME() 메소드

CODE

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

let hasEvenNumber = number.some(function(item) {

  return item % 2 === 0
})

console.log(hasEvenNumber) // true
```

CODE

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

let hasEvenNumber = number.some(function(item) {

  return item % 2 === 0
})

console.log(hasEvenNumber) // true
```

조건에 맞는 요소가 배열안에 하나라도 있으면 true 반환

13. EVERY() 메소드

CODE

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

let isAllNumber = number.every(function(item) {

  return item % 2 === 0
})

console.log(isAllNumber) // false
```

CODE

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

let isAllNumber = number.every(function(item) {

  return item % 2 === 0
})

console.log(isAllNumber) // false
```

조건에 맞는 요소가 배열안에 다 있을때 true 반환
없다면 false 반환

14. SORT() 메소드

CODE

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

number.sort(function(a, b) {

  return a - b (오름차순) // b - a (내림차순)
})

console.log(number) // [1, 2, 3, 4, 5]
```

CODE

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

number.sort(function(a, b) {

  return a - b (오름차순) // b - a (내림차순)
})

console.log(number) // [1, 2, 3, 4, 5]
```

배열의 요소를 정렬하여 그 배열을 반환

CODE

```
let number = [2, 1, 3, 10]

number.sort()

console.log(number) // [1, 10, 2, 3]
```

CODE

```
1 let number = [2, 1, 3, 10]
2
3 number.sort()
4
5 console.log(number) // [1, 10, 2, 3]
```

값이 생략되면, 유니코드 값 순서대로 정렬

15. REVERSE() 메소드

CODE

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

number.reverse()

console.log(number) // [4, 1, 5, 2, 3]
```

CODE

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

number.reverse()

console.log(number) // [4, 1, 5, 2, 3]
```

배열의 요소를 역순으로 정렬

THE END