8253jang@daum.net
장희정

JavaScript Array 메서드 정리

1. push()

- **기능** : 배열의 **끝에 새로운 요소를 추가**
- **구문** : array.push(element1, element2, ...)
- **매개변수** : 추가할 요소들
- **반환값** : 변경된 배열의 길이

예시)

```
const fruits = ['apple', 'banana'];
fruits.push('orange'); // ['apple', 'banana', 'orange']
```

2. pop()

- **기능** : 배열의 **마지막 요소 제거**
- **구문** : array.pop()
- **매개변수** : 없음
- **반환값** : 제거된 요소

예시)

```
const fruits = ['apple', 'banana', 'orange'];
fruits.pop(); // 'orange' -> ['apple', 'banana']
```

3. unshift()

- **기능** : 배열의 **앞에 새로운 요소 추가**
- **구문** : array.unshift(element1, element2, ...)
- **매개변수** : 추가할 요소들
- **반환값** : 변경된 배열의 길이

예시)

```
const fruits = ['banana'];
fruits.unshift('apple'); // ['apple', 'banana']
```

8253jang@daum.net
장희정

4. shift()

- **기능** : 배열의 **첫 번째 요소 제거**
- **구문** : array.shift()
- **매개변수** : 없음
- **반환값**: 제거된 요소

예시)

```
const fruits = ['apple', 'banana'];
fruits.shift(); // 'apple' -> ['banana']
```

5. includes()

- **기능** : 특정 값이 배열에 **포함되어 있는지 확인**
- **구문** : array.includes(searchElement, fromIndex)
- **매개변수** :
    - searchElement: 찾고자 하는 요소
    - fromIndex *(선택)*: 검색 시작 위치 (기본값 0)
- **반환값** : true 또는 false

예시)

```
const numbers = [1, 2, 3];
numbers.includes(2); // true
```

6. indexOf()

- **기능** : 특정 요소의 **인덱스 반환**, 없으면 -1
- **구문** : array.indexOf(searchElement, fromIndex)
- **매개변수** :
    - searchElement: 찾을 값
    - fromIndex *(선택)*: 검색 시작 위치
- **반환값** : 요소의 인덱스 또는 -1

예시)

```
const items = ['pen', 'pencil', 'pen'];
items.indexOf('pen'); // 0
```

7. findIndex()

- **기능** : **조건을 만족하는 첫 요소의 인덱스** 반환

8253jang@daum.net
장희정

- **구문** : array.findIndex(callback(element, index, array))
- **매개변수** :
    - Callback : 각 요소에 대해 실행될 함수 (true 면 해당 인덱스 반환)
- **반환값** : 인덱스 또는 -1

예시)

```
const nums = [10, 20, 30];
nums.findIndex(num => num > 15); // 1
```

8. find()

- **기능**: 조건을 만족하는 **첫 번째 요소 반환**, 없으면 undefined
- **구문**: array.find(callback)

예시)

```
const users = [ {id : 1}, {id : 2} ];
users.find(user => user.id === 2); // {id: 2}
```

9. filter()

- **기능**: 조건을 만족하는 **모든 요소를 새 배열로 반환**
- **구문**: array.filter(callback)

예시)

```
const nums = [1, 2, 3, 4];
nums.filter(n => n % 2 === 0); // [2, 4]
```

10. map()

- **기능**: 모든 요소에 콜백 함수 실행, **결과를 배열로 반환**
- **구문**: array.map(callback)

예시)

```
const nums = [1, 2, 3];
nums.map(n => n * 2); // [2, 4, 6]
```

8253jang@daum.net
장희정

11. reduce()

- **기능**: 배열의 값을 누적하여 **하나의 값 반환**
- **구문**: array.reduce(callback, initialValue)
- **매개변수**:
    - callback(prev, curr, index, array)
    - initialValue *(선택)* : 누적 시작값

예시

```
const nums = [1, 2, 3];
nums.reduce((sum, n) => sum + n, 0); // 6
```

12. some()

- **기능**: **하나라도 조건을 만족하면 true**
- **구문**: array.some(callback)

예시)

```
[1, 2, 3].some(n => n > 2); // true
```

13. every()

- **기능**: **모두 조건을 만족하면 true**
- **구문**: array.every(callback)

예시)

```
[2, 4, 6].every(n => n % 2 === 0); // true
```

14. sort()

- **기능**: 배열을 **정렬(문자열 사전순 또는 비교함수 기반)**
- **구문**: array.sort([compareFunction])

예시)

```
[3, 1, 2].sort((a, b) => a - b); // [1, 2, 3]
```

15. toSorted()

- **기능** : **원본 배열을 유지하면서 정렬된 새로운 배열 반환**
- **구문** : array.toSorted(compareFunction)

8253jang@daum.net
장희정

예시)

```
const arr = [3, 1, 2];
const sorted = arr.toSorted((a, b) => a - b); // [1, 2, 3]
```

16. join()

- **기능**: 배열을 문자열로 **합침**
- **구문**: array.join(separator)

예시)

```
['a', 'b', 'c'].join('-'); // 'a-b-c'
```

17. slice(start, end)

- **기능** : **일부 요소를 잘라 새 배열 반환**
- **구문** : array.slice(start, end)

예시:

```
[1, 2, 3, 4].slice(1, 3); // [2, 3]
```

18. splice(start, count)

- **기능**: 배열에서 **요소 제거 또는 추가**
- **구문**: array.splice(start, deleteCount, item1, item2, ...)

예시)

```
const arr = [1, 2, 3];
arr.splice(1, 1); // [2], arr는 [1, 3]이 됨
```

19. concat()

- **기능**: 배열을 **합쳐 새로운 배열 반환**
- **구문**: array1.concat(array2)

예시)

```
[1, 2].concat([3, 4]); // [1, 2, 3, 4]
```

20. reverse()

- **기능**: 배열의 **순서를 반대로 바꿈**

8253jang@daum.net
장희정

- **구문**: array.reverse()

예시)

```
[1, 2, 3].reverse(); // [3, 2, 1]
```

실습예제

```
const students =[
            {name:"A", age:20 , state:true , score:80},
            {name:"B", age:29 , state:true , score:85},
            {name:"C", age:22 , state:false , score:90},
            {name:"D", age:21 , state:true , score:45},
            {name:"E", age:28 , state:false , score:95},
 ];

//1. 점수가 90점인 학생 정보 - find()
//2. 점수가 90점인 이상인 모든 학생  - filter()
//3. 수강중(state-true) 학생들만 골라서 배열로 만들기. - filter()
//4.학생 배열에서 점수만 뽑아서 배열로 만들기 - map()
//5.학생 들중에 점수가 50점 보다 낮은 친구 있는지 없는지 체크하기-some()
//6. 모든 학생들의 점수가 80이상인지 체크 -every()
//7.학생들의 평균점수를 구하기 - reduce()
//8. 학생들의 모든 점수를 구해서 하나의  string으로 변환(map , join)
//9.학생들의 모든 점수를 배열로 만들어서 50점 이상인 학생들만 뽑아서 하나의 string으로 변환
  (map, filter, join)
//10.학생들의 점수를 정렬하기. - sort()
```

실습 02

```
const products = [
  { id: 1, name: "Laptop", price: 1200000, category: "electronics" },
  { id: 2, name: "Phone", price: 800000, category: "electronics" },
  { id: 3, name: "T-shirt", price: 30000, category: "clothing" },
  { id: 4, name: "Keyboard", price: 50000, category: "electronics" },
  { id: 5, name: "Sneakers", price: 90000, category: "clothing" }
];
```

8253jang@daum.net
장희정

문제01 : 가격이 10만 원 이상인 첫 번째 상품을 찾아 출력하시오 – find().

문제02 : 카테고리가 "clothing"인 상품들만 골라 새로운 배열을 만드시오. – filter()

문제03 : 모든 상품 이름만 추출하여 배열로 만드시오. – map()

문제 04 : 전자제품 중 100 만 원 이상인 상품이 하나라도 있는지 확인하시오. fiter(), some()

문제 05 : 모든 상품이 가격 2 만 원 이상인지 확인하시오. - every()

문제 06 : 상품들의 총합 금액을 구하시오. - reduce() 사용, price 누적

문제 07 : 상품 가격 기준으로 오름차순 정렬된 배열을 만들되, 원본 배열도 변경되게 하시오.
- sort() (주의: 원본 배열 변경됨)

문제 08 : 상품 이름을 기준으로 알파벳순으로 정렬하되, 원본 배열은 그대로 유지되게 하시오.
- toSorted() , localeCompare()

문제 09 : 상품 배열의 처음 3 개만 잘라 새 배열로 반환하시오. - slice(start, end)

문제 10 : 상품 배열에서 id 가 3 인 상품을 찾아 배열에서 제거하시오.
– findIndex()와 splice() 조합