

배열과 메서드

forEach

map

reduce

sort

split / join

배열 전체를 순회하며 반복작업하는 메서드

forEach

map

reduce

forEach

주어진 함수를 배열 요소 각각을 대상으로 반복 작업



```
1 let some = ["Bilbo", "Gandalf", "Nazgul"].forEach((item, index, array) => {  
2     alert(`${item} is at index ${index} in ${array}`);  
3 }); //Bilbo is at index 0 in Bilbo,Gandalf,Nazgul  
4  
5 console.log(some) // undefind 어떤 값을 반환하지않음
```

forEach

주어진 함수를 배열 요소 각각을 대상으로 반복 작업



```
1 let some = ["Bilbo", "Gandalf", "Nazgul"].forEach((item, index, array) => {  
2   alert(`${item} is at index ${index} in ${array}`);  
3 }); //Bilbo is at index 0 in Bilbo,Gandalf,Nazgul  
4  
5 console.log(some) // undefind 어떤 값을 반환하지않음
```

forEach

주어진 함수를 반복 실행할 뿐. 값을 반환 X



```
1 let some = ["Bilbo", "Gandalf", "Nazgul"].forEach((item, index, array) => {  
2     alert(`${item} is at index ${index} in ${array}`);  
3 }); //Bilbo is at index 0 in Bilbo,Gandalf,Nazgul  
4  
5 console.log(some) // undefind 어떤 값을 반환하지않음
```

map

주어진 함수를 배열 요소 각각을 대상으로 반복 작업한 **결과를 모아 새로운 배열로 반환** O



```
1 let result = arr.map(function(item, index, array) {  
2     // 요소 대신 새로운 값을 반환합니다.  
3 });
```

map

주어진 함수를 배열 요소 각각을 대상으로 반복 작업한 **결과를 모아 새로운 배열로 반환** O



```
1 let lengths = ["Bilbo", "Gandalf", "Nazgul"].map(item => item.length);  
2 alert(lengths); // 5,7,6
```

map

```
1 let users = [  
2   { name: "John", surname: "Smith", id: 1 },  
3   { name: "Pete", surname: "Hunt", id: 2 },  
4   { name: "Mary", surname: "Key", id: 3 }  
5 ];  
6  
7 let usersMapped = users.map(item => ({  
8   fullName: item.name + item.surname,  
9   id: item.id  
10  })))  
11  
12 console.log(usersMapped)
```


map

```
1 let users = [  
2   { name: "John", surname: "Smith", id: 1 },  
3   { name: "Pete", surname: "Hunt", id: 2 },  
4   { name: "Mary", surname: "Key", id: 3 }  
5 ];  
6  
7 let usersMapped = users.map(item => ({  
8   fullName: item.name + item.surname,  
9   id: item.id  
10  }));  
11  
12 console.log(usersMapped)
```

```
▼ (3) [{...}, {...}, {...}] ⓘ  
  ▶ 0: {fullName: 'JohnSmith', id: 1}  
  ▶ 1: {fullName: 'PeteHunt', id: 2}  
  ▶ 2: {fullName: 'MaryKey', id: 3}  
    length: 3  
  ▶ [[Prototype]]: Array(0)
```

반복작업
↓
새로운 배열로 반환

forEach**배열 내 요소를 대상으로 반복 작업 + 반환값 X****map****배열 내 요소를 대상으로 반복 작업 + 작업 결과물을 새로운 배열 형태로 반환****reduce****?**

forEach

배열 내 요소를 대상으로 반복 작업 + 반환값 X


map

배열 내 요소를 대상으로 반복 작업 + 작업 결과물을 새로운 배열 형태로 반환

reduce

배열 내 요소를 대상으로 반복 작업 + 배열을 기반으로 누적연산하여 값 하나를 반환

reduce



```
1 let value = arr.reduce(function (accumulator, current, index, array) {  
2     // ...  
3 }, [initial]);
```

- **accumulator** – 이전 함수 호출의 결과. (누적값)
- **current** – 현재 값
- **index** – 현재 요소의 인덱스
- **array** – 순회 중인 배열
- **initial** – 함수 최초 호출 시 사용되는 초기값

reduce

```
1 let value = arr.reduce(function (accumulator, current, index, array) {  
2     // ...  
3 }, [initial]);
```

- **accumulator** – 이전 함수 호출의 결과. (누적값)
- **current** – 현재 값
- **index** – 현재 요소의 인덱스
- **array** – 순회 중인 배열
- **initial** – 함수 최초 호출 시 사용되는 초기값

reduce

주어진 함수를 배열 요소 각각을 대상으로 반복 작업
실행문을 통해 한 값(accumulator)에 누적시켜 **값 하나를 반환**



```
1 let arr = [1, 2, 3, 4, 5];  
2  
3 let result = arr.reduce((sum, current) => sum + current, 0);  
4 let result = arr.reduce((sum, current) => sum + current); // 초기값 생략 가능  
5 alert(result); // 15
```

	sum	current	result
첫 번째 호출	0	1	1
두 번째 호출	1	2	3
세 번째 호출	3	3	6
네 번째 호출	6	4	10
다섯번째 호출	10	5	15

reduce

주어진 함수를 배열 요소 각각을 대상으로 반복 작업
실행문을 통해 한 값(accumulator)에 누적시켜 **값 하나를 반환**



```
1 arr = [] // 빈배열일 경우
2 let result = arr.reduce((sum, current) => sum + current); // 초기값 생략 시
3 alert(result); // 오류 발생
```

sort

배열 자체를 변경시키며, 배열의 요소를 정렬

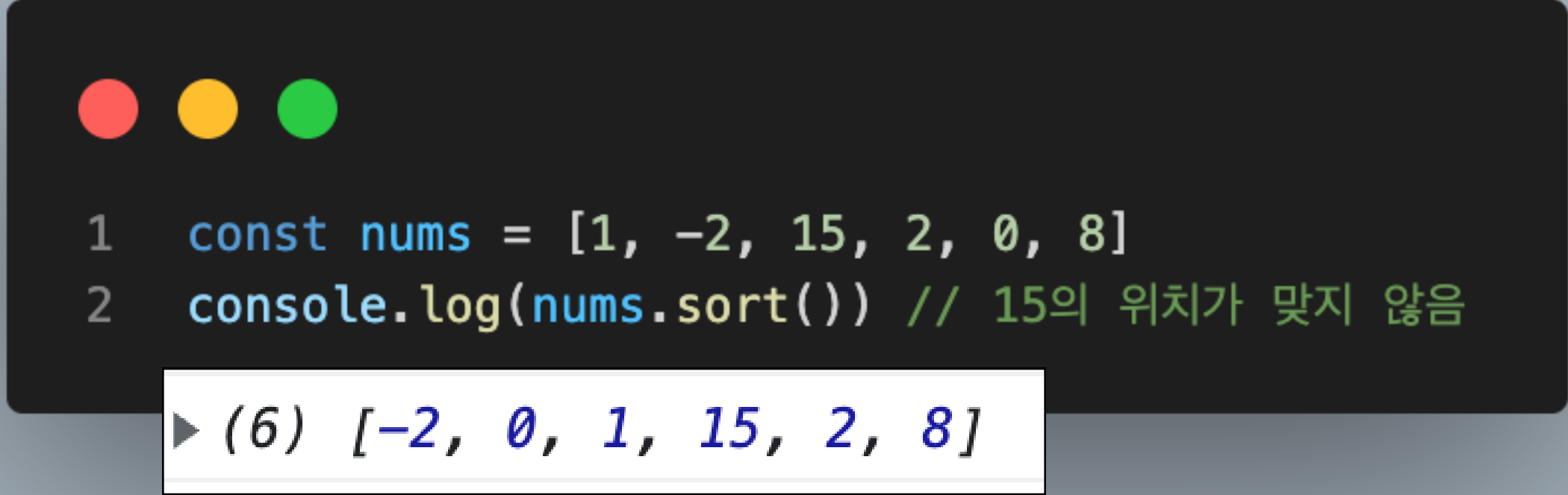


```
1 const nums = [1, -2, 15, 2, 0, 8]
2 console.log(nums.sort()) // 15의 위치가 맞지 않음
```

```
▶ (6) [-2, 0, 1, 15, 2, 8]
```


sort

배열 자체를 변경시키며, 배열의 요소를 정렬



```
1 const nums = [1, -2, 15, 2, 0, 8]
2 console.log(nums.sort()) // 15의 위치가 맞지 않음
```

▶ (6) [-2, 0, 1, 15, 2, 8]

문자열로 취급되어 정렬 (사전순)

sort

배열 자체를 변경시키며, 배열의 요소를 정렬

문자열로 취급되어 정렬 (사전순)



새로운 정렬 기준을 만들려면 `arr.sort()`에 새로운 함수를 넘겨야한다

sort

```
1  nums.sort(function (a, b) {  
2      console.log(a + " <> " + b);  
3      return a - b;  
4  }); //a - b 의 결과가 -1 이면 앞으로 정렬  
5  
6  
7  
8  
9  
10
```

a - b 의 결과가 -1이면 앞으로 정렬
(a가 작으면)

```
[1, -2, 15, 2, 0, 8].sort(function(a, b) {  
    console.log( a + " <> " + b );  
    return a - b;  
});  
-2 <> 1  
15 <> -2  
15 <> 1    -2 1 15  
2 <> 1  
2 <> 15    -2 1 2 15  
0 <> 2  
0 <> 1  
0 <> -2    -2 0 1 2 15  
8 <> 1  
8 <> 15  
8 <> 2    -2 0 1 2 8 15  
▶ (6) [-2, 0, 1, 2, 8, 15]
```

sort

```
1  nums.sort(function (a, b) {  
2      console.log(a + " <> " + b);  
3      return a - b;  
4  }); //a - b 의 결과가 -1 이면 앞으로 정렬  
5  
6  //오름차순  
7  console.log(nums.sort((a, b) => a - b))  
8  
9  //내림차순  
10 console.log(nums.sort((a, b) => b - a))
```

split

인자로 받은 구분자를 기준으로 문자열을 쪼개서 배열로 반환

```
1 let names = 'Bilbo, Gandalf, Nazgul';
2
3 let arr = names.split(', ');
4 for (let name of arr) {
5     console.log(`${name}에게 보내는 메시지`); // Bilbo에게 보내는 메시지
6 }
7
8 //---
9 let str = arr.join(';'); // 배열 요소 모두를 ;를 사용해 하나의 문자열로 합칩니다.
10 console.log(str); // Bilbo;Gandalf;Nazgul
```

join

인자로 받은 요소를 기준으로 요소를 합쳐서 문자열로 반환

```
1 let names = 'Bilbo, Gandalf, Nazgul';
2
3 let arr = names.split(', ');
4 for (let name of arr) {
5     console.log(`${name}에게 보내는 메시지`); // Bilbo에게 보내는 메시지
6 }
7
8 //---
9 let str = arr.join(';'); // 배열 요소 모두를 ;를 사용해 하나의 문자열로 합칩니다.
10 console.log(str); // Bilbo;Gandalf;Nazgul
```

THANK YOU

border-left-width를 borderLeftWidth로 변경하기

split

```
1 camelize("background-color") == 'backgroundColor';
2 camelize("list-style-image") == 'listStyleImage';
3 camelize("-webkit-transition") == 'WebkitTransition';
4
5 function camelize(string) {
6     string.split('-')
7         .map((word, index) => index == 0 ? word //0번째 단어는 그대로
8             :
9             word[0].toUpperCase + word.slice(1)) //그뒤 단어는 word[0] 첫번째문자를 대문자 + 인덱스1번부터 끝까지 잘라서 붙이기
10 }
```


border-left-width를 borderLeftWidth로 변경하기

split**map**

```
1 camelize("background-color") == 'backgroundColor';
2 camelize("list-style-image") == 'listStyleImage';
3 camelize("-webkit-transition") == 'WebkitTransition';
4
5 function camelize(string) {
6     string.split('-')
7     .map((word, index) => index == 0 ? word //0번째 단어는 그대로
8     :
9     word[0].toUpperCase + word.slice(1)) //그뒤 단어는 word[0] 첫번째문자를 대문자 + 인덱스1번부터 끝까지 잘라서 붙이기
10 }
```

border-left-width를 borderLeftWidth로 변경하기

split

map

slice

```
1 camelize("background-color") == 'backgroundColor';
2 camelize("list-style-image") == 'listStyleImage';
3 camelize("-webkit-transition") == 'WebkitTransition';
4
5 function camelize(string) {
6     string.split('-')
7         .map((word, index) => index == 0 ? word //0번째 단어는 그대로
8             :
9             word[0].toUpperCase + word.slice(1)) //그뒤 단어는 word[0] 첫번째문자를 대문자 + 인덱스1번부터 끝까지 잘라서 붙이기
10 }
```