자바스크립트 배열 메소드 정리

▼ Array.pop()

🛠️ X

🔙 제거된 요소
※ 빈 배열의 경우 undefined

💡 배열의 마지막 요소를 제거합니다.

```
> let number = ['1', '2', '3', '4', '5'];
<- undefined
> number.pop();
<- '5'
> number;
<- ▶ (4) ['1', '2', '3', '4']
```

▼ Array.shift()

🛠️ X

🔙 제거된 요소
※ 빈 배열의 경우 undefined

💡 배열의 첫 번째 요소를 제거합니다.

```
> let number = ['1', '2', '3', '4', '5'];
<- undefined
> number.shift();
<- '1'
> number
<- ▶ (4) ['2', '3', '4', '5']
```

▼ Array.splice()

▼ Array.push()

🛠️ 배열의 끝에 추가할 N개의 요소.

🔙 배열의 새로운 길이

💡 배열의 끝에 하나 이상의 요소를 추가합니다.

```
> let number = ['1', '2', '3', '4', '5'];
<- undefined
> number.push('6', '7');
<- 7
> number
<- ▶ (7) ['1', '2', '3', '4', '5', '6', '7']
```

▼ Array.unshift()

🛠️ 배열의 가장 앞에 추가할 N개의 요소

🔙 배열의 새로운 길이

💡 배열의 가장 앞에 하나 이상의 요소를 추가합니다.

```
> let number = ['1', '2', '3', '4', '5'];
<- undefined
> number.unshift('-1', '0');
<- 7
> number
<- ▶ (7) ['-1', '0', '1', '2', '3', '4', '5']
```

▼ Array.slice()

🛠️ `start`
배열의 변경을 시작할 인덱스

`delete count (Option)`
배열에서 제거할 요소의 수

`item N (Option)`
배열에 추가할 N개의 요소

🔙 제거된 요소

💡 배열의 요소를 삭제 또는 교체 하거나 새 요소를 추가합니다.

```
> let number = ['6', '7', '3', '4', '5'];
< undefined
> number.splice(0, 2, '1', '2');
< ▶ (2) ['6', '7']
> number
< ▶ (5) ['1', '2', '3', '4', '5']
```

▼ Array.concat()

🛠️ `value N (option)`
N개의 배열 또는 N개의 값
※ value N을 생략할 경우 기존 배열의 얕은 복사본을 반환

🔙 새로운 array 객체

💡 호출한 배열의 뒤에 N개의 value를 붙여 새로운 값을 반환합니다.

```
> let number1 = ['1', '2', '3'];
< undefined
> let number2 = ['4', '5'];
< undefined
> let number3 = number1.concat(number2);
< undefined
> number3;
< ▶ (5) ['1', '2', '3', '4', '5']
```

▼ Array.some()

🛠️ `begin (Option)`
추출을 시작할 인덱스
※ begin이 없을 경우 0번부터 추출

`end (Option)`
추출을 종료할 인덱스
※ end값에 해당하는 요소는 추출되지 않음
※ end 값이 배열의 길이보다 크다면 배열의 끝까지 추출

🔙 추출된 요소를 포함한 새로운 배열

💡 배열에서 일부 또는 전체 요소를 추출한 새로운 배열을 반환합니다.
※ 반환한 배열은 복사본일 뿐 원본 배열에 영향을 주지 않습니다.

```
> let number = ['1', '2', '3', '4', '5'];
< undefined
> number.slice(1, 4);
< ▶ (3) ['2', '3', '4']
> number
< ▶ (5) ['1', '2', '3', '4', '5']
```

▼ Array.every()

🛠️ `callbackFn`
각 요소를 시험할 함수

`element`
배열에서 처리되는 현재 요소

`index`
처리할 현재 요소의 인덱스

`array`
every를 호출한 배열

`thisArg (Option)`
callbackFn을 실행할 때 this로 사용하는 값

🔙 `true`
callbackFn이 모든 배열 요소에 대해 참인 값을 반환하는 경우

`false` 그 외

🛠️ `callbackFn`
각 요소를 시험할 함수

`element`
배열에서 처리되는 현재 요소

`index`
처리할 현재 요소의 인덱스

`array`
every를 호출한 배열

`thisArg (Option)`
callbackFn을 실행할 때 this로 사용하는 값

🔙 `true`
callbackFn이 배열 요소 중 하나라도 참인 값을 반환하는 경우
※ 하나의 요소라도 참이 반환 될 시 반복문을 멈추고 `true` 를 반환

`false`
하나의 요소도 참을 반환하지 못할 경우

💡 배열의 마지막 요소를 제거하고 해당 요소를 반환합니다.

※ 화살표 함수 사용 예시

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> const isBigEnough = (array) => array > 3;
< undefined
> console.log(number.some(isBigEnough));
  true
< undefined
```

▼ Array.map()

💡 **해당 배열이 모든 요소가 함수 callbackFn을 만족하는지 확인 후 결과를 `true` `false`로 반환합니다.**

※ 화살표 함수 사용 예시

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> const isBigEnough = (array) => array > 3;
< undefined
> console.log(number.every(isBigEnough));
  false
< undefined
> const number = ['5', '6', '7', '8', '9'];
< undefined
> console.log(number.every(isBigEnough));
  true
< undefined
```

▼ Array.forEach()

🛠️ `callback`
각 요소에 대해 실행할 함수. 다음 세 가지 매개변수를 받습니다.

`currentValue`
처리할 현재 요소.

`index`
처리할 현재 요소의 인덱스.

`array`
forEach()를 호출한 배열.

`thisArg`
`callback`을 실행할 때 `this`로 사용할 값.

🔙 **undefined**

💡 **배열을 한 번 순회한다**

※ 화살표 함수 사용 예시

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> number.forEach(element => console.log(element));
  1
  2
  3
  4
  5
< undefined
```

▼ Array.filter()

🛠️ `callback`
새로운 배열 요소를 생성하는 함수. 다음 세 가지 인수를 가집니다.

`currentValue`
처리할 현재 요소.

`index (Option)`
처리할 현재 요소의 인덱스.

`array (Option)`
`map()`을 호출한 배열.

`thisArg (Option)`
`callback`을 실행할 때 `this`로 사용되는 값.

🔙 배열의 각 요소에 대해 `callbackFn`을 실행한 결과를 모은 새로운 배열

💡 배열의 각 요소를 불러 callback 함수의 반환값으로 새로운 배열을 만듭니다.

※ 화살표 함수 사용 예시

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> const number2 = number.map(x => x * 2);
< undefined
> console.log(number2);
  ▶ (5) [2, 4, 6, 8, 10]
```

▼ Array.fill()

🛠️ `value`
배열을 채울 값.

`start (Option)`
시작 인덱스, 기본 값은 0.

`end (Option)`
끝 인덱스, 기본값은 `this.length`.

🔙 변형한 배열

💡 시작 인덱스부터 끝 인덱스까지 `value`값으로 채운 후 해당 배열을 반환합니다.
※ 복사본 또는 가상의 배열을 반환하는 것이 아닌 원본의 배열을 변형하여 반환합니다.

🛠️ `callback`
각 요소를 시험할 함수.
`true`를 반환하면 요소를 유지하고, `false`를 반환하면 버립니다.
다음 세 가지 매개변수를 받습니다.

`element`
처리할 현재 요소.

`index (Option)`
처리할 현재 요소의 인덱스.

`array (Option)`
`filter`를 호출한 배열.

`thisArg (Option)`
`callback`을 실행할 때 `this`로 사용하는 값.

🔙 테스트를 통과한 요소로 이루어진 새로운 배열을 반환합니다.
※ 어떠한 요소도 테스트를 통과하지 못할 경우 빈 배열을 반환합니다.

💡 주어진 함수의 테스트를 통과한 모든 요소들을 모아 새로운 배열로 반환합니다.

※ 화살표 함수 사용 예시

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> const number2 = number.filter(num => num > 3);
< undefined
> console.log(number2);
  ▶ (2) ['4', '5']
```

▼ Array.includes()

🛠️ `valueToFind`
탐색할 요소.
※ 문자나 문자열을 비교할 때, `includes()`는 대소문자를 구분합니다.

`fromIndex (Option)`
이 배열에서 검색을 시작할 위치이며 기본값은 0입니다.
※ 음의 값일 경우 (배열의 마지막 인덱스 - `fromIndex`)부터 계산합니다.

🔙 `Boolean`

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> console.log(number.fill(6, 2, 5));
  ▸ (5) ['1', '2', 6, 6, 6]
< undefined
> console.log(number.fill(0, 0, 3));
  ▸ (5) [0, 0, 0, 6, 6]
```

▼ Array.reduce()

🛠️ `callback`
배열의 각 요소에 대해 실행할 함수. 다음 네 가지 인수를 받습니다.

`accumulator`
누적값 입니다.
만약 `initialValue (기본값)`를 제공한 경우에는 `initialValue (기본값)`의 값입니다.

`currentValue`
처리할 현재 요소.

`currentIndex (Option)`
처리할 현재 요소의 인덱스. `initialValue (기본값)`를 제공한 경우 0, 아니면 1부터 시작합니다.

`array (Option)`
`reduce()`를 호출한 배열.

`initialValue (Option)`
최초 호출 시 `accumulator`에 제공하는 기본값 입니다.
※ 빈 배열에서 초기값 없이 `reduce()`를 호출하면 오류가 발생합니다.

🔙 누적 계산의 결과 값.

💡 지정한 인덱스부터 누적 합산한 값을 반환합니다.

※ 화살표 함수 사용 예시

```
> const number = [1, 2, 3, 4, 5];
< undefined
> const initialValue = 0;
< undefined
> const sumWithInitial = number.reduce(
    (accumulator, currentvalue) => accumulator
      + currentvalue, initialValue
  );
< undefined
> console.log(sumWithInitial);
  15
```

💡 해당 배열이 `valueToFind` 가지고 있는지 확인 후 결과를 `Boolean`으로 반환합니다.

```
> const number = [1, 2, 3, 4, 5];
< undefined
> console.log(number.includes(3));
  true
< undefined
> console.log(number.includes(3, 3));
  false
```

▼ Array.reverse()

🛠️ X

🔙 순서가 반전된 배열.

💡 배열의 순서를 반전시킵니다.
※ 원본 배열을 변형시킵니다.

```
> const number = [1, 2, 3, 4, 5];
< undefined
> number.reverse();
< ▸ (5) [5, 4, 3, 2, 1]
> console.log(number);
  ▸ (5) [5, 4, 3, 2, 1]
```

▼ Array.toString()

🛠️ X

🔙 배열의 요소를 표현하는 문자열을 반환합니다.

💡 지정된 배열의 요소를 문자열로 나타냅니다.
※ 각 요소는 쉼표로 구분합니다.

```
> const number = [1, 2, 3, 4, 5];
< undefined
> number.toString();
< '1,2,3,4,5'
> const number = ['1', '2', '3', '4', '5'];
< undefined
> number.toString();
< '1,2,3,4,5'
```

▼ Array.join()

▼ Array.sort()

🛠️ `compareFunction (Option)`
정렬 순서를 정의하는 함수.

🔙 오름차순으로 정렬된 배열

💡 배열을 유니코드의 포인트 값에 따라 오름차순으로 정렬시킵니다.
※ 원본 배열을 변형합니다.

```
> const number = [3, 2, 5, 4, 1];
< undefined
> number.sort();
< ▶ (5) [1, 2, 3, 4, 5]
> console.log(number);
  ▶ (5) [1, 2, 3, 4, 5]
```

▼ Array.toLocaleString()

🛠️ `locale (Option)`
Locale Codes를 통해 `Locale` 값을 변경할 수 있다.
ex) ko, ja, en …
※ 생략 시 웹브라우저의 기본 `Locale` 값 사용

`options (Option)`
원하는 속성들을 포함한 객체로 지정할 수 있다.

🔙 배열의 요소를 표현하는 문자열을 반환합니다.

💡 배열의 요소에 toLocaleString을 적용 후 문자열로 반환합니다.
※ 각 요소는 쉼표로 구분합니다.

```
> const arr = [5, '문자', 2000000000, new Date];
< undefined
> console.log(arr.toLocaleString());
  5,문자,2,000,000,000,2023. 4. 3. 오후 9:25:22
```

🛠️ `separator (Option)`
배열의 각 요소들을 구분할 구분자를 지정합니다.
※ 지정하지 않을 시 기본값은 쉼표입니다.

🔙 배열의 모든 요소들을 연결한 하나의 문자열을 반환합니다.

💡 배열의 모든 요소들을 선택한 구분자로 연결하여 하나의 문자열로 만듭니다.

```
> const number = ['1', '2', '3', '4', '5'];
< undefined
> number.join();
< '1,2,3,4,5'
> number.join(':');
< '1:2:3:4:5'
```