6. Array

6-1. 배열 활용방법

6-2. 배열 API

■ Array

● 배열의 필요성

06-01-배열필요성.html

```
const sun = '일';
const mon = '월';
const tue = '화';
const wed = '수';
const thu = '목';
const fri = '금';
const sat = '토';

const date = new Date();
const week = date.getDay();

if(week == 0) console.log(sun);
else if(week == 1) console.log(mon);
// ...
else console.log(sun);
```

```
const sun = '일';
const mon = '월';
const tue = '화';
const wed = '수';
const thu = '목';
const fri = '금';
const sat = '토';

const date = new Date();
const week = date.getDay();

switch(week) {
  case 0: console.log(mon); break;
  case 1: console.log(tue); break;
  // ...
  default: console.log(sun);
}
```

```
const daysOfWeek = [ '일', '월', '화', '수', '목', '금', '토' ];

console.log(daysOfWeek[week]);
```

■ Array

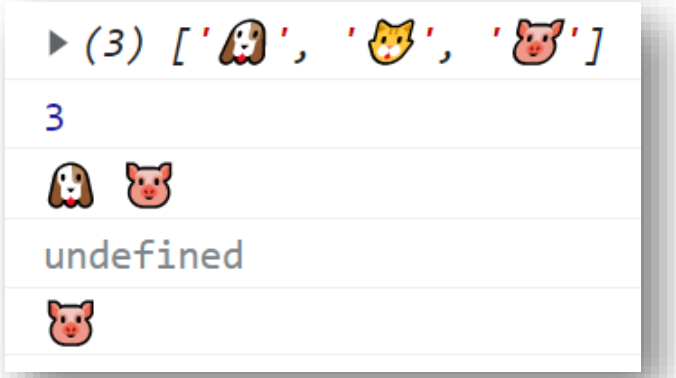
● 배열 초기화

06-02-배열초기화.html

```
const arr1 = new Array();
const arr2 = [1, 2];
```

● 인덱스를 이용한 접근

```
const animals = ['🐶', '🐱', '🐷'];
console.log(animals);
console.log(animals.length);
console.log(animals[0], animals[2]);
console.log(animals[3]); // undefined
console.log(animals[animals.length - 1]); // last
```



■ Array

● 반복문을 이용한 요소 제어

06-03-배열요소접근.html

```
const animals = ['🐶', '🐱', '🐷'];
```

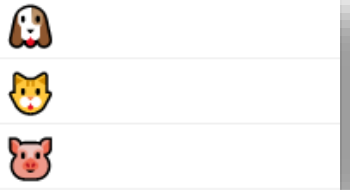
- index(key)를 알아낸 후 value 확인 (for ... in)

```
for(const key in animals) {
  console.log(key);
}
```

```
0
1
2
```

- 모든 value를 순서대로 하나씩 꺼내어 확인 (for ... of)

```
for(const value of animals) {
  console.log(value);
}
```



■ Array

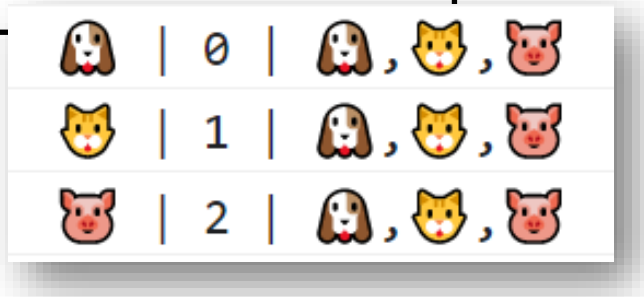
● 반복문을 이용한 요소 제어

06-03-배열요소접근.html

```
const animals = ['🐶', '🐱', '🐷'];
```

- 모든 요소(key/value)를 순서대로 하나씩 꺼내어 확인 (forEach)

```
animals.forEach(function(animal, index, array) {
  console.log(`${animal} | ${index} | ${array}`);
});

animals.forEach((animal, index, array) => {
  console.log(`${animal} | ${index} | ${array}`);
});
```



■ Array

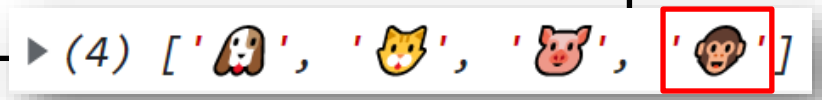
● 배열의 요소 추가

06-04-배열요소추가.html

```
const animals = ['🐶', '🐱', '🐷'];
```

- 배열의 마지막에 요소 추가 (push)

```
animals.push('🐵');
console.log(animals);
```



- 지정된 인덱스에 요소 추가 (이미 요소가 있다면 값 수정)

```
animals[5] = '🐰';
console.log(animals);
```

자동 추가

■ Array

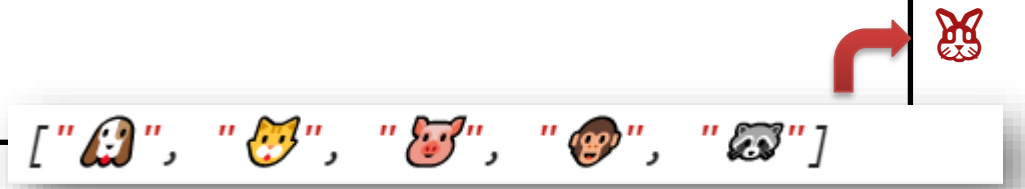
● 배열의 요소 제거

06-05-배열요소삭제.html

```
['🐶', '🐱', '🐷', '🐵', '🦝', '🐰']
```

- 배열의 마지막 요소 삭제 (pop)

```
console.log(animals);
animals.pop();
console.log(animals);
```

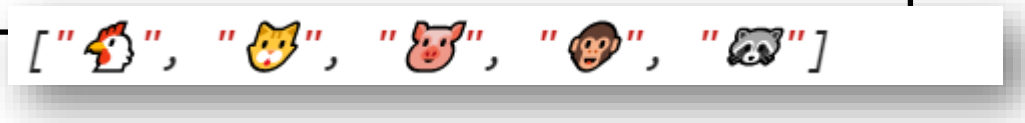


- 배열의 처음 요소 제거 (shift)

```
animals.shift();
console.log(animals);
```

- 배열의 처음에 요소 추가 (unshift)

```
animals.unshift('🐔');
console.log(animals);
```

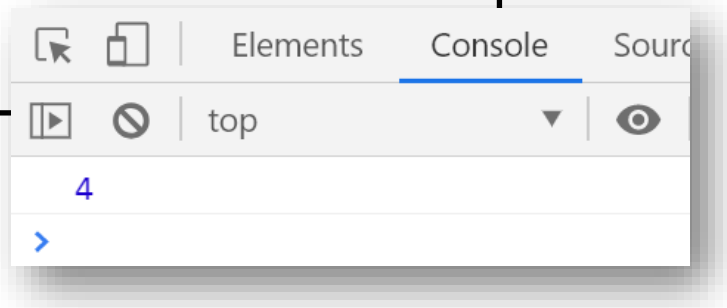


■ 연습문제 (06-연습문제1.html)

● 반복문을 사용하여 배열 요소 중 최대값을 찾는 max() 함수 작성하기

(단, 배열의 요소는 0 보다 큰 정수)

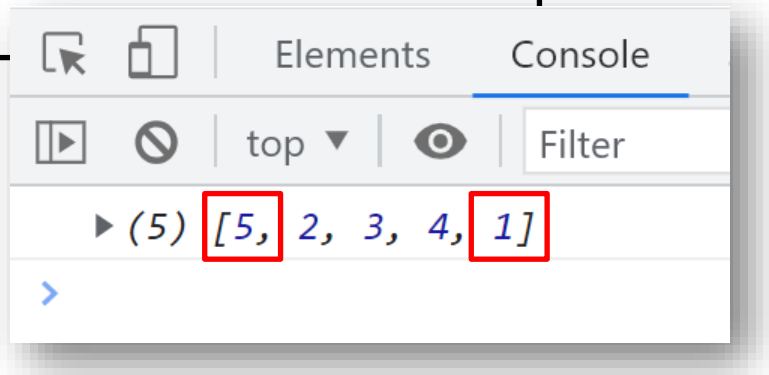
```
const max = (    ①    ) {
  let output = 0;

  (    ②    )

  return output;
};

console.log(max([1, 2, 3, 4]));
```



■ 연습문제 (06-연습문제2.html)

● 처음과 마지막 요소의 위치 변경하기 (pop / shift / push / unshift)

```
const array = [1, 2, 3, 4, 5];
```



■ Array API

● 찾기

- includes : 요소 포함 여부
- indexOf : 요소의 위치
- find : 해당 요소 1개
- filter : 해당하는 요소를 찾은 후 배열 생성
- some : 해당 요소가 1개 이상이면 true
- every : 모든 요소가 해당하면 true

● 기능 적용

- **map** : 모든 요소에 접근하여 제어
- **reduce** : 모든 요소에 접근하여 제어

● 합치기

- join : 배열 ➔ 문자열
- concat : 다수의 배열 ➔ 하나의 배열

● 분리

- split : 문자열 ➔ 배열
- slice : 지정 요소가 삭제된 배열 생성
- splice : 지정 요소가 삭제된 배열 생성

● 정렬

- sort : 요소 값을 기준으로 정렬
- reverse : 인덱스를 기준으로 역순 정렬

■ Array API - 찾기

● includes : 요소 포함 여부

06-06-배열API-includes.html

```
includes(searchElement: T, fromIndex?: number): boolean;
```

```
const array = ['🍕', '🍔', '🍟', '🌭', '🍗'];
console.log(array.includes('🍔')); // true
console.log(array.includes('🍙')); // false
```

● indexOf : 요소의 위치

06-07-배열API-indexOf.html

```
indexOf(searchElement: T, fromIndex?: number): number;
```

```
const array = ['🍕', '🍔', '🍟', '🌭', '🍗'];
console.log(array.indexOf('🍔')); // 1
console.log(array.indexOf('🍙')); // -1
console.log(array.indexOf('🍗')); // 4
```

■ Array API - 찾기

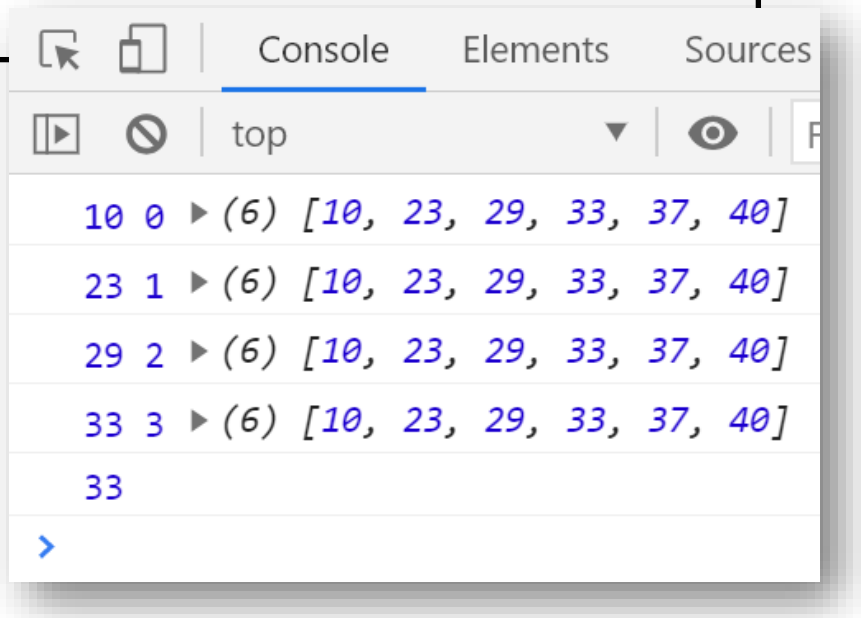
● find : 해당 요소 1개

- 조건에 맞는 경우 첫번째 요소 반환

06-08-배열API-find.html

```
find(predicate: (value: T, index: number, obj: T[])
```

```
const array = [10, 23, 29, 33, 37, 40];
let result = array.find((value, index, arr) => {
  console.log(value, index, arr);
  return value % 3 == 0;
});
console.log(result);
```



■ Array API - 찾기

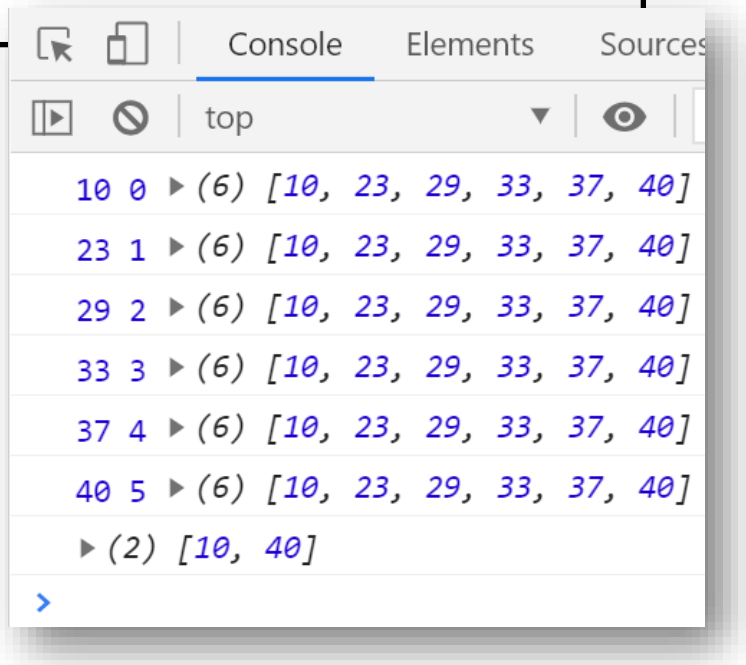
● filter : 해당하는 요소를 찾은 후 배열 생성

- 조건에 맞는 모든 요소로 새로운 배열을 생성

06-09-배열API-find.html

```
filter(predicate: (value: T, index: number, array: T[])
```

```
const array = [10, 23, 29, 33, 37, 40];
let result = array.filter((value, index, arr) => {
  console.log(value, index, arr);
  return value % 2 == 0;
});
console.log(result);
```



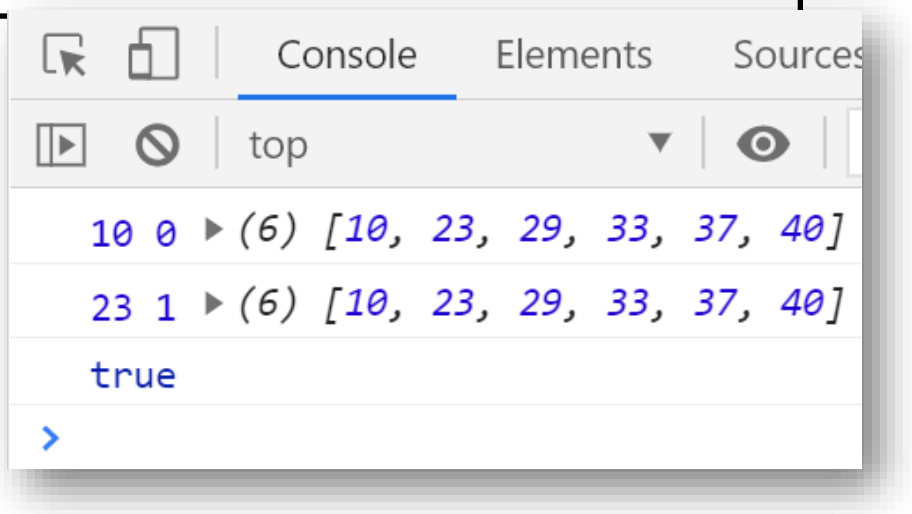
■ Array API - 찾기

● some : 해당하는 요소가 1개 이상이면 true

06-10-배열API-some.html

```
some(predicate: (value: T, index: number, array: T[])
```

```
const array = [10, 23, 29, 33, 37, 40];
let result = array.some((value, index, array) => {
  console.log(value, index, array);
  return value > 20;
});
console.log(result);
```



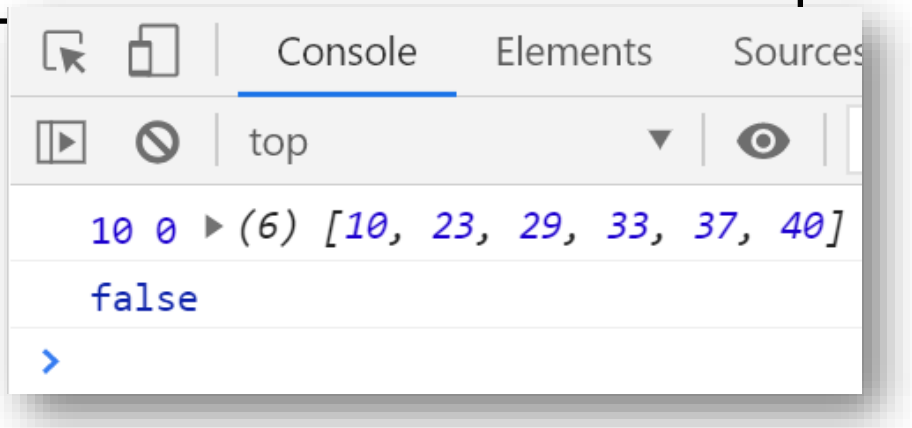
■ Array API - 찾기

● every : 모든 요소가 해당하면 true

06-11-배열API-every.html

```
every(predicate: (value: T, index: number, array: T[])
```

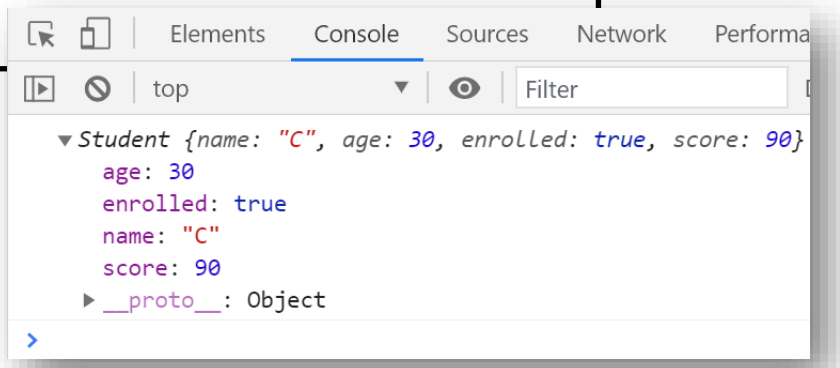
```
const array = [10, 23, 29, 33, 37, 40];
let result = array.every((value, index, array) => {
  console.log(value, index, array);
  return value > 20;
});
console.log(result);
```



■ 연습문제 (06-연습문제3.html)

- 점수가 90점인 1명(첫번째)의 학생 정보 찾아내기 (find)

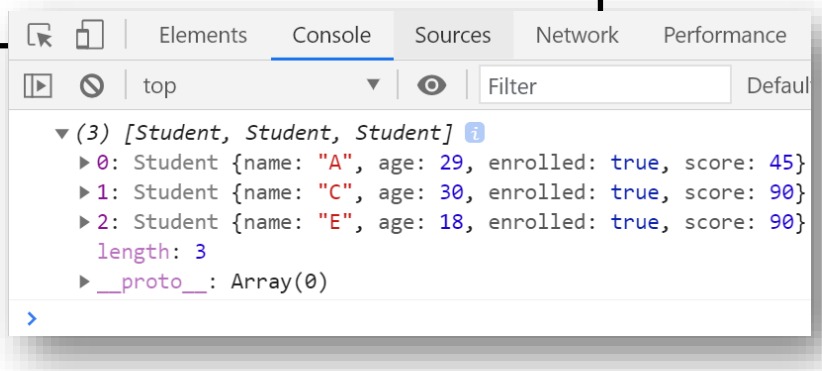
```
class Student {
  constructor(name, age, enrolled, score) {
    this.name = name;
    this.age = age;
    this.enrolled = enrolled;
    this.score = score;
  }
}
const students = [
  new Student('A', 29, true, 45),
  new Student('B', 28, false, 80),
  new Student('C', 30, true, 90),
  new Student('D', 40, false, 66),
  new Student('E', 18, true, 90),
];
```



■ 연습문제 (06-연습문제4.html)

● 수업에 등록(enrolled)되어 있는 모든 학생의 정보 찾아내기 (filter)

```
class Student {
  constructor(name, age, enrolled, score) {
    this.name = name;
    this.age = age;
    this.enrolled = enrolled;
    this.score = score;
  }
}
const students = [
  new Student('A', 29, true, 45),
  new Student('B', 28, false, 80),
  new Student('C', 30, true, 90),
  new Student('D', 40, false, 66),
  new Student('E', 18, true, 90),
];
```



■ 연습문제 (06-연습문제5.html)

● 점수가 50점 미만인 학생이 있는지 확인하기 (some)

```
class Student {
  constructor(name, age, enrolled, score) {
    this.name = name;
    this.age = age;
    this.enrolled = enrolled;
    this.score = score;
  }
}
const students = [
  new Student('A', 29, true, 45),
  new Student('B', 28, false, 80),
  new Student('C', 30, true, 90),
  new Student('D', 40, false, 66),
  new Student('E', 18, true, 90),
];
```

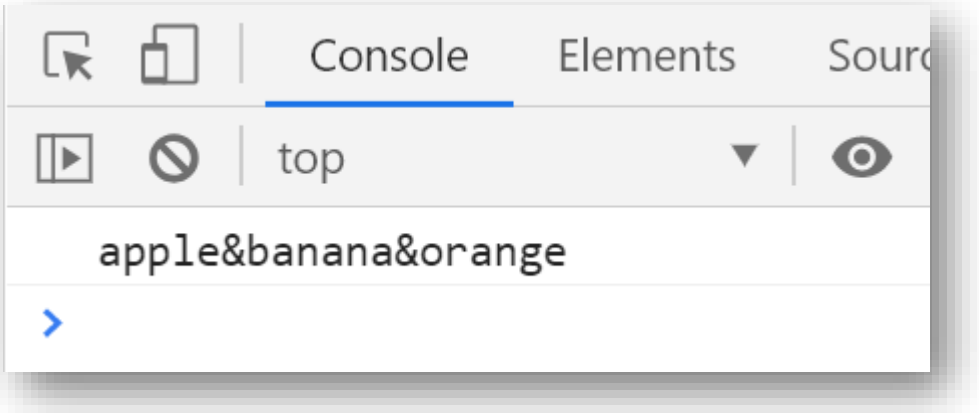
■ Array API - 합치기

● join : 배열 ➔ 문자열

06-12-배열API-join.html

```
join(separator?: string)
```

```
const fruits = ['apple', 'banana', 'orange'];
const result = fruits.join('&');
console.log(result);
```



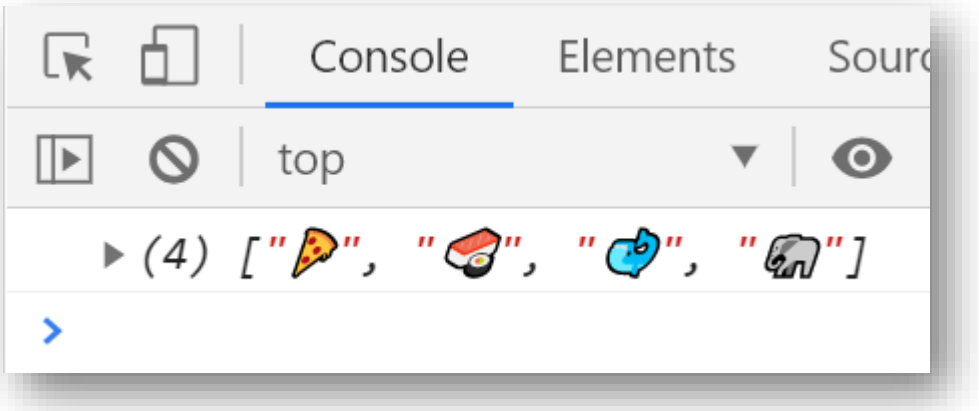
■ Array API - 합치기

● concat : 다수의 배열 ➔ 하나의 배열

06-13-배열API-concat.html

```
concat(...items: ConcatArray<T>[])
```

```
const foods = ['🍕', '🍣'];
const animals = ['🐔', '🐘'];
const newArray = foods.concat(animals);
console.log(newArray);
```



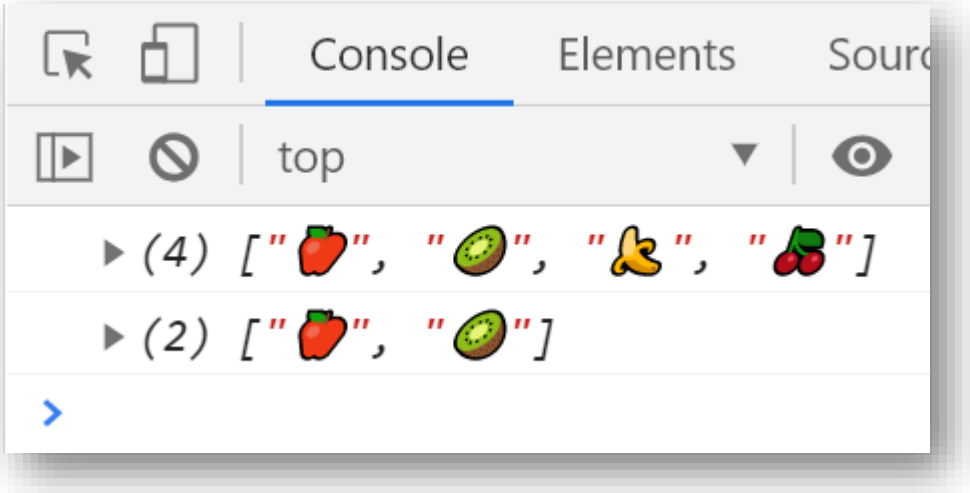
■ Array API - 분리

● split : 문자열 ➔ 배열

06-14-배열API-split.html

```
split(string: string, limit?: number)
```

```
const fruits = '🍎, 🥝, 🍌, 🍒';
const result = fruits.split(', ')
console.log(result);
const result2 = fruits.split(', ', 2)
console.log(result2);
```



■ Array API - 분리

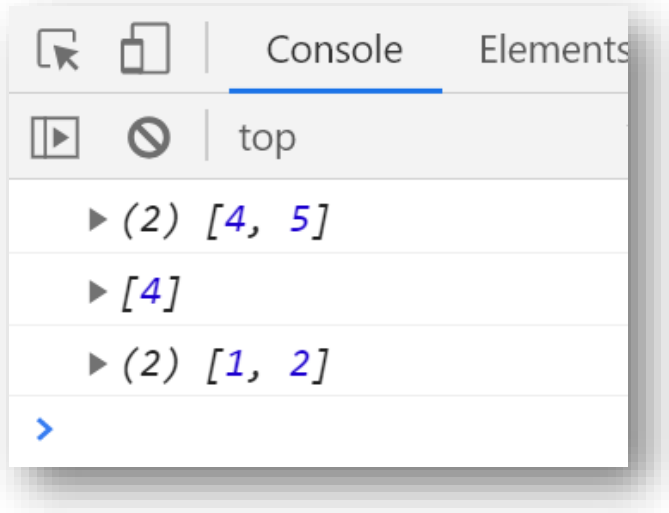
● slice : 지정 요소가 삭제된 배열 생성

- 시작 인덱스와 끝 인덱스를 사이의 요소로 새로운 배열 생성
- 원본 데이터 유지 (array 변함 없음)

06-15-배열API-slice.html

```
slice(start?: number, end?: number)
```

```
const array = [1, 2, 3, 4, 5];
const result = array.slice(3)
console.log(result); // [4, 5]
const result2 = array.slice(3, 4)
console.log(result2); // [4]
const result3 = array.slice(0, 2)
console.log(result3); // [1, 2]
```



■ Array API - 분리

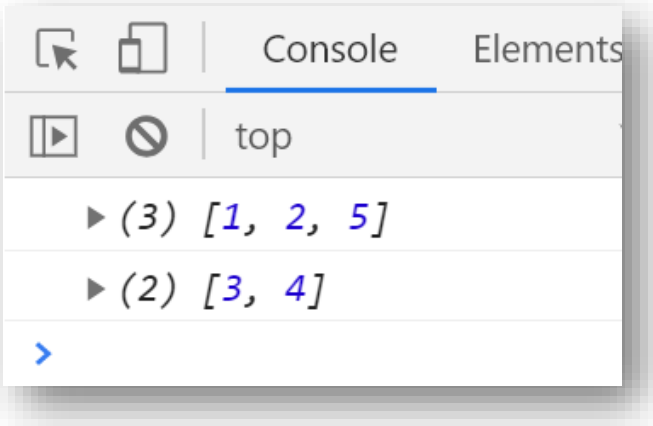
● splice : 지정 요소가 삭제된 배열 생성

- 시작 인덱스와 분리(삭제)시킬 요소의 개수 지정으로 새로운 배열 생성
- 원본 데이터 수정

06-16-배열API-splice.html

```
splice(start: number, deleteCount?: number)
```

```
const array = [1, 2, 3, 4, 5];
const result = array.splice(2, 2)
console.log(array); // [1, 2, 5] - 원본 데이터 수정
console.log(result); // [3, 4] - 2번 인덱스 ~ 요소 2개
```

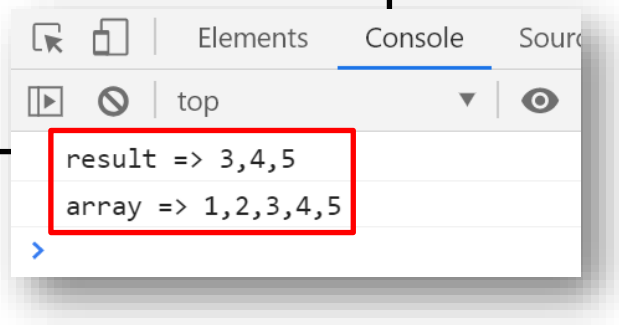


■ 연습문제 (06-연습문제6.html)

● 주어진 배열 요소 중 첫번째와 두번째 요소를 제외한 새로운 배열 생성하기

- 원본 데이터 유지 (slice)

```
const array = [1, 2, 3, 4, 5];
const result = (    ①    )

console.log(`result => ${result}`);
console.log(`array => ${array}`);
```

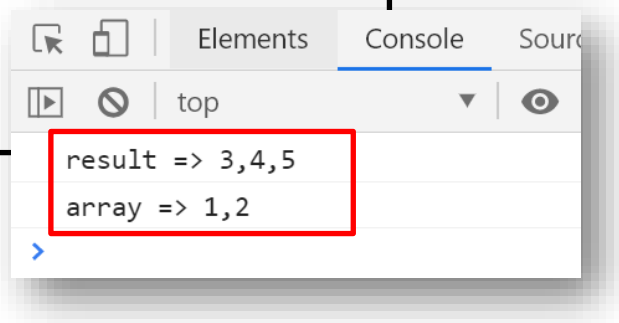


■ 연습문제 (06-연습문제7.html)

● 주어진 배열 요소 중 첫번째와 두번째 요소를 제외한 새로운 배열 생성하기

- 원본 데이터 수정 (splice)

```
const array = [1, 2, 3, 4, 5];
const result = (    ①    )

console.log(`result => ${result}`);
console.log(`array => ${array}`);
```



■ Array API - 정렬

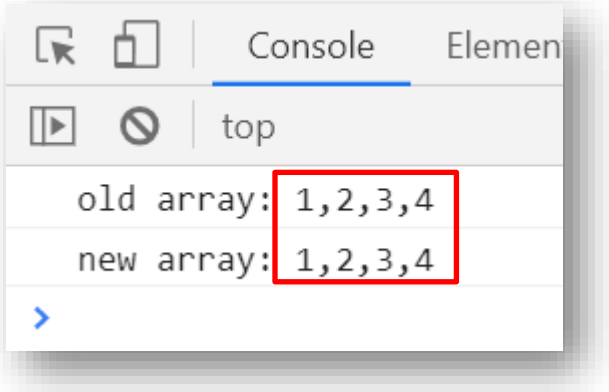
● sort : 요소 값을 기준으로 정렬

- 원본 데이터를 정렬하고, 새로운 배열도 생성

06-17-배열API-sort.html

```
sort(compareFn?: (a: T, b: T) => number)
```

```
const array = [1, 3, 4, 2];
const newArray = array.sort();
console.log(`old array: ${array}`);
console.log(`new array: ${newArray}`);
```



■ Array API - 정렬

● sort : 요소 값을 기준으로 정렬

- 비교 연산으로 역순 정렬

06-18-배열API-sort-custom.html

```
sort(compareFn?: (a: T, b: T) => number)
```

```
const orderArray = [1, 3, 4, 2];
orderArray.sort(function (next, prev) {
  console.log(`next: ${next}, prev: ${prev}`);
  console.log(`prev - next = ${prev - next}`);
  return prev - next;
});
console.log(`ordered array: ${orderArray}`);
```

(*) 비교 연산 결과가 음수인 경우 위치 변경

```
next: 3, prev: 1
prev - next = -2
next: 4, prev: 3
prev - next = -1
next: 2, prev: 4
prev - next = 2
next: 2, prev: 3
prev - next = 1
next: 2, prev: 1
prev - next = -1
ordered array: 4,3,2,1
```

■ Array API - 정렬

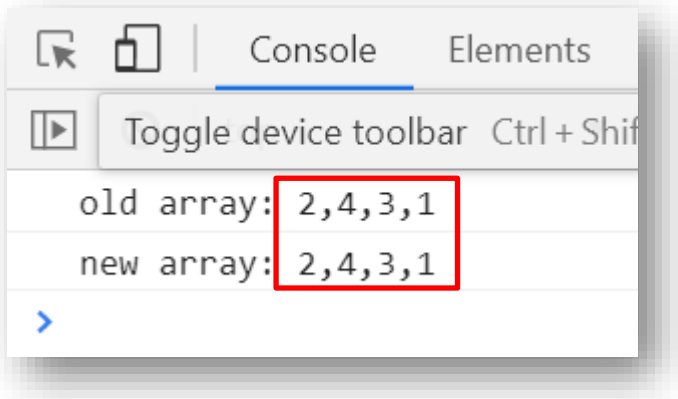
● reverse : 인덱스를 기준으로 역순 정렬

- 원본 데이터를 정렬하고, 새로운 배열도 생성

06-19-배열API-reverse.html

```
reverse()
```

```
const array = [1, 3, 4, 2];
const newArray = array.reverse();
console.log(`old array: ${array}`);
console.log(`new array: ${newArray}`);
```



■ 연습문제 (06-연습문제8.html)

● 정렬 연습문제

```
const array = [6, 7, 8, 9, 10, 1, 2, 3, 4, 5];

const newArray = array.sort((next, prev) => {

  // 홀수 오름차순, 짝수 내림차순 정렬

});
console.log(`${newArray}`);
```

```
1,3,5,7,9,10,8,6,4,2
```

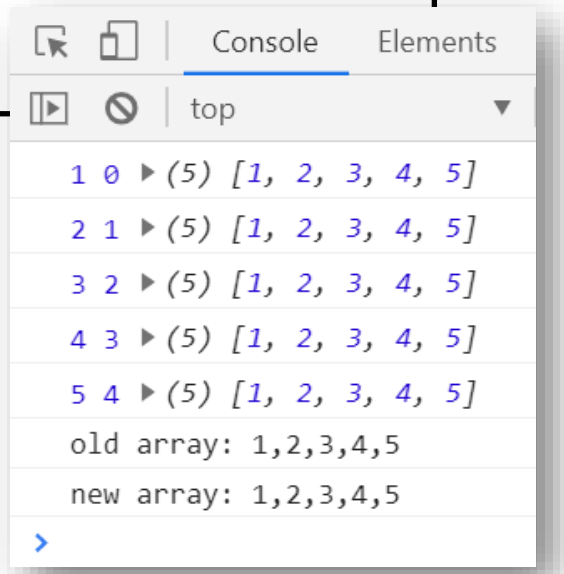
■ Array API - 기능 적용

● map : 모든 요소에 접근하여 제어

06-20-배열API-map.html

```
map<U>(callbackfn: (value: T, index: number, array: T[])
```

```
const arr = [1, 2, 3, 4, 5];

const newArr = arr.map(function(value, idx, arr) {
  console.log(value, idx, arr);
  return value;
});

console.log(`old array: ${arr}`);
console.log(`new array: ${newArr}`);
```



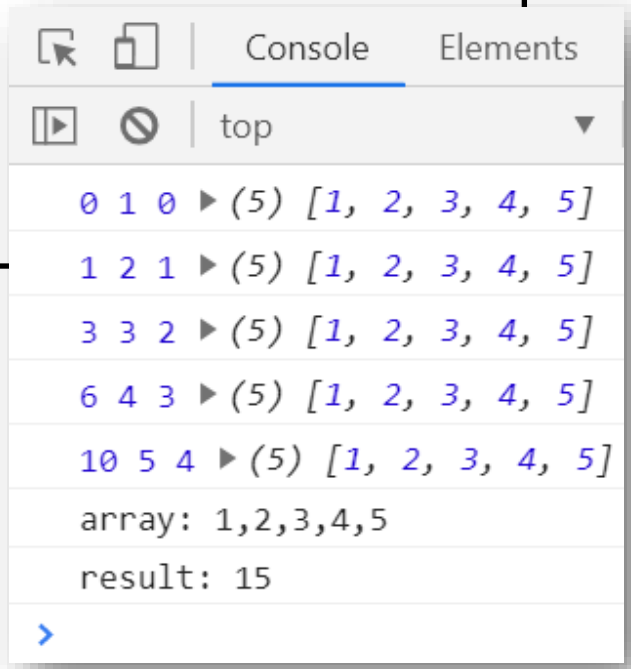
■ Array API - 기능 적용

● reduce : 모든 요소에 접근하여 제어

06-21-배열API-reduce.html

```
reduce(callbackfn: (previousValue: number, currentValue: number, currentIndex: number, array: number[])
```

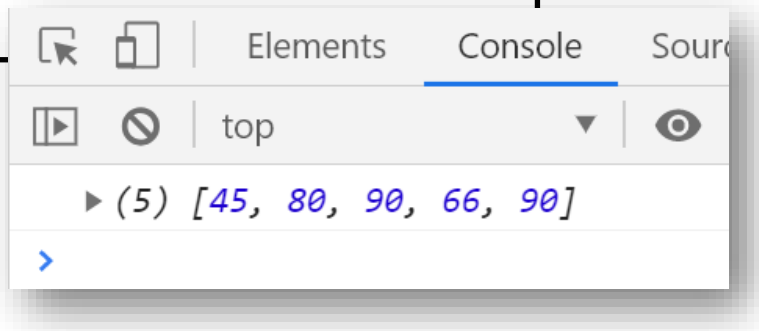
```
const arr = [1, 2, 3, 4, 5];

const result = arr.reduce(function(pv, cv, idx, arr) {
  console.log(pv, cv, idx, arr);
  return pv + cv;
});

console.log(`array: ${arr}`);
console.log(`result: ${result}`);
```



■ 연습문제 (06-연습문제9.html)

● 학생들의 점수만 요소로 가지는 새로운 배열 생성하기

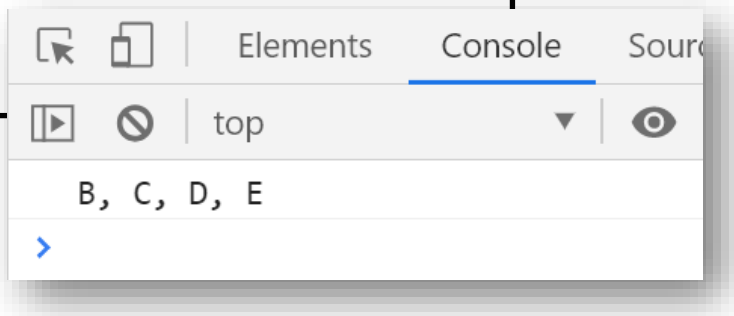
```
class Student {
  constructor(name, age, enrolled, score) {
    this.name = name;
    this.age = age;
    this.enrolled = enrolled;
    this.score = score;
  }
}
const students = [
  new Student('A', 29, true, 45),
  new Student('B', 28, false, 80),
  new Student('C', 30, true, 90),
  new Student('D', 40, false, 66),
  new Student('E', 18, true, 90),
];
```



■ 연습문제 (06-연습문제10.html)

● 점수가 50점 이상인 학생의 이름을 결과와 같이 문자열로 출력하기 (filter / map / join)

```
class Student {
  constructor(name, age, enrolled, score) {
    this.name = name;
    this.age = age;
    this.enrolled = enrolled;
    this.score = score;
  }
}
const students = [
  new Student('A', 29, true, 45),
  new Student('B', 28, false, 80),
  new Student('C', 30, true, 90),
  new Student('D', 40, false, 66),
  new Student('E', 18, true, 90)
];
```



■ 연습문제 (06-연습문제11.html)

● 모든 학생의 점수 합과 평균 구하기

- reduce(callbackfn, **initValue**) : **초기값을 0으로 지정**

```
const result = students.reduce(    ①    , 0);

console.log(`합 => ${result}`);
console.log(    ②    );
```