KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

배열

[KB] IT's Your Life

KB 국민은행

배열

배열

- 여러 값을 하나의 변수로 접근할 수 있는 객체
- 값은 특정 위치 정로로 관리함
- 특정 위치 → index

- length 속성
  - 배열의 길이 값
  - for 루프 순회 시 종료 조건에 사용

배열

배열 생성자

생성 함수	설명
Array()	빈 배열을 만든다
Array(n)	크기가 n인 배열을 만든다
Array(a, b, c, …)	인수로 전달된 요소를 가지는 배열을 만든다.
Array.of(a, b, c, …)	인수로 전달된 요소를 가지는 배열을 만든다.
Array.of(순회가능 객체[, 함수])	순환 가능한 객체의 값을 배열로 만든다. 함수가 제공되는 경우, 기존 값을 인자로하며 리턴값을 배열의 요소로 만든다.

리터럴을 통한 생성

- const array = [ ] // 비어 있는 배열
- const arrary = [1, 2, 5, 9, 15]; // 요소를 가지는 배열

intarray.js

```
const ar = [1, 2, 5, 9, 15];
//const ar = new Array(1, 2, 5, 9, 15);

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] =' + ar[i]);
}
```

```
ar[0] = 1
ar[1] = 2
ar[2] = 5
ar[3] = 9
ar[4] = 15
```

arraylength.js

```
const ar = ['태연', '유리', '윤아', '써니'];
for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] = ' + ar[i]);
}
```

```
ar[0] = 태연
ar[1] = 유리
ar[2] = 윤아
ar[3] = 써니
```

arraymix.js

```
const ar = [1234, '문자열', true, { name: '김상형', age: 29 }];

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] = ' + ar[i]);
}
```

```
ar[0] = 1234
ar[1] = 문자열
ar[2] = true
ar[3] = [object Object]
```

dynamiclength.js

```
const ar = [0, 1, 2, 3];
ar[6] = 6;

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] = ' + ar[i]);
}
console.log('ar[' + 100 + '] = ' + ar[100]);
```

```
ar[0] = 0
ar[1] = 1
ar[2] = 2
ar[3] = 3
ar[4] = undefined
ar[5] = undefined
ar[6] = 6
ar[100] = undefined
```

sparsearray.js

```
const ar = [0, 1, 2, 3, , , 6];

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] = ' + ar[i]);
}
```

```
ar[0] = 0
ar[1] = 1
ar[2] = 2
ar[3] = 3
ar[4] = undefined
ar[5] = undefined
ar[6] = 6
```

deleteitem.js

```
const ar = [0, 1, 2, 3];

delete ar[2];

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] = ' + ar[i]);
}
```

```
ar[0] = 0
ar[1] = 1
ar[2] = undefined
ar[3] = 3
```

changelength.js

```
let ar = [0, 1];
ar.length = 5;

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] =' + ar[i]);
}
console.log('-------------');

ar = [0, 1, 2, 3, 4, 5, 6, 7];
ar.length = 3;

for (let i = 0; i < ar.length; i++) {
  console.log('ar[' + i + '] =' + ar[i]);
}
```

```
ar[0] =0
ar[1] =1
ar[2] =undefined
ar[3] =undefined
ar[4] =undefined
-------------
ar[0] =0
ar[1] =1
ar[2] =2
```

stringindex.js

```
const ar = [0, 1, 2, 3];
console.log('ar[1] = ' + ar[1]);

ar['korea'] = 4;
console.log('ar["korea"] = ' + ar['korea']);
console.log('ar.korea = ' + ar.korea);

ar[-3.14] = 5;
console.log('ar[3.14] = ' + ar['-3.14']);
```

```
ar[1] = 1
ar["korea"] = 4
ar.korea = 4
ar[3.14] = 5
```

배열

배열의 순회

- 배열의 인덱스 정보는 필요 없고 값만 사용하는 경우

```
for(let 변수 of 배열) {     // 변수에 순회하는 값을 저장
  명령;
}
```

- 순회할 때 배열의 인덱스를 자동 설정

```
for(let 변수 in 배열) {     // 변수에 순회하는 인덱스를 저장
  명령;
}
```

→ 중간의 요소 값이 undefined인 경우 건너 뜀

forin.js

```
const arScore = [88, 78, 96, 54, 23];

for (let score of arScore) {
  console.log('학생의 성적 : ' + score);
}
console.log('----------------------------');
for (let st in arScore) {
  console.log(`${st}번째 학생의 성적 : ${arScore[st]}`);
}
```

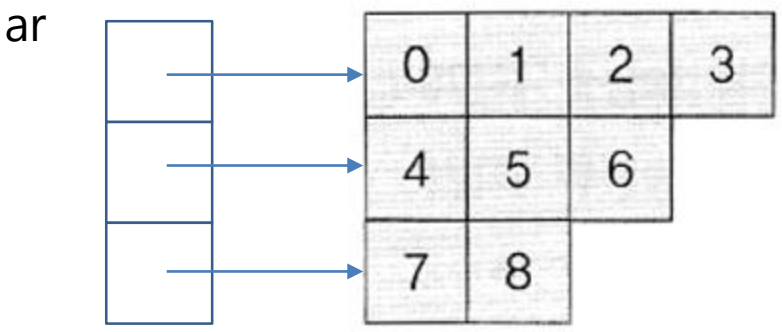
```
학생의 성적 : 88
학생의 성적 : 78
학생의 성적 : 96
학생의 성적 : 54
학생의 성적 : 23
----------------------------
0번째 학생의 성적 : 88
1번째 학생의 성적 : 78
2번째 학생의 성적 : 96
3번째 학생의 성적 : 54
4번째 학생의 성적 : 23
```

forin2.js

```
const arScore = [88, 78, 96, 54, 23];
delete arScore[2];
arScore['반장'] = 100;

for (let st in arScore) {
  console.log(`${st}번째 학생의 성적 : ${arScore[st]}`);
}
console.log('----------------------');
for (let st = 0; st < arScore.length; st++) {
  console.log(`${st}번째 학생의 성적 : ${arScore[st]}`);
}
```

```
0번째 학생의 성적 : 88
1번째 학생의 성적 : 78
3번째 학생의 성적 : 54
4번째 학생의 성적 : 23
반장번째 학생의 성적 : 100
----------------------
0번째 학생의 성적 : 88
1번째 학생의 성적 : 78
2번째 학생의 성적 : undefined
3번째 학생의 성적 : 54
4번째 학생의 성적 : 23
```

2차원 배열



배열의 배열

```
const ar = [ ];
ar[0] = [ 0, 1, 2, 3 ];
ar[1] = [ 4, 5, 6 ];
ar[2] = [ 7, 8 ];
```

```
const ar = [
    [ 0, 1, 2, 3 ],
    [ 4, 5, 6 ],
    [ 7, 8 ]
];
```

nestarray.js

```
const ar = [
  [0, 1, 2, 3],
  [4, 5, 6],
  [7, 8],
];

for (let i = 0; i < ar.length; i++) {
  for (let j = 0; j < ar[i].length; j++) {
    console.log('ar[' + i + '][' + j + '] =' + ar[i][j]);
  }
  console.log('------------------');
}
```

```
ar[0][0] =0
ar[0][1] =1
ar[0][2] =2
ar[0][3] =3
------------------
ar[1][0] =4
ar[1][1] =5
ar[1][2] =6
------------------
ar[2][0] =7
ar[2][1] =8
------------------
```

배열

유사 배열

- 객체의 속성 명으로 인덱스(숫자)를 사용하여 값을 배정하고,
- length 속성을 가지는 객체

→ 객체지만 배열처럼 운영할 수 있으나 배열이 아니기 때문에 배열의 메서드는 호출할 수 없음

arraylike.js

```
const human = {
  name: '김상형',
  age: 29,
};
human[0] = 87;
human[1] = 79;
human[2] = 92;
human.length = 3;

for (let i = 0; i < human.length; i++) {
  console.log(`human[${i}] = ${human[i]}`);
}
console.log('--------------');

for (let i in human) {
  console.log(`human[${i}] = ${human[i]}`);
}
```

```
human[0] = 87
human[1] = 79
human[2] = 92
--------------
human[0] = 87
human[1] = 79
human[2] = 92
human[name] = 김상형
human[age] = 29
human[length] = 3
```

KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

배열 메서드

[KB] IT's Your Life

KB 국민은행

배열 메서드

배열 메서드

메서드	설명
indexOf(item, start)	배열에서 요소를 찾아 위치를 리턴한다.
lastIndexOf(item, start)	역순으로 요소를 찾아 위치를 리턴한다.
push(a,b,c,....)	배열 끝에 요소를 추가한다.
pop()	마지막 요소를 제거하고 리턴한다.
shift()	배열 처음의 원소를 제거하고 리턴한다.
unshift(a,b,c,....)	배열 처음에 요소를 추가한다.
reverse()	배열을 거꾸로 뒤집는다.
sort(sortfunction)	배열을 정렬한다. 인수로 값을 비교하는 함수를 지정할 수 있으며 생략시 사전순으로 정렬된다.
slice(start, end)	start~end 범위의 요소를 따로 떼어내어 새로운 배열을 만든다.
splice(index,n,a, b, c, ...)	배열 일부를 수정한다. 일정 범위를 삭제하고 새로운 요소를 삽입한다.
concat(a,b,c,....)	여러 개의 배열을 합친다.
join(deli)	배열 요소를 하나의 문자열로 합친다. 구분자를 지정할 수 있으며 생략시 콤마로 구분한다.

배열 메서드

- **.indexOf(value [,start])**
  - 값을 지정한 경우 해당 값과 일치하는 첫 번째 요소의 인덱스를 리턴
  - start를 지정하면 지정한 위치에서 부터 검색
  - 일치하는 요소가 없는 경우 -1리턴

- **.indexOf(함수)**
  - 배열의 요소를 매개변수로 전달
  - 전달된 요소가 찾고자 하는 요소인지 검사하여 맞다면 true 리턴
  - 첫 번째로 true를 리턴하는 요소의 인덱스를 리턴

- **.lastIndexOf()**
  - 기능은 indexOf()와 같지만 뒤에서부터 검색

배열 메서드

indexof.js

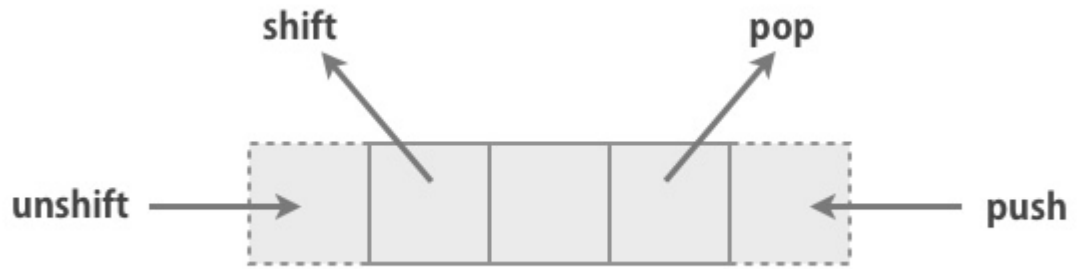
```
const ar = ['태연', '유리', '윤아', '써니', '수영', '유리', '서현', '효연'];
console.log('ar = ' + ar);

const sunny = ar.indexOf('써니');
console.log('써니는 ' + sunny + '번째에 있다.');

let yuri = ar.indexOf('유리');
console.log('유리는 ' + yuri + '번째에 있다.');
yuri = ar.lastIndexOf('유리');
console.log('유리는 뒤에서 ' + yuri + '번째에 있다.');

const suji = ar.indexOf('수지');
if (suji == -1) {
  console.log('수지는 소녀시대가 아니다.');
}
```

```
ar = 태연,유리,윤아,써니,수영,유리,서현,효연
써니는 3번째에 있다.
유리는 1번째에 있다.
유리는 뒤에서 5번째에 있다.
수지는 소녀시대가 아니다.
```

배열 메서드

배열에 요소 추가/삭제하기

- .unshift()
  - 배열의 맨 앞에 값을 추가
- .shift()
  - 배열의 맨 앞에서 값을 제거하고, 해당 값을 리턴
- .push()
  - 배열의 맨 뒤에 값 추가
- .pop()
  - 배열의 맨 뒤에서 값을 제거하고, 해당 값을 리턴



배열 메서드

pushpop.js

```
const ar = [0, 1, 2, 3];

console.log('ar = ' + ar);
ar.push(100, 200);
console.log('ar = ' + ar);
ar.push(300);
console.log('ar = ' + ar);
ar.pop();
console.log('ar = ' + ar);
```

```
ar = 0,1,2,3
ar = 0,1,2,3,100,200
ar = 0,1,2,3,100,200,300
ar = 0,1,2,3,100,200
```

배열 메서드

shiftunshift.js

```
const ar = [0, 1, 2, 3];

console.log('ar = ' + ar);
ar.unshift(100, 200);
console.log('ar = ' + ar);
ar.unshift(300);
console.log('ar = ' + ar);
ar.shift();
console.log('ar = ' + ar);
```

```
ar = 0,1,2,3
ar = 100,200,0,1,2,3
ar = 300,100,200,0,1,2,3
ar = 100,200,0,1,2,3
```

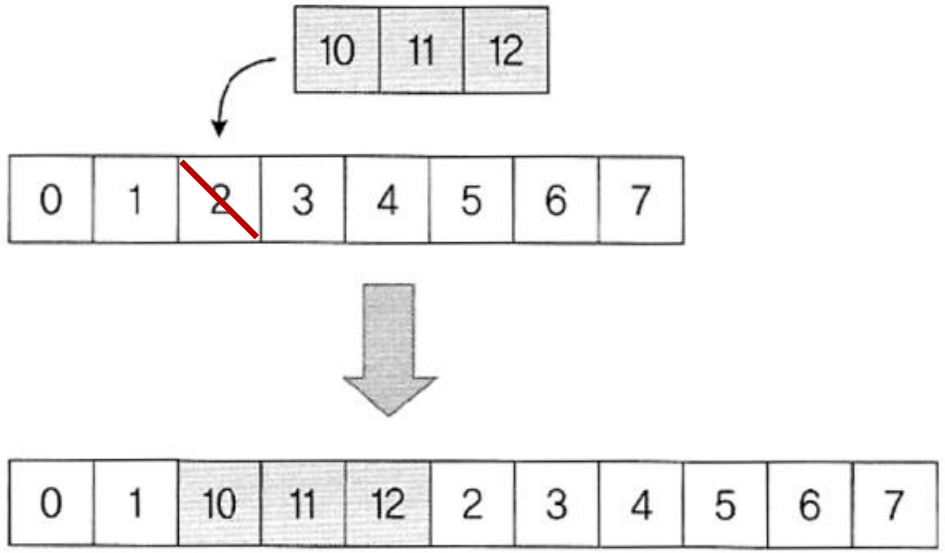
배열 메서드

splice()

- 배열의 중간에 요소를 삭제하거나, 추가함

```
splice(index,howmany,item1,.....,itemX)
```

- **splice(2, 1, 10, 11, 12)**



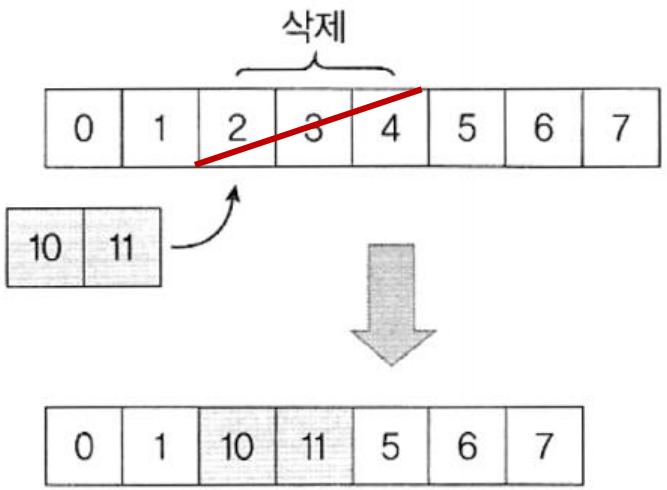
splice(2, 0, a, b, c)는 2번째 위치에서 지우지는 말고 a, b, c를 삽입

- **splice(2, 1)**
  - **2번째 위치에서 1개 요소 제거, 추가는 없음**

배열 메서드

splice()

- splice(2, 3, 10, 11)



배열 메서드

splice.js

```
let ar = [0, 1, 2, 3, 4, 5, 6, 7];
ar.splice(2, 0, 10, 11, 12);
console.log('ar = ' + ar);

// 삭제만 하기
ar = [0, 1, 2, 3, 4, 5, 6, 7];
ar.splice(2, 3);
console.log('ar = ' + ar);

// 삭제 후 삽입 하기
ar = [0, 1, 2, 3, 4, 5, 6, 7];
ar.splice(2, 3, 10, 11);
console.log('ar = ' + ar);
```

```
ar = 0,1,10,11,12,2,3,4,5,6,7
ar = 0,1,5,6,7
ar = 0,1,10,11,5,6,7
```

배열 메서드

- **slice(start, end)**
  - 배열의 부분 집합을 새로운 배열로 리턴
  - start부터 end-1까지 복사(얕은 복사)
  - start만 지정한 경우 끝까지 복사
  - start, end 모두 생략 시 모두 복사

배열 메서드

slice.js

```
const ar = [0, 1, 2, 3, 4, 5, 6, 7];
const subar = ar.slice(2, 5);

console.log('ar = ' + ar);
console.log('subar = ' + subar);
```

```
ar = 0,1,2,3,4,5,6,7
subar = 2,3,4
```

배열 메서드

- **.concat(arr1, arr2, ...)**
  - 배열의 끝에 배열의 내용을 추가하여 새로운 배열 리턴
  - 원본 배열의 내용은 변경없음

배열 메서드

concat.js

```
const ar1 = [0, 1, 2];
const ar2 = [3, 4, 5, 6, 7];
const ar3 = ['수지', '아이유', '김태희'];

const ar4 = ar1.concat(ar2);
console.log('ar1 = ' + ar1);
console.log('ar4 = ' + ar4);

const ar5 = ar1.concat(ar2, ar3);
console.log('ar5 = ' + ar5);
```

```
ar1 = 0,1,2
ar4 = 0,1,2,3,4,5,6,7
ar5 = 0,1,2,3,4,5,6,7,수지,아이유,김태희
```

배열 메서드

- **.join(deli)**
  - 배열의 요소들을 지정한 구분 문자로 결합하여 문자열로 리턴
  - 구분 문자를 생략하면 ',' 문자로 결합

- **.reverse()**
  - 배열의 요소 순서를 반대로 재배치함

→ 원본 배열이 변경됨

배열 메서드

join.js

```
const ar = [0, 1, 2, 3];
console.log('ar = ' + ar.join());
console.log('ar = ' + ar.join(', '));
console.log('ar = ' + ar.join(' -> '));
console.log('ar = ' + ar.toString());
console.log('ar = ' + ar);
```

```
ar = 0,1,2,3
ar = 0, 1, 2, 3
ar = 0 -> 1 -> 2 -> 3
ar = 0,1,2,3
ar = 0,1,2,3
```

배열 메서드

reverse.js

```
const ar = [0, 1, 2, 3];
console.log('ar = ' + ar);

ar.reverse();
console.log('ar = ' + ar);
```

```
ar = 0,1,2,3
ar = 3,2,1,0
```

배열 메서드

.sort([함수])

- 배열의 요소를 정렬함, 원본의 내용이 변경됨
- 주의 사항
  - 요소에는 어떤 타입인지 알 수 없으므로 문자열로 변환한 값을 비교함

- 함수가 매개변수로 전달된 경우
  - 비교 대상이 되는 2개의 요소를 매개변수로 전달
  - 리턴값
    - 0: 두 값이 동일한 경우
    - 양수: 첫 번째 인수가 더 큰 경우
    - 음수: 두 번째 인수가 더 큰 경우

    → a - b의 값을 리턴

배열 메서드

sortarray.js

```
const score = [82, 96, 54, 76, 92, 99, 69, 88];
console.log('before = ' + score);

score.sort();
console.log('after = ' + score);

const names = ['이길동', '홍길동', '박길동', '고길동'];
names.sort();
console.log(names);
```

```
before = 82,96,54,76,92,99,69,88
after = 54,69,76,82,88,92,96,99
[ '고길동', '박길동', '이길동', '홍길동' ]
```

배열 메서드

numbersort.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
console.log('before = ' + score);

score.sort();
console.log('after = ' + score);
```

```
before = 82,96,54,76,9,100,69,88
after = 100,54,69,76,82,88,9,96
```

배열 메서드

sortcompare.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
console.log('before = ' + score);

function compare(left, right) {
  return left - right;
}
score.sort(compare);
console.log('after = ' + score);
```

```
before = 82,96,54,76,9,100,69,88
after = 9,54,69,76,82,88,96,100
```

sortcompare.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
console.log('before = ' + score);

score.sort((left, right) => left - right);
console.log('after = ' + score);
```

```
before = 82,96,54,76,9,100,69,88
after = 9,54,69,76,82,88,96,100
```

배열 메서드

descending.js

```
const score = [82, 96, 54, 76, 92, 99, 69, 88];
console.log('before = ' + score);

score.sort();
score.reverse();
console.log('after = ' + score);
```

```
before = 82,96,54,76,92,99,69,88
after = 99,96,92,88,82,76,69,54
```

배열 메서드

casesort.js

```
const country = ['korea', 'USA', 'Japan', 'China'];
console.log('before = ' + country);

country.sort();
console.log('after = ' + country);
```

```
before = korea,USA,Japan,China
after = China,Japan,USA,korea
```

배열 메서드

casesort2.js

```
const country = ['korea', 'USA', 'Japan', 'China'];
console.log('before = ' + country);

country.sort((left, right) => {
  const left2 = left.toLowerCase();
  const right2 = right.toLowerCase();

  if (left2 < right2) return -1;
  if (left2 > right2) return 1;
  return 0;
});
console.log('after = ' + country);
```

```
before = korea,USA,Japan,China
after = China,Japan,korea,USA
```

배열 메서드

.foreach(함수)

- 내부적으로 배열 전체를 순회하면서 지정한 함수를 호출
- 함수의 매개변수
  - item : 이번에 순회하는 요소
  - index: 해당 요소의 인덱스
  - array: 전체 배열 참조

배열 메서드

foreach.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
let sum = 0;
for (let i = 0; i < score.length; i++) {
  sum += score[i];
}
console.log('sum  = ' + sum);
```

```
sum  = 574
```

foreach2.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
let sum = 0;

score.forEach((value) => (sum += value));
console.log('sum  = ' + sum);
```

```
sum  = 574
```

배열 메서드

.filter(함수)

- 배열을 순회하면서 각 요소에 대해서 함수를 호출
- 해당 요소를 함수의 매개변수로 전달
- 함수가 true를 리턴하는 요소들만 새로운 배열에 담아 리턴

배열 메서드

filter.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
const score2 = score.filter((value) => value >= 80);
console.log('score2  = ' + score2);
```

```
score2  = 82,96,100,88
```

배열 메서드

- **.map(함수)**
  - 배열을 순회하면서 각 요소에 대해서 함수를 호출
  - 해당 요소를 함수의 매개변수로 전달
  - 요소를 가공하여 새로운 값을 리턴
  - 리턴된 새로운 값들을 요소로 가지는 새로운 배열 리턴

배열 메서드

map.js

```
const score = [82, 96, 54, 76, 9, 100, 69, 88];
const score2 = score.map(function (value) {
  return value * 2;
});
console.log('score2  = ' + score2);
```

```
score2  = 164,192,108,152,18,200,138,176
```