

## 06

## 배열

- 배열의 개념
- 배열과 반복문
- 배열 메서드

## ❖ 배열(Array)

- 하나의 변수에 여러 값을 저장할 수 있는 객체 자료형(object)
- 자료 관리를 쉽게 해 주는 객체

seasons				
인덱스 (index)	0	1	2	3
값 (element)	봄	여름	가을	겨울

```
① let seasons = [ '봄', '여름', '가을', '겨울'];  
② let seasons = new Array('봄', '여름', '가을', '겨울');  
console.log(seasons);  
console.log(seasons.length); --- 4  
console.log(seasons[2]); --- 가을
```

## ❖ 배열(Array) 과 반복문

### for 반복문

- 배열의 길이를 이용


```
let seasons = new Array('봄', '여름', '가을', '겨울');  
for (let i = 0 ; i < seasons.length; i++){  
    console.log(seasons[i]);    배열 객체 속성  
}
```

```
for (let i = seasons.length-1 ; i >=0; i--){  
    console.log(seasons[i]);  
}
```

## ❖ 배열(Array) 과 반복문

### for in 반복문

- 배열의 인덱스 번호 만큼 반복 실행

```
let seasons = new Array('봄', '여름', '가을', '겨울');  
for (let i = 0 ; i < seasons.length; i++){  
    console.log(seasons[i]);  
}  
  
for( let  i in seasons){  
    console.log(seasons[i]);  
}
```

## ❖ 배열(Array) 과 반복문

### for of 반복문

- 배열의 요소를 (인덱스 번호 없이) 바로 활용할 수 있는 반복문

```
for( let i in seasons){  
  console.log( seasons[i]);  
}
```

```
for( const element of seasons){  
  console.log(element);  
}
```

## ❖ 배열(Array) 메서드

### concat() : 둘 이상의 배열 연결

- 둘 이상의 배열을 연결하여 새로운 배열
- 기존의 배열 그대로 유지

```
let colors = [ '연두', '파랑', '갈색', '흰색'];  
let seasons = new Array('봄', '여름', '가을', '겨울');  
let concats = seasons.concat(colors)  
concats = concats.concat(seasons);  
console.log(concats);  
console.log(concats.length);  
console.log(concats[2]);
```

## ❖ 배열(Array) 메서드

### join() : 배열의 요소 연결

- 배열의 요소를 지정한 구분기호로 나열

```
let colors = [ '연두', '파랑', '갈색', '흰색'];  
console.log( colors.join()) --- 기본 콤마(,)로 구분  
console.log( colors.join(' : '))
```

연두,파랑,갈색,흰색

연두 : 파랑 : 갈색 : 흰색

### reverse( ) : 배열의 요소 순서 거꾸로 뒤집기

- 배열 자신의 위치가 변화

```
console.log(seasons.reverse());  
console.log(seasons);
```

▶ (4) ['겨울', '가을', '여름', '봄']

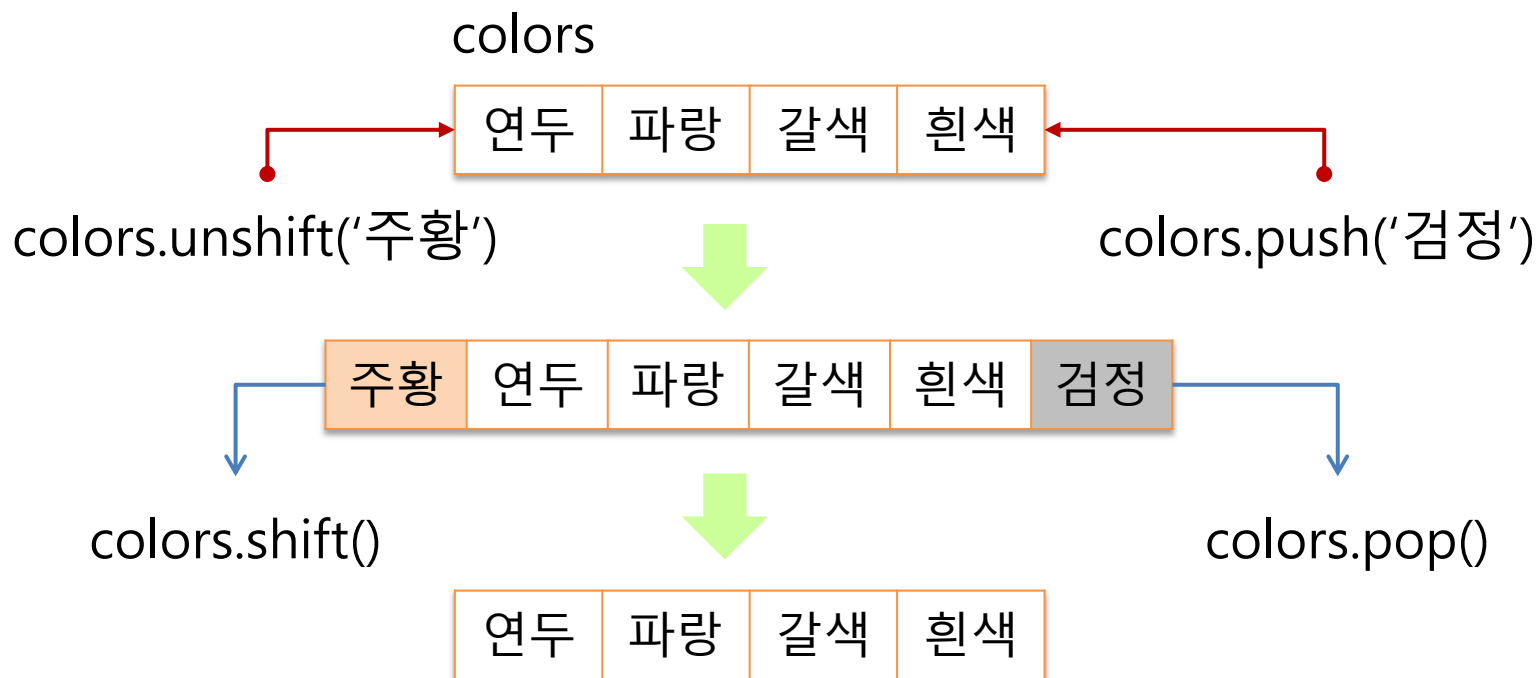
▶ (4) ['겨울', '가을', '여름', '봄']

## ❖ 배열(Array) 메서드

`push()`, `unshift()`: 새로운 요소 추가

`pop()`, `shift()` : 배열 요소 추출(제거)

- 요소 추가와 추출 후 기존의 배열이 바뀜





## ❖ 배열(Array) 메서드

### splice(시작 번호, 삭제 개수, 추가 요소)

- 원하는 위치에 요소 추가 제거, 기존의 배열이 변화됨

colors

[0]	[1]	[2]	[3]	[4]	[5]
주황	연두	파랑	갈색	흰색	검정



colors

[0]	[1]
주황	연두

splice(2)

[0]	[1]	[2]	[3]	[4]	[5]
주황	연두	파랑	갈색	흰색	검정



[0]	[1]	[2]
주황	연두	검정

splice(2, 3)

[0]	[1]	[2]	[3]	[4]	[5]
주황	연두	파랑	갈색	흰색	검정



[0]	[1]	[2]
주황	노랑	검정

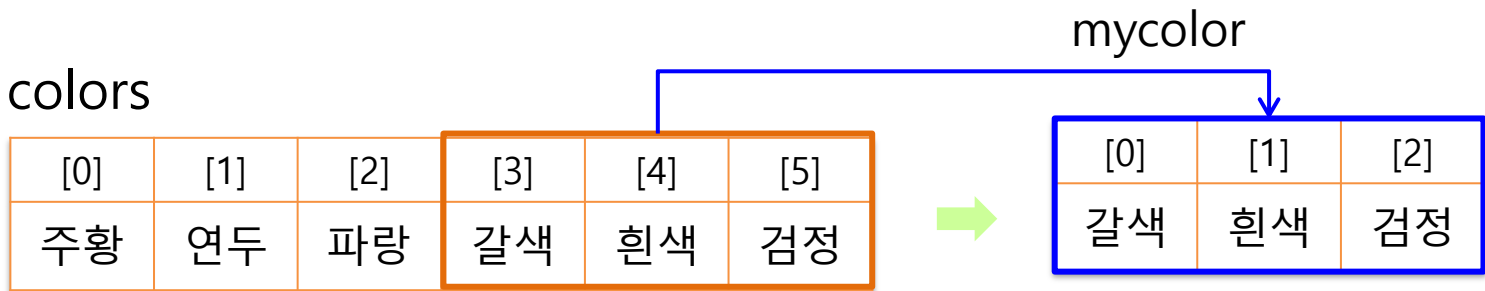
splice(1, 4, '노랑')

'노랑'

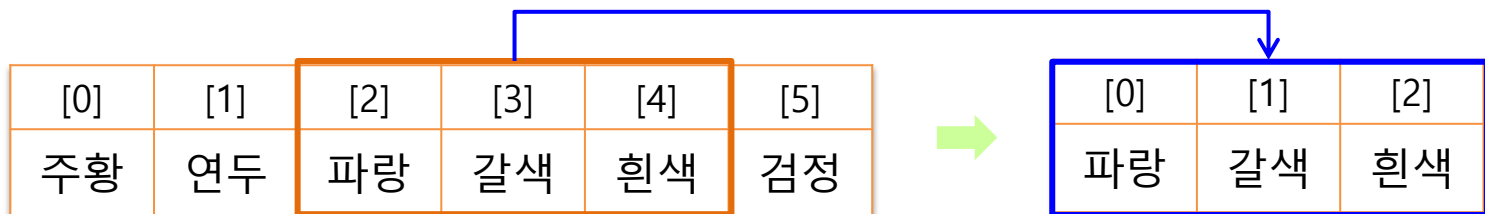
## ❖ 배열(Array) 메서드

### slice(시작 번호, 끝 번호)

- 시작 위치에서 끝 번호 사이의 요소 추출(끝 번호 -1 까지)
- 기존의 배열 그대로 유지, 추출해서 새로운 배열을 만듦



slice(3)



slice(2, 5)

## ❖ 배열(Array) 메서드

**Array.isArray() : 배열인지 확인**

- true/false 리턴

```
console.log(Array.isArray(colors));
```

**indexOf() : 앞에서 부터 검색, 요소의 인덱스 리턴, 없으면 -1**

**lastIndexOf() : 뒤에서 부터 검색**

```
let colors = [ '연두', '파랑', '갈색', '흰색'];  
console.log(colors.indexOf('파랑'));  
console.log(colors.indexOf('빨강'))  
console.log(colors.lastIndexOf('파랑'))
```

## ❖ 배열(Array) 메서드

### forEach() : 배열 요소 사용한 함수 활용

- 배열.forEach(function(배열요소, 인덱스, 배열){});

```
colors.forEach(function(element, index, colors){  
    console.log(`index : ${index}, element : ${element} of colors \n`);  
});
```

### map() : 기존 배열을 이용해 새로운 배열 생성

```
let array = [1, 2, 3, 4, 5];  
let array2 = array.map(function(element){  
    return element * element;  
})  
console.log(array2)
```

```
▶ (5) [1, 4, 9, 16, 25]
```

## ❖ 배열(Array) 메서드

### 조건 메서드

- filter() : 특정조건 만족 하는 요소 추출 -> 새로운 배열 생성
- every() : and     ■ some() : or

```
let array = [1, 2, 3, 4, 5];  
let filterArr = array.filter(function(element, index, array){  
    return element <= 3;  
})  
console.log(filterArr);
```

```
function lessthan3(element, index, array){  
    return element < 3;  
}  
let everylt3 = array.every(lessthan3);  
let somelt3 = array.some(lessthan3);  
console.log(everylt3);  
console.log(somelt3);
```

▶ (3) [1, 2, 3]

false

true

## ❖ 배열(Array) 메서드

### 연산 메서드

- `reduce()` : 왼쪽에서 두 개씩 묶어 하나가 될 때까지 줄여 가는 연산
- `reduceRight()` : 오른쪽에서 두 개씩 묶어 하나가 될 때까지 줄여가는 것

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

```
let reducearr = array.reduce(function(pre, curr){  
  console.log(pre + ' : ' + curr);  
  return pre + curr;  
})  
console.log(reducearr);
```

1 : 2
3 : 3
6 : 4
10 : 5
15

```
let reduceRarr = array.reduceRight(function(pre, curr){  
  console.log(pre + ' : ' + curr);  
  return pre + curr;  
})  
console.log(reduceRarr);
```

5 : 4
9 : 3
12 : 2
14 : 1
15

## ❖ 배열(Array) 메서드

### sort() : 배열 요소를 순서대로 나열

- 비교함수를 인자로 넣어 배열 자체 정렬, 요소 위치 변화
- 오름차순( $a > b$ ,  $a - b$ ), 내림차순( $a < b$ ,  $b - a$ )

// 문자 정렬

```
let colors = [ '연두', '파랑', '갈색', '흰색'];
```

```
console.log(colors.sort(function(a,b){return a>b ? 1: -1;}));
```

```
console.log(colors.sort(function(a,b){return a<b ? 1: -1;}));
```

// 숫자 정렬

```
const arr1 = [3,2,7,1,4,1,6,9,8];
```

```
console.log(arr1.sort(function(a,b){return a-b;}));
```

```
console.log(arr1.sort(function(a,b){return b-a;}));
```





## ❖ 도전! 문제 풀이

### 문제1

```
function Student(name, kor, eng, math, art){
  this.name = name;
  this.kor = kor;
  this.eng = eng;
  this.math = math;
  this.art = art;
}
Student.prototype.getSum = function(){
  return this.kor + this.eng + this.math + this.art;
}
Student.prototype.getAvg = function(){
  return this.getSum()/4;
}
Student.prototype.toResult = function(){
  return this.name + '의 ' + this.getSum() + '점' + this.getAvg();
}
```

## ❖ 도전! 문제 풀이

```
let students = [];  
students.push(new Student('홍길동', 96, 98, 92, 98));  
students.push(new Student('성춘향', 88, 74, 78, 92));  
students.push(new Student('이몽룡', 69, 89, 92, 78));  
students.push(new Student('임영웅', 97, 95, 89, 98));  
students.push(new Student('성유리', 66, 80, 92, 94));  
  
let sungJuk = ' 이름\t총점\t평균\n';  
for (let i in students){  
    sungJuk += students[i].toResult() + '\n';  
}  
console.log(sungJuk);
```