Chap 37 Set과 Map

▼ Set

Set객체는 **중복되지 않는 유일한 값**들의 집합

▼ Set 객체의 생성

- 생성자함수 Set 사용
- 이터러블을 인수로 전달받아 Set 객체를 생성한다.

```
const set1=new Set([1,2,3,3]);
//중복되지 않는 유일한 값들의 집합이므로 3은 한개만 저장된다.
console.log(set1); //set(3){1,2,3}
```

▼ 요소 개수 확인

- Set.prototype.size 프로퍼티 사용
  - setter 함수 없이 getter함수만 존재하는 접근자 프로퍼티이기때문에 size 프로퍼티에 숫자를 할당하여 Set 객체의 요소 개수를 변경할 수 없다.

▼ 요소 추가 Set.prototype.add 메서드

- 새로운 요소가 추가된 Set 객체를 반환.
- add메서드 호출 후 바로 add 메서드 연속적으로 호출 가능
- 중복된 요소의 추가는 허용되지 않는다.
- **NaN과 NaN**, **0과-0**은 **같다고 판단**하여 중복해서 요소를 추가했을 시에는 하나만 추가된다.
- 자바스크립트 모든 값을 요소로 저장할 수 있다.

▼ 요소 존재 여부 확인 → Set.prototype.has 메서드

- 특정 요소의 존재 여부를 나타내는 불리언 값을 반환한다.

```
const set=new Set([1,2,3]);

console.log(set.has(2)); //true
console.log(set.has(4)); //false
```

▼ 요소 삭제 → Set.prototype.delete 메서드

- 삭제 성공 여부를 나타내는 불리언 값을 반환
- 삭제하려는 요소 값을 인수로 전달해야한다. (why? Set은 인덱스가 없기때문)
- 만약 존재하지 않는 요소를 삭제하려고 하면 에러 없이 무시된다.
- 연속적으로 호출할 수 없다.

```
const set = new Set([1,2,3]);
set.delete(2);
console.log(set);//set(2){1,3}

//연속 호출 x
set.delete(1).delete(3); //TypeError:set.delete(...)~
```

▼ 요소 일괄 삭제 → Set.prototype.clear 메섣

- 언제나 undefinedㄹ르 반환

```
const set= new Set([1,2,3]);
set.clear();
console.log(set);//set(0){}
```

▼ 요소 순회

1. Set.prototype.forEach 메서드
    - 3개의 인수를 전달받는다.
        - 첫 번째 인수 : 현재 순회 중인 요소 값
        - 두 번째 인수 : 현재 순회 중인 요소 값
        - 세 번째 인수 : 현재 순회 중인 Set 객체 자체

→ 첫번째 인수와 두번째 인수가 같은 것은 Set.prototype.forEach 메서드와 인터페이스를 통일하기 위함이며 다른 의미는 없다.

2. for...of문 순회

- Set 객체는 이터러블이기 때문에 for..of문으로 순회할 수 있으며, 스프레드 문법과 배열 디스트럭처링의 대상이 될 수도 있다.

▼ 집합 연산

▼ Set과 배열의 차이

배열과 유사하지만 아래와 같은 차이가 있다.Set은 수학적 집합의 특성과 일치한다. 따라서 Set을 통해 교집합, 합집합, 차집합, 여집합 등을 구현할 수 있다.

구분	배열	Set 객체
동일한 값을 중복하여 포함할 수 있다.	O	X
요소 순서에 의미가 있다.	O	X
인덱스로 요소에 접근할 수 있다.	O	X

▼ Map

키와 값의 쌍으로 이루어진 컬렉션

▼ Map 객체의 생성

- Map 생성자 함수로 생성
- 이터러블을 인수로 받아 Map 객체 생성. 이터러블은 키와 값의 쌍으로 이루어진 요소로 구성되어야 한다.
- 중복된 키를 갖는 요소가 존재하면 값이 덮어써진다. → 중복된 키를 갖는 요소 존재할 수 없다.

▼ 요소 개수 확인 → Map.prototype.size 프로퍼티

- size 프로퍼티는 setter함수 없이 getter함수만 존재하는 접근자 프로퍼티이다. 따라서 size프로퍼티에 숫자를 할당하여 Map 객체의 요소 개수를 변경할 수 없다.

▼ 요소 추가 → Map.prototype.set 메서드

- 연속적으로 호출 가능
- 중복된 키를 갖는 요소를 추가하면 값이 덮어 써진다.
- NaN과 NaN, +0과 -0은 갖다고 평가하여 중복 추가를 허용하지 않는다.

- Map 객체는 그냥 객체와 달리 키 타입에 제한이 없다.

▼ 요소 취득 → Map.prototype.get

- 인수로 키를 전달하면 Map객체에서 인수로 전달한 키를 갖는 값을 반환한다.
- 인수로 전달한 키를 갖는 요소가 존재하지 않으면 undefiend반환

▼ 요소 존재 여부 확인 → Map.prototype.has 메서드

- 특정 요소의 존재 여부를 나타내는 불리언 값을 반환

▼ 요소 삭제 → Map. prototype.delete

- 삭제 성공 여부를 나타내는 불리언 값을 반환
- 연속 호출 불가능

▼ 요소 일괄 삭제 → Map.prototype.clear 메서드

- clear메서드는 언제나 undefined를 반환한다.

▼ 요소 순회

1. Map.prototype.forEach

- 3개의 인수를 전달받는다.
  - 첫번째 인수 : 현재 순회 중인 요소 값
  - 두번째 인수 : 현재 순회 중인 요소키
  - 세 번째 인수 : 현재 순회 중인 Map 객체 자체

2. for...of문

- Map 객체는 이터러블이기때문에 for...of문으로 순회할 수 있다.
- 스프레드 문법과 배열 디스트럭처링 할당의 대상이 될 수도 있다.

▼ 이터러블이면서 동시에 이터레이터인 객체를 반환하는 메서드 제공

- Map.prototype.keys
- Map.prototype.values
- Map.prototype.entries

▼ Map과 객체의 차이점

구분	객체	Map 객체
키로 사용할 수 있는 값	문자열 또는 심벌 값	객체를 포함한 모든 값
이터러블	X	O
요소 개수 확인	Object.keys(obj).length	map.size