Set

set은 중복을 허용하지 않는 값을 모아놓은 특별한 컬렉션이다.
map은 key-value 로 저장하는 반면 set은 value값만 저장한다.

주요메서드

-new Set(iterable) => 셋을 만든다. 이터러블 객체를 전달 받으면 그안의 값을 복사해 set에 넣는다.
-set.add(value) => value값을 추가하고 set 자신을 반환합니다.
*set.add(value) 동일한 value값이 있다면 아무리 많이 호출하더라도 반응이 없다. set 내에 중복이 없는 이유
방문자 방명록을 만든다고 가정. 한 방문자가 여러 번 방문해도 방문자를 중복해서 기록하지 않겠다고 결정 내린 상황이다.
즉, 한 방문자는 '단 한 번만 기록'되어야 한다. 이때 적합한 자료구조가 바로 `Set`이다.
-set.delete(value) => value값을 제거한다.
-set.has(value) => value값이 존재한다면 true 없다면 false로 반환한다.
-set.size => set에 몇 개의 값이 있는지 센다.

set의 반복 작업

- for(..of..)와 forEach 를 사용하여 set의 값을 대상으로 반복작업을 수행할 수 있다.

```
var setIter = set.keys()  // set.key() => set 내에 모든 값을 포함하는 이터러블 객체를 반환한다.

for (var key of setIter){   //set에 있는 key만큼 동작하며 {}조건을 수행한다.
 console.log(key)}

 set.forEach(value){   //forEach를 이용하여 set에 있는 value값들을 각각 {}조건을 수행하게 한다.
 console.log(value)}
```

-set.entries() => set 내의 각 값을 이용해 만든 [value,value] 배열을 포함하는 이터러블 객체를 반환한다. **Map과의 호환성을 위해 만들어 졌다.**

Map

map은 'key'가 있는 데이터를 저장한다. 또한 key에 대한 다양한 자료형을 허용한다.

주요메서드

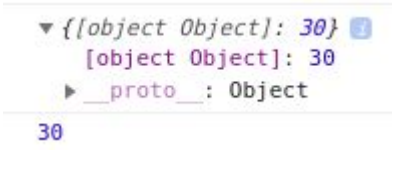
-new Map() => 맵을 만든다.
-map.set(key,value) => key를 이용해 value를 저장한다.
-map.get(key) => key값에 해당하는 value 값을 반환한다.
-map.has(key) => key값이 존재한다면 true 없다면 false로 반환한다.
-map.delete(key) => key에 해당하는 value값을 삭제한다.
-map.clear() => 맵 안의 모든 요소를 제거한다.
-map.size => set에 몇 개의 값이 있는지 센다.

*Map[key] =value 는 Map을 쓰는 바른 방법이 아니다.
map[key] =2 값을 설정하듯 map[k]=v 를 사용할 수 있긴 하지만
이 방법은 map을 일반 객체처럼 취급하게 되며, 여러가지 제약이 생기게 된다.

```
let test = { name : "ted"}
let mapmap = new Map()
mapmap.set(test,30) // key는 name: "ted" 가 되고 30은 name 의 value 값으로 30이 된다.
console.log(mapmap)
console.log(mapmap.get(test)) // test key 'name'의 value인 30을 출력.
```

객체형 키를 객체에 쓸 경우.

```
let test = { name : "ted"}
let mapmap ={}
mapmap[test]=30
console.log(mapmap)//  mapmap은 객체이기 때문에 모든 키를 문자형으로 변환한다
                       이 과정에서 test는 문자형으로 변환되어 [object Object] 가 된다.
console.log(mapmap["[object Object]"])  // 30 value값을 출력하기 위해선
                                           [object Object]를 호출해야된다.
```



Map

Map의반복작업
-map.keys() => 각 요소의 key값을 모은 반복 가능한 객체를 반환
-map.values() => 각 요소의 value 값을 모은 이터러블 객체를 반환한다.
-map.entries => 요소의 [key,value]를 한 쌍으로 하는 이터러블 객체를 반환한다. **이 객체는 for(...of...) 반복문의 기초로 쓰인다.**

```
let mapData = new Map(
  [   ["apple","red"],
      ["grape","purple"]  ])
for(let fruit of mapData.key()){ //key를 대상으로 순회한다.
    console.log(fruit) }   // apple,grape…

for(let color of mapData.value()){ //value를 대상으로 순회한다.
    console.log(color) }   // red,purple...

for(let entry of mapData(.entries())){ //  () 생략해도 상관없다.  [key,value] 쌍으로 순회한다.
    console.log(color) }   // apple,red,grape,purple...

mapData.forEach((value,key)
          =>  console.log(key:value)) // mapData의 key,value값을 순회하여 key:value로 출력
```

Iterator Interface

```
{   next() {
          return {value: 1, done: true};
  } }
```

next()라는 key를 갖는다.
값으로 인자를 받지 않고, IteratorResultObject 를 반환하는 함수가 온다.
IteratorResultObject는 value 와 done 이라는 'key'를 갖고 있다.
value는 여러타입이 올수 있고 done은 반복할 수 있는지 없는지 Boolean값을 반환한다.