Set

WE'LL COVER THE FOLLOWING ^

- add()
 - Adding multiple values
- has()
- forEach()
- entries()
- keys()/values()
- delete()
- clear()
- Unique Values only
- WeakSet
- Additional Resources

In the previous chapter we looked at the Map object, and how we can use it to store key/value pairs. In this chapter we will look at Set and how to store unique values.

A Set is created in the same way as Map.

```
const shoppingList = new Set();
```

add()

With a Set we can use the .add() method to add elements to the Set. This is where a Map and Set differ: there are no keys in a Set.

```
snoppingList.add('Cneese');
```

Adding multiple values

When we create a new Set we can pass in an array of values for our Set.

```
const shoppingList = new Set(['Milk','Cheese']);
```

has()

The .has method can be used to check if a value exists in the Set.



forEach()

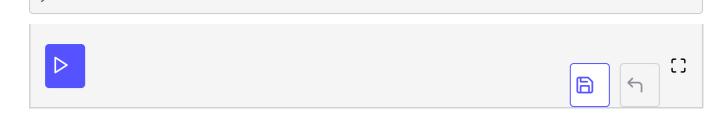
There is no <code>.get()</code> method for a Set, so we have to use either a <code>.forEach()</code> or an Iterator Object to get access to them. The <code>.forEach()</code> behaves as expected: it takes a callback function that is provided three arguments. Oddly, since there are no keys in a Set, the first two are the values from the Set. The last is the the original Set being iterated over.

```
shoppingList.forEach((value) => {
  console.log(value);
});
```

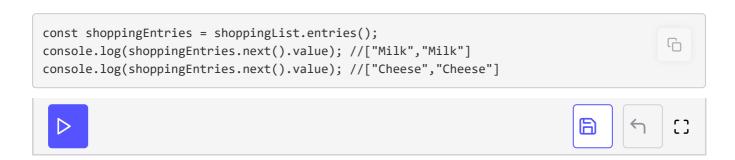
entries()#

Much of the Set API matches that of Map. A Set also has an .entries method which returns an Iterator Object, and we can use a for...of and the .next() methods to iterate over it.

```
for(let [value, sameValue] of shoppingList.entries()) {
  console.log(value);
  console.log(sameValue);
}
```



Again, since a Set has no key, the .entries() method returns the same values.

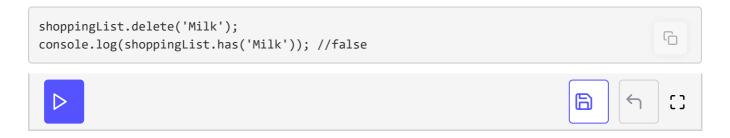


keys()/values()

Similar to Maps, we can use the .keys() and .values() methods to access the elements in our Set. However since there are no keys in a Set both of these methods will return an Iterator Object that will provide the values in the Set.

delete()

Again, much like a Map, we can delete a property with the .delete() method, in this case we can just pass the value we want to remove from our Set.



clear()#

If we need to completely wipe our Set, we can use the .clear() method.

Unique Values only

A Set can only contain unique values, so if you try to add a duplicate it will do nothing.

This is a great little trick for if you have an array of duplicate vales, and you want to get the unique values out of it.

WeakSet

Just like a Map, the Set has a weak alternative. However there is a key difference here. For a WeakSet, you are only allowed to store objects, not any other value.



Just like a WeakMap, it will hold on to these values weakly, meaning when they are not referenced anymore, they will be garbage collected.

Additional Resources

- https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_O bjects/Set
- https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Global_O bjects/WeakSet