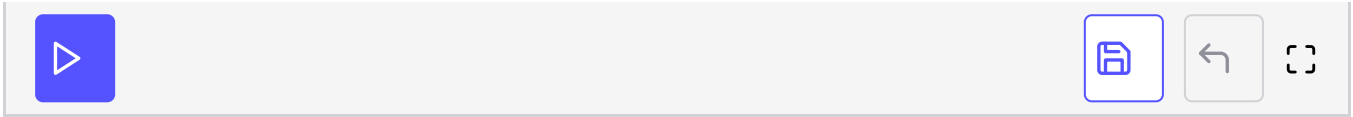
Iterating Sets

set traversal using the forEach method and for...of loop

Sometimes we have to traverse all the elements of a set. This can be done in multiple ways:

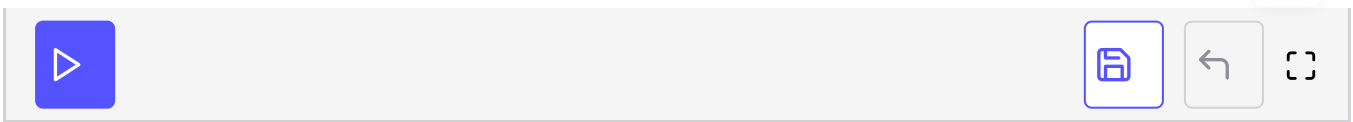
- `forEach` method,
- `for...of` loop,
- transforming a set into an array by using the spread operator

```
console.log('forEach function:')
moreColors.forEach( value => { console.log( value ) } );
//> red
//> blue

console.log('\nfor...of loop:')
for ( let value of moreColors ) {
    console.log( value );
}
//> red
//> blue

console.log('\nspread operator:')
console.log( [...moreColors] );
//> ["red", "blue"]
```



Technically, the function argument of the `forEach` method accepts up to three parameters:

```
moreColors.forEach( console.log );
```



As `console.log` accepts a variable number of arguments, it prints all the arguments the iteration provides. These are:

- the upcoming value stored in the set,
- the belonging key,
- a reference to the set itself.

The second and the third arguments are not too useful. Keys and values are pairwise equal in sets. Therefore, the first two arguments are always equivalent in sets. The reason why a key was provided in `forEach` is compatibility with maps. We may need the third argument in case we don't have a reference to the set. However, use cases for this are rare.

As `Set` is an ordered list, iteration is performed in the order of adding the elements.

Now, let's move on to Maps.