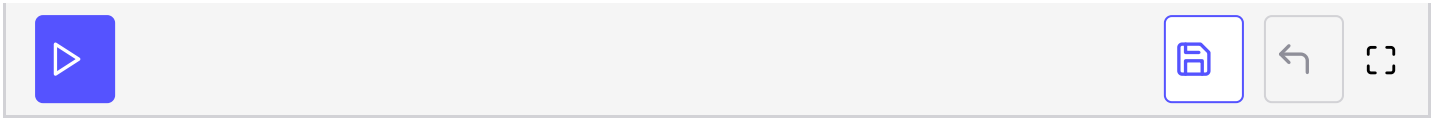
Iterating an Object's Keys with For-In

We'll cover the following

Iterating an object's keys with for-in

The first way is to iterate the key of an object. Looping the key is the role of the `for-in` loop. You can use for-in on a normal array. The result is a list of indices which would be the sequential order of each element. On an object, you will iterate all members' names.

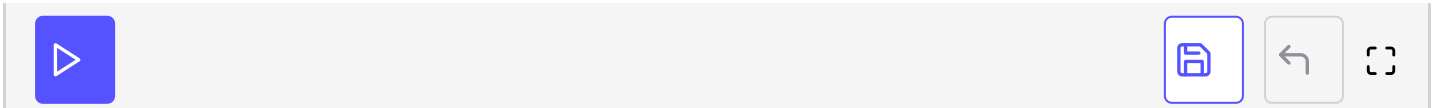
```
let list1: (number | string)[] = [1, 2, 3, "a", "b", "c"];
for (let i in list1) { // Loop all indexes, not values
    console.log(i); // Print: 0, 1, 2, 3, 4, 5
}
```



Iterating an object's keys with for-of

The second way to iterate a collection is on the value of an object which is done using `for-of`. Using it on an array will provide the values in the array. Using it on an object will not work; `for-of` is more restrictive because it must implement `Symbol.iterator`. The difference is that you cannot use this iteration mechanism on a literal object like you can with `for-in`.

```
let list2: (number | string)[] = [1, 2, 3, "a", "b", "c"];
for (let i of list2) { // Loop all values
    console.log(i); // 1, 2, 3, "a", "b", "c"
}
```



Iterating with `forEach`

There is a short-hand for a structure using `Symbol.iterator` which is to use the function `forEach`. The parameter is the element inside the iterable structure. If you have an array of numbers, each result will be a number; if it's an array of objects, it will be an object. `forEach` also has a second parameter which is the index of the element in the array. The third parameter is the array itself. The last parameter is rarely used.

```
let list3: (number | string)[] = [1, 2, 3, "a", "b", "c"];
list3.forEach((v: string | number, index: number, array: (string | number)[]) => {
    console.log("Value " + v + " at position " + index);
});
```