Preserving the Order in Arrays

The problem with iterating over objects is that the order of iteration can vary. We will use State Object Normalization to solve this.

Caveat #2 : Preservation of Order.

This is perhaps the number one reason people use arrays. Arrays preserve the order of their values.

You have to see an example to understand this.

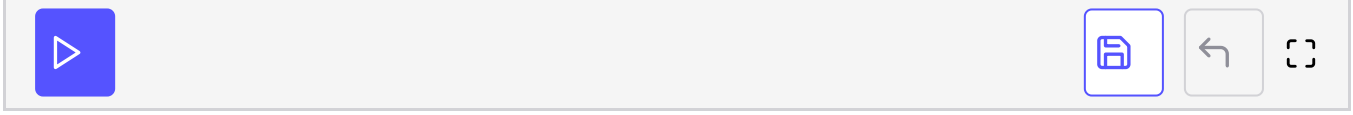
```
const numbers = [0,3,1,6,89,5,7,9]
```

Whatever you do, fetching the value of numbers will always return the same array, with the order of the inputs unaltered.

How about an object? Try it out below.

```
const numbers = {
    0: "Zero",
    3: "Three",
    1: "One",
    6: "Six",
    89: "Eighty-nine",
    5: "Five",
    7: "Seven",
    9: "Nine"
};

console.log(numbers);
```



The order of the object values aren't returned in the same way!

Now, depending on the kind of application you're building, this can cause very serious problems. Especially in apps where order is paramount.

You know any examples of such an app? Well, I do. A chat application!

You know any examples of such an app? Well, I do. A chat application!

If you're representing user conversations as an object, you sure care about the order in which the messages are displayed!

You don't want a message sent yesterday, showing like it was sent today. Order matters. So, how would you solve this?

Solution: Keep a Separate Array of IDs to denote Order.

You must have seen this before, but you perhaps didn't pay attention.

For example, If you had the object created above, you could keep another array to denote the order of values.

```
numbersOrderIDs: [0, 3, 1, 6, 89, 5, 7, 9]
```

This way you can always keep track of the order of values - regardless of the behaviour of the object.

If you need to add values to the object, you do so, but also push the associated ID to the **numbersOrderIDs** as well.

It is important to be aware of these things as you may not always have control over somethings. You may pick up applications with state modelled in this way. And even if you don't like the idea, you definitely should be in the know.

For the sake of simplicity, the IDs of the messages for the Skypey application will always be in order - as they are numbered in increasing values from zero upwards.

This may not be the case in a real app. You may have weird auto generated IDs that looks like gibberish e.g y68fnd0a9wyb

In such cases, you want to keep a separate array to track the order of values. That is it!

It is worth stating that the entire process of normalizing the state object may be summarised as these:

- Each type of data should have its own key in the state object.
- Each key should store the individual items in an object, with the IDs of

the items as keys and the items themselves as the values.

- Any references to individual items should be done by storing the item's ID.
- Ideally, keep an array of IDs to indicate ordering.

And there you have it. After developing a detailed design, our state object is ready for use. Time for a recap.