Arrays

This lesson highlights the key features of the array data structure.

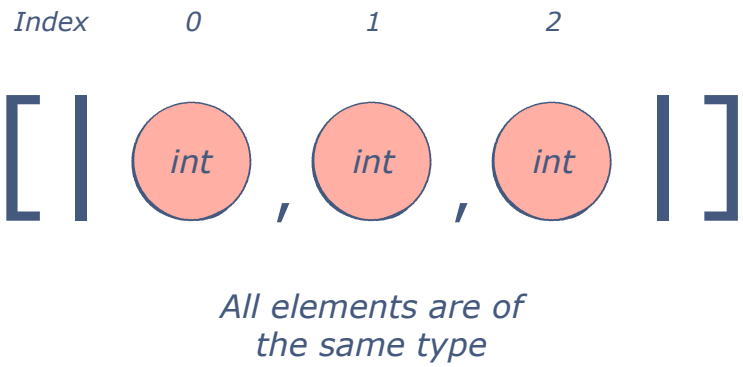
WE'LL COVER THE FOLLOWING

The Structure

An array stores elements of the same data type. What makes arrays unique is that they are not always immutable like *tuples* and *records*. An array can be static, which is immutable, or dynamic, which is mutable.

Similar to strings, an array of `n` elements is indexed from `0` to `n-1`. This means the first element of the array is considered to be at the zeroth position.

The contents of an array are enclosed in the `[| |]` operators.



All elements are of the same type

The Type of an Array

Similar to integers and strings, arrays have their own custom type in ReasonML. For any type called `data type`, the type of an array would be

```
array(data type)
```

This data type can be used in type definitions as well.

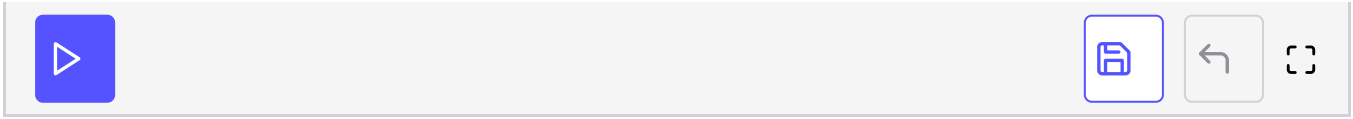
Static Arrays

To define a static array, we need to list all of its contents in the definition, similar to what we do for tuples or records.

Here's an example of a static array:

```
let arr = [| 1, 2, 3, 4, 5 |];
Js.log(arr);

/* Erroneus code */
/* arr = [| 1, 2, 3, 4, 5, 6 |]; */
```



Running the code in **line 5** would produce an error since the array is immutable.

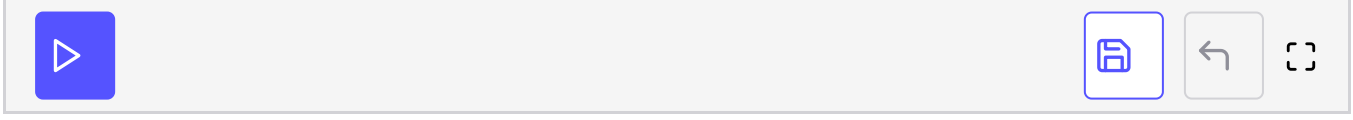
Dynamic Arrays

Dynamic arrays can be created using built-in array functions. The simplest way to create a dynamic array is the `Array.make()` method. The syntax for this method is as follows:

```
Array.make(length, value_of_elements)
```

Each index in the array will contain the same value. Here is an example:

```
let arr = Array.make(10, 0);
Js.log(arr);
```



An alternative approach is to use the `Array.init()` method. The first parameter of this method requires a number, `n`. The second argument is a *function*, that takes values from `0` to `n-1`, performs an operation on them, and

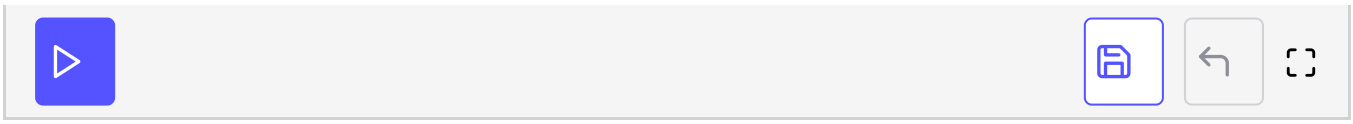
inserts these new values into the array.

Since we haven't visited functions yet, we don't need to worry about the code below.

Let's just have a look at `Array.init()` for conceptual purposes:

```
let f = (x) => x; /* For now, consider x to be an element of an array.
f simply returns x as it is. */

let arr = Array.init(10, f); /* Each element becomes x one by one and goes through f */
Js.log(arr);
```



The next lesson will showcase some of the operations we can perform with arrays.