Lists

In this lesson, we will be introduced to the advanced variant data structure called the list.

WE'LL COVER THE FOLLOWING

The Definition

> A list is an immutable data structure which stores homogenous elements (elements of the same type) in sequential form.

The reason it differs from other structures like tuples or arrays is that a list is a **polymorphic variant type**.

To understand why a list is a variant, we need to look at its structure.

The Structure

As a general convention in Reason, the start of a list is called the `head`, while the end of the list is known as its `tail`.

Here is the template for a Reason list:

```
type list('a) =
  | []
  | [head, ...tail]
```

Just like any other variant type, the list is made up of constructors, specifically two.

The `[]` constructor is used for an empty list.

The `[head, ...tail]` constructor uses pattern matching to divide the list into two parts:

- The `head` or the first element.
- The rest of the list defined using the spread operator.

The type variable, `'a`, tells us that the list is polymorphic and works for all data types.

We can use the `switch` expression to perform various operations with these two constructors, but before that, let's lay down the basics first.

Creating a List

A list can be created just like an array, except that instead of the `[| |]` operators, the elements of a list are closed within simple square brackets, `[ ]`.

Another difference between lists and arrays is that lists are *static*, which means that their content has to be defined at the time of creation.

Let's make a simple list of strings:

```
let aList = ["Reason", "OCaml", "JavaScript"];
Js.log(aList);
```

The code above is just syntactic sugar for the actual structure of the list. Our `aList` is actually interpreted in the `[head, ...tail]` format:

```
let aList = ["Reason", ...["OCaml", ...["JavaScript", ...[]]]];
Js.log(aList);
```

From the structure above, we can see that a single list is actually a list of recursively nested lists. Due to this, the list is known as a **recursive data structure**.

> **Note**: The empty list at the end is displayed as a `0` because its constructor defines it to be an integer of value `0`.

The Length of a List

The list structure comes with a built-in method for calculating its length. This function is called `List.length()`.

Let's calculate the length of the list we created earlier:

```
let aList = ["Reason", "OCaml", "JavaScript"];
Js.log(List.length(aList));
```

In the next lesson, we'll discover more operations that a list can perform.