Mutability

In this lesson, we'll make a mutable let variable using the ref() wrapper.

WE'LL COVER THE FOLLOWING

- The `ref()` Wrapper
- Reference

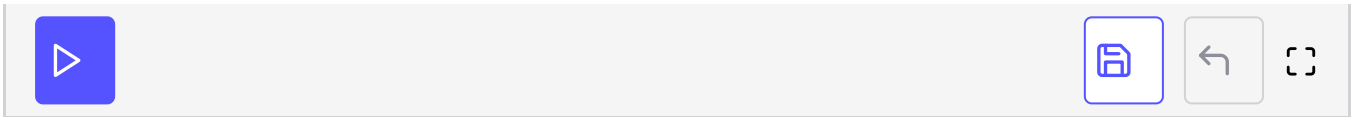
We've already observed how normal `let` binding variables are *immutable*. One approach was to redefine the binding with a new value.

However, there is an actual way to make `let` bindings mutable!

The `ref()` Wrapper

In order to achieve mutability, the value being assigned to a variable must be enclosed in a `ref()` wrapper. To assign a new value, we must use the `:=` operator:

```
let x = ref(10);
Js.log(x);
x := 10 * 5;
Js.log(x);
```



The `[]` brackets indicate the presence of a `ref` wrapper. Since we are printing the entire wrapper, these brackets are visible.

Reference

The value inside a `ref` wrapper is known as a reference. We can access this value using the `^` operator. This is extremely useful if the value of the variable is being changed using its current value:

```
let x = ref(10);
Js.log(x);
x := x^ + 5;
Js.log(x);
Js.log(x^);

/* Erroneous code */

/* x := x + 5; A wrapper and integer cannot be summed! */
```

Now, the `[]` brackets have disappeared as we are only accessing the value inside the wrapper.

And it's as simple as that!

The reason `ref()` works is because it's actually a type of **record**. A record is a mutable data structure which we'll cover in the next section.

We're done with our discussion on data and type variables. Let's test our theoretical understanding of identifiers with a fun quiz.