The if-else Expression

This lesson will teach us how to create if-else expressions in Reason syntax.

WE'LL COVER THE FOLLOWING

The `if` condition is one of the most relevant statements in the world of computer programming. It is supported in many popular languages such as Java, C++, and Python. The condition is used to execute certain operations **if** a condition is fulfilled.

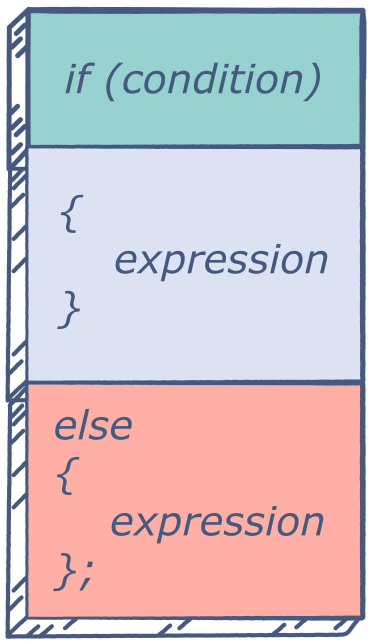
The `else` keyword is derived from `if`. We'll understand more about this later in the lesson.

Let's talk about the structure of the `if-else` expression.

The Structure

The structure of Reason's `if-else` expression is fairly similar to the one followed in other languages. it can be divided into 3 parts:

1. The `if` condition
2. The expression to be executed
3. The `else` expression



The `if` Condition

In this part, we define a condition which can either be `true` or `false`. Therefore, the condition **must** return a boolean value.

The condition is enclosed in parentheses.

Keep in mind that this `if` condition is not the same thing as the entire `if` expression.

The Expression to be Executed

This is the block of code which will be executed by the compiler only when the `if` condition returns `true`.

Since its an expression, it will naturally return a value or values of a certain data type.

The expression is enclosed inside the scope operators, `{}`, implying that the `if` expression has its own scope. `let` bindings and types created here will exist only inside this scope.

The `else` Expression

This block of code is executed in case the `if` condition returns `false`. It is optional if the `if` block doesn't return anything.

The Syntax

Let’s write a couple of `if-else` expressions in Reason!

```
let a = 7;
let b = 3;

let result = if (a + b == 10) /* A boolean expression which is true */
{
  Js.log("The if condition is true");
  a * b; /* This expression is returned */
}
else
{
  Js.log("The if condition is false"); /* This expression is not executed */
  a - b;
};

Js.log(result);
```

In the code above, the `if` condition is `a + b == 10`. Since this condition is true, the block of code below it is executed and `a * b` is returned.

If we change the condition to `a + b > 10`, it will return false and the `else` block will be executed.

> **Note**: The return type of the expressions in the `if` and `else` blocks must be of the same type. In the example above, the type is `int`.

Compound Conditions

In the `if` expressions, we can also provide multiple conditions using the logical operators, `||` and `&&`.

Here’s an example:

```
let a = 10;
let b = 20;

if(a < b && a mod 2 == 0){ /* Multiple conditions */
  Js.log("The if condition is true");
} else {
  Js.log("The if condition is false");
};
```

In the code above, `a` has to fulfill both conditions in order to execute the `if` block.

> **Note**: When making comparisons, there should always be a space between `<` and the identifier to its right. For example, `a < b` or `a<10` will work. However, `a<b` or `10 <b` will produce a syntax error. This problem does not occur with any other comparison operators.

In the next lesson, we’ll take a look at nested `if-else` expressions.