

# Function Definition

Let's learn the logic and syntax needed to create our first function!

## WE'LL COVER THE FOLLOWING ^

- The Need for Functions
- The Structure of a Function
  - Function Name
  - Arguments
  - Expression
- The Template
- The Implementation

As we discussed earlier, a function performs a certain task and returns a value. This implies that the function will contain an *expression*.

## The Need for Functions #

A function can be used repeatedly, which helps us avoid writing redundant code.

Let's look at a simple program where we calculate and print the double of an integer:

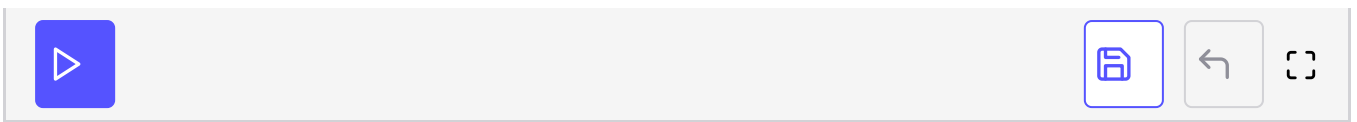
```
let two = 2;
let seven = 7;
let ten = 10;

let doubleOfTwo = two * 2;
Js.log(doubleOfTwo); /* 4 */

let doubleOfSeven = seven * 2;
Js.log(doubleOfSeven); /* 14 */

let doubleOfTen = ten * 2;
Js.log(doubleOfTen); /* 20 */
```



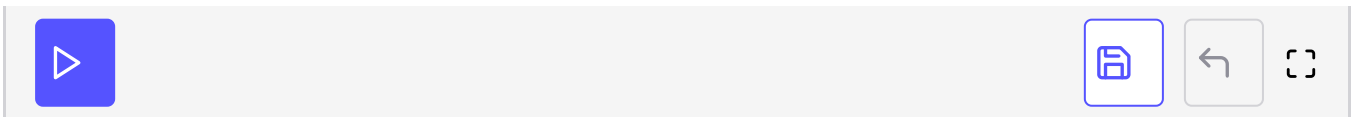


The code above is redundantly verbose for no reason. The simple steps are:

1. Calculate the double.
2. Print its value.

Let's suppose we had a function called `double()`, which could print the doubled value for any integer. Here's how it would work:

```
double(2);  
double(7);  
double(10);
```



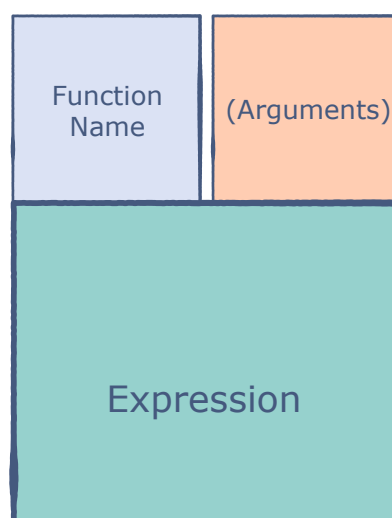
Pretty neat right? The function hides the needless code and performs its assigned task whenever called.


It's time for us to create our own `double()` method, but before that, we need to understand how a function is created.

## The Structure of a Function #

In programming terms, we refer to function creation as function **definition** or **implementation** since we are *defining* a function.

Below, we can find a simple illustration of the different components of a function.





A basic function.

## Function Name #

This is the identifier of the function. In the example above, the function name is `double`. We use `let` bindings to specify this name.

## Arguments #

Also known as parameters, arguments are the input values in a function. They will be used to perform computations within the function. In the `double()` method, the argument was the number whose double was computed. A function can have as many arguments as required.

## Expression #

This is where the actual task of the function is implemented.

## The Template #

Here's what a typical function in Reason looks like:

```
let functionName = (arguments) => {  
  expression  
};
```



- The `=` and `=>` operators are compulsory for a function.
- Multiple arguments can be separated by commas.
- The expression should return a value or object.

## The Implementation #

Finally, let's write the implementation for our `double()` function:

```
let double = (num: int) => {  
  num * 2;  
};  
let doubleOfTen = double(10);  
Js.log(doubleOfTen);
```



As a safety check, we've annotated the type of the `num` argument. The value returned by the function is bound to the `doubleOfTen` variable.

And just like that, we've created our first function! Since our function only contains a single-line expression, we could make it more concise in the following way:

```
let double = (num: int) => num * 2;  
let doubleOfTen = double(10);  
Js.log(doubleOfTen);
```



---

In the next lesson, we'll learn more about the scope of a function.