

# Defining a Function

In this lesson, we will start our discussion on functions and learn how to define our very own function.

## We'll cover the following ^

- Writing Your First Function
- Syntax
- Parameterized Functions
- Syntactic Sugar
- Where is main()?

## Writing Your First Function #

In a previous [lesson](#), you were introduced to built-in functions; also known as methods. In this chapter, we will cover user-defined functions and learn how to write our very own functions. Let's get straight to it and look at an example of a very basic user-defined function.

Ignore the *Driver Code* throughout this lesson.

```
void newPrint(){
    print("Function Called");
}

// Driver Code
main() {
    newPrint();
}
```



The above function is not really of any use to us as all it does is print the statement **Function Called**. However, regardless of what a function does, it follows a general syntax.

# Syntax #

- `void` is telling us that the function does not return anything. Remember when we were discussing the `contains` method used by collections? `contains` returned a boolean value. If your function returns something, this is where you would insert its return type, i.e., `int`, `String`, `bool`, etc.

Inserting the return type is not required as Dart can use [type inference](#) to infer the return type of the function. However, it is preferred that you insert the return type.

- `newPrint` is the name of the function in which you get to choose yourself. Make sure the name is meaningful to the functionality of the function.
- `newPrint` is followed by `()`.
- The body of the function is wrapped in curly brackets `{}`.
- `print("Function Called");` is the body of the function.

The general syntax is as follows:

```
returnType functionName () {  
    function body  
}
```

If your function has a return value, you have to return the value using the `return` keyword as we will see in the next section.

## Parameterized Functions #

We can create functions that take values just like methods.

Parameters are a mechanism to pass values to functions.

Let's create a function that takes two parameters and returns their sum.

```
num sum(num x, num y){  
    return x+y;  
}
```

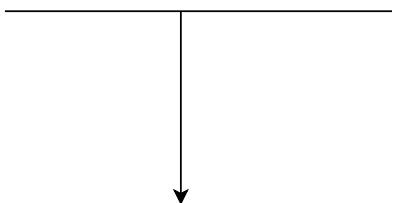
```
// Driver Code  
main() {  
    print(sum(1,2));  
}
```

In the first example, our function name was followed by empty parentheses `()`. This time around, the parentheses contain the parameters to be passed to the function.

On **line 2**, `(num x, num y)` is telling us that our function takes two parameters. The first one is named `x` and must be of type `num`. The second one is named `y` and must also be of type `num`.

On **line 3**, we are returning `x+y` using the `return` keyword.

```
returnType functionName(parameters) {  
    function body  
}
```



`dataType parameterName`

Parameters are separated by a comma `,`.

The type of the return value of the function must match the return type of the function. For instance, if the return type of the function is `String`, the function must return a `String` value.

## Syntactic Sugar #

As we already discussed, a function's body is encompassed in curly brackets `{ }`. However you can choose not to use the curly brackets if the function's body only

However, you can choose not to use the curly brackets if the function's body only consists of a single expression and write everything in a single line.

The shorthand syntax is a bit different.

**returnType functionName(parameters) => function body**

When using this syntax, you don't have to use the `return` keyword to return a value. The arrow separates the function parameters and function type from the function body.

This method of writing functions is simply syntactic sugar. It does not change the functionality of the function, only its syntax.

Let's write our `sum` function using the shorthand syntax.

```
num sum(num x, num y) => x+y;

// Driver Code
main() {
    print(sum(1,2));
}
```

## Where is `main()`? #

Up until now, we have been writing all of our code in the `main()` function. User-built functions do not need to be *defined* in the main function, but they can be. They do however need to be *called* in the `main()` function. Let's explore this a bit further in the next lesson.

Now that you know how to create your own functions, let's learn how to use them in the next lesson.