

Dart Functions:

Dart function is a set of codes that together perform a specific task. It is used to break the large code into smaller modules and reuse it when needed. Functions make the program more readable and easy to debug. It improves the modular approach and enhances the code reusability.

Suppose, we write a simple calculator program where we need to perform operations number of times when the user enters the values. We can create different functions for each calculator operator. By using the functions, we don't need to write code for adding, subtracting, multiplying, and divide again and again. We can use the functions multiple times by calling.

The function provides the flexibility to run a code several times with different values. A function can be called anytime as its parameter and returns some value to where it called.

Definition: Collection of statements group together to perform an operation.

Defining a Function

A function can be defined by providing the name of the function with the appropriate parameter and return type. A function contains a set of statements which are called function body. The syntax is given below.

Syntax:

```
return_type func_name (parameter_list):  
{  
    //statement(s)  
    return value;  
}
```

Let's understand the general syntax of the defining function.

- **return_type** - It can be any data type such as void, integer, float, etc. The return type must be matched with the returned value of the function.
- **func_name** - It should be an appropriate and valid identifier.
- **parameter_list** - It denotes the list of the parameters, which is necessary when we called a function.

- **return value** - A function returns a value after complete its execution.

Let's understand the following example.

Example - 1

```
int mul(int a, int b){  
    int c;  
    c = a+b;  
    print("The sum is:${c}");  
}
```

Calling a Function

After creating a function, we can call or invoke the defined function inside the main function body. A function is invoked simply by its name with a parameter list, if any. The syntax is given below.

Syntax:

```
fun_name(<argument_list>;  
or  
variable = function_name(argument);
```

Note - Calling function must be ended with semicolon (;).

When we call a function, the control is transferred to the called function. Then the called function executes all defined statements and returns the result to the calling function. The control returns to the main() function..

Example :

```
mul(10,20);
```

Passing Arguments to Function

When a function is called, it may have some information as per the function prototype is known as a parameter (argument). The number of parameters passed and data type while the function call must be matched with the number of parameters during function declaration. Otherwise, it will throw an error. Parameter passing is also optional, which means it is not compulsory to pass during function declaration. The parameter can be two types.

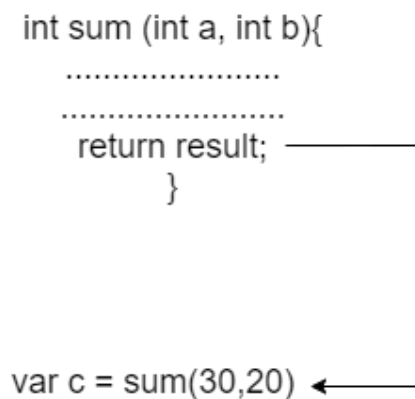
Actual Parameter - A parameter which is passed during a function definition is called the actual parameter.

Formal Parameter - A parameter which is passed during a function call is called the formal parameter.

Return a Value from Function

A function always returns some value as a result to the point where it is called. The **return** keyword is used to return a value. The return statement is optional. A function can have only one return statement. The syntax is given below.

```
int sum (int a, int b){  
    .....  
    .....  
    return result;  
}  
  
var c = sum(30,20) ←
```

A diagram illustrating the flow of a return value. It shows a function definition for 'sum' with two parameters, 'a' and 'b'. Inside the function, there is a 'return result;' statement. Below the function definition, there is a variable assignment 'var c = sum(30,20)'. A line connects the 'return result;' statement to the 'sum(30,20)' part of the assignment, indicating that the value returned by the function is assigned to the variable 'c'.

Syntax:

return <expression/values>

Example -

return result;

Note: A return-type in dart function is optional but it is highly recommended.

Function Examples

Let's understand the functions by using a program of adding two numbers using functions.

Dart Function with parameter and return value

In the following example, we are creating a sum() function to add two numbers.

Example - 1

```
void main() {  
    print("Example of add two number using the function");  
    // Creating a Function  
  
    int sum(int a, int b){  
        // function Body  
        int result;  
        result = a+b;  
        return result;  
    }  
    // We are calling a function and storing a result in variable c  
    var c = sum(30,20);  
    print("The sum of two numbers is: ${c}");  
}
```

We can also make the different other function:

- Dart Function with no-parameter and no-return value
- Dart Function with parameter and no-return value
- Dart Function with no-parameter and return value

Dart Anonymous Function

We have learned the Dart Function, which is defined by using a user-defined name. Dart also provides the facility to specify a nameless function or function without a name. This type of function is known as an **anonymous function, lambda, or closure**. An anonymous function behaves the same as a regular function, but it does not have a name with it. It can have zero or any number of arguments with an optional type annotation.

We can assign the anonymous function to a variable, and then we can retrieve or access the value of the closure based on our requirement.

An Anonymous function contains an independent block of the code, and that can be passed around in our code as function parameters. The syntax is as follows.

Syntax:

```
(parameter_list) {  
    statement(s)  
}
```

Let's consider the following example.

Example -

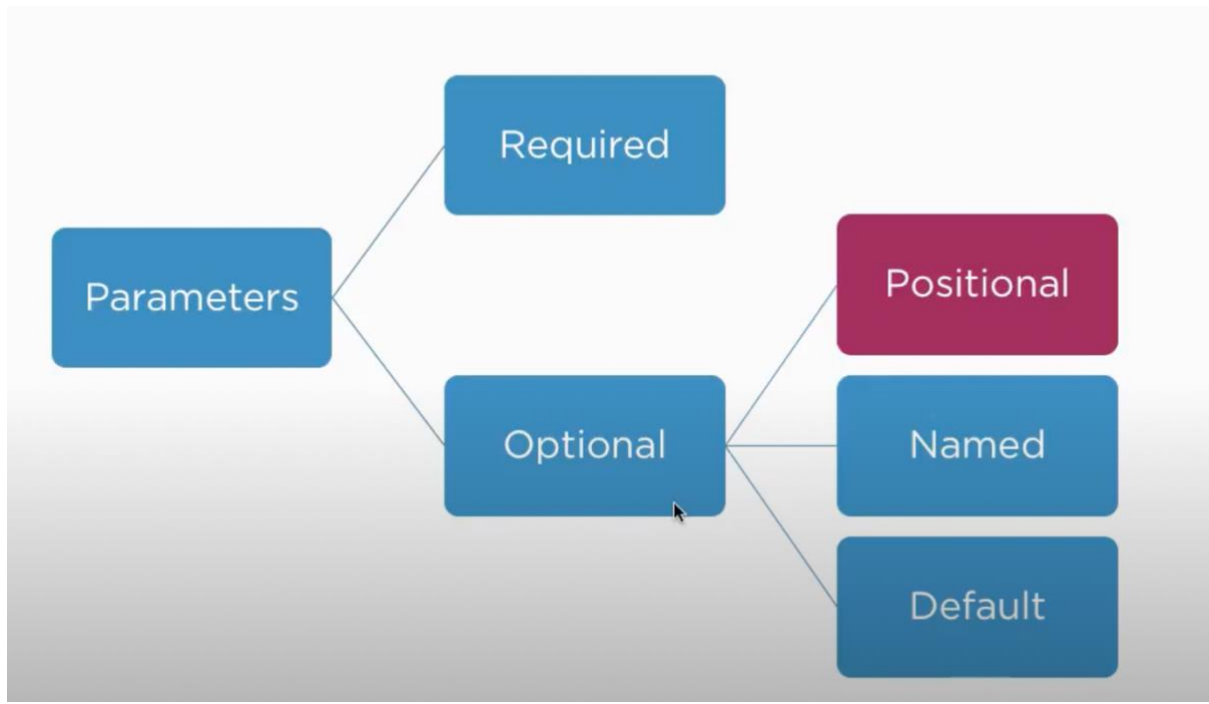
```
void main() {  
    var list = ["James", "Patrick", "Mathew", "Tom"];  
    print("Example of anonymous function");  
    list.forEach((item) {  
        print('${list.indexOf(item)}: $item');  
    });  
}
```

If the function consists of one statement, then we can also write the above code in the following way.

```
list.forEach(  
    (item) => print('${list.indexOf(item)}: $item'));
```

It is equivalent to the previous code. You can verify it by paste in your dart pad and run.

Types of Parameters in Dart Functions:



Required Parameters:

```
1
2 // function definition
3 void printCities(String name1, String name2, String name3) {
4     print("first city $name1");
5     print("first city $name2");
6     print("first city $name3");
7 }
8
9 Run | Debug
10 void main() {
11     //function call: all the parameters are required here.
12     printCities('ktm', 'pokhara', 'chitwan');
13 }
```

Optional Positional Parameters:

First remember the Optional parameter is determined by square brackets used in parameter list. As we define the Optional parameter, we must make it nullable.

```
1
2 // function definition
3 void printCities(String name1, String name2, [String? name3]) {
4     print("first city $name1");
5     print("first city $name2");
6     print("first city $name3");
7 }
8
9 Run | Debug
9 void main() {
10     //function call: all the parameters are required here.
11     printCities('ktm', 'pokhara');
12 }
```

Here we have defined the third parameter as optional position parameter. Consider the calling of the function where we have called the function just by passing only two parameters where third one is optional.

Note :Either we may pass or may not pass optional positional parameters.

Optional Named parameters:

- To prevent errors if there are large number of parameters.
- Named parameter is define by using the curly braces in parameters.

```

1
2 // function definition
3 void printCities(String name1, String name2, {String? name3}) {
4     print("first city $name1");
5     print("first city $name2");
6     print("first city $name3");
7 }
8
9 Run | Debug
10 void main() {
11     //function call: all the parameters are required here.
12     printCities('ktm', 'pokhara', name3: 'chitwan');
13 }

```

Don't forget to make the named parameters as nullable. In the above given example, we have made the third parameter as named parameter where it is surrounded by the curly braces.

When we called the function, we have used the parameter name.

Optional Default Parameters:

- We can assign default values to parameters.
- For default parameters either we can pass the new value or if we do not pass then it will take the default one.
- Optional default parameter is optional at the time of calling function.

```

1 // function definition
2 void printCities(String name1, String name2, {String name3='chitwan'}) {
3     print("first city $name1");
4     print("first city $name2");
5     print("first city $name3");
6 }
7
8 Run | Debug
9 void main() {
10     //function call: all the parameters are required here.
11     printCities('ktm', 'pokhara');
12 }

```