Dart Programming - Functions

Functions are the building blocks of readable, maintainable, and reusable code. A function is a set of statements to perform a specific task. Functions organize the program into logical blocks of code. Once defined, functions may be called to access code. This makes the code reusable. Moreover, functions make it easy to read and maintain the program's code.

A function declaration tells the compiler about a function's name, return type, and parameters. A function definition provides the actual body of the function.

Sr.No	Functions & Description
1	Defining a Function A function definition specifies what and how a specific task would be done.
2	Calling a Function A function must be called so as to execute it.
3	Returning Functions Functions may also return value along with control, back to the caller.
4	Parameterized Function Parameters are a mechanism to pass values to functions.

Optional Parameters

Optional parameters can be used when arguments need not be compulsorily passed for a function's execution. A parameter can be marked optional by appending a question mark to its name. The optional parameter should be set as the last argument in a function.

We have three types of optional parameters in Dart −

Sr.No	Parameter & Description
1	Optional Positional Parameter To specify optional positional parameters, use square [] brackets.
2	Optional named parameter Unlike positional parameters, the parameter's name must be specified while the value is being passed. Curly brace {} can be used to specify optional named parameters.
3	Optional Parameters with Default Values Function parameters can also be assigned values by default. However, such parameters can also be explicitly passed values.

Recursive Dart Functions

Recursion is a technique for iterating over an operation by having a function call to itself repeatedly until it arrives at a result. Recursion is best applied when you need to call the same function repeatedly with different parameters from within a loop.

Example

Live Demo

```
void main() {
   print(factorial(6));
}
factorial(number) {
   if (number <= 0) {
      // termination case
      return 1;
   } else {
      return (number * factorial(number - 1));
      // function invokes itself
   }
}
```

It should produce the following **output** −

```
720
```

Lambda Functions

Lambda functions are a concise mechanism to represent functions. These functions are also called as Arrow functions.

Syntax

```
[return_type]function_name(parameters)=>expression;
```

Example

Live Demo

```
void main() {
   printMsg();
   print(test());
}
printMsg()=>
print("hello");

int test()=>123;
// returning function
```

It should produce the following **output** −

```
hello 123
```