ScopeTreasure Courses

DART COURSES · LESSON D.6 · CODE SCOPE SERIES

Functions in Dart

Wrap steps into a named machine: parameters, return values, arrow functions and named parameters with defaults.

Introduction

When the same steps appear twice, a good programmer refuses to copy them — they pack the steps into a **function**: a named machine that takes inputs, does the work, and (often) hands back an answer. You have been using functions since Lesson D.1 — `print()` is one, and even `main()` is one. Today you build your own. *(Roman Urdu: Function aik naam wali machine hai — input do, kaam hota hai, jawab milta hai.)*

Key points

- A function is **declared once** and **called many times**. Declaration: return type, name, `(parameters)`, then `{ work }` — for example `void salam(String name) { ... }`. Calling is just the name plus values: `salam('Ali');`.
- `void` means "this function gives nothing back — it only *does* something" (like printing). A type like `int` before the name means the function **returns** that type of answer, using the `return` word.
  (Roman Urdu: void = sirf kaam karta hai; return = jawab wapas deta hai.)
- **Parameters** are the function's input boxes, written in the `( )` of the declaration. The values you pass when calling are called **arguments**. Order matters: the first argument goes into the first parameter.
- `return` **ends the function immediately** — lines after it in that function never run. The returned value can be stored: `int result = square(6);`.
- **Arrow functions:** when a function's whole body is one small expression, write it in one line with `=>`: `double area(double l, double w) => l * w;` — no `{ }`, no `return` word; the value after `=>` *is* the answer.
- **Named parameters** use `{ }` inside the `( )`: `void greet({String name = 'Guest'})`. Callers then pass by name — `greet(name: 'Ali')` — and any parameter with a **default value** may be skipped entirely. Flutter's widget code is full of named parameters, so learn this well.

Code examples

Example 1 — A void function, called twice

```
void salam(String name) {
  print('Assalam o Alaikum, $name!');
}

void main() {
  salam('Ali');
  salam('Sara');
}
```

What it does: The function is declared once, *above* main. Each call runs the same steps with a different name. Writing the greeting logic once means fixing it once — if the greeting text ever changes, there is exactly one place to edit.

What you will see: two lines — `Assalam o Alaikum, Ali!` and `Assalam o Alaikum, Sara!`. Notice `salam` itself never runs until main calls it: a declared function is a machine sitting ready, not a machine running.

Example 2 — return: the function answers back

```
int square(int n) {
  return n * n;
}

void main() {
  int result = square(6);
  print(result);
  print(square(10));
}
```

What it does: `square` receives a number, multiplies it by itself, and `return` hands the answer back to the caller. The first answer is stored in `result` and printed; the second is printed directly, without storing.

What you will see: two lines — `36` and `100`. If you remove the word `return` (just `n * n;`), Dart gives an error: a function that promises an `int` must actually hand one back. The type before the function name is a promise.

Example 3 — Arrow function + named parameters with defaults

```
double area(double length, double width) => length * width;

void greet({String name = 'Guest', int times = 1}) {
  for (int i = 0; i < times; i++) {
    print('Hello, $name!');
  }
}

void main() {
  print(area(5, 3));
  greet(name: 'Ali', times: 2);
  greet();
}
```

What it does: `area` is a one-line arrow function — its answer is whatever `length * width` gives. `greet` has two named parameters, both with defaults, so the caller may pass both, one, or none; the loop from Lesson D.5 repeats the greeting `times` times.

What you will see: four lines — `15.0`, then `Hello, Ali!` twice, then `Hello, Guest!` once. The last call, `greet()`, used both defaults. Named parameters are exactly how Flutter widgets take settings like `color:` and `size:` — remember this pattern when F.3 arrives.

Quick recap

- Declare once, call many times. Inputs = parameters; outputs come back with `return`.
- `void` = does work, returns nothing. A type name = the function must return that type.
- `return` stops the function at once.
- Arrow `=>` = one-expression body, the value is the answer.
- Named parameters `{name = default}` are called by name and can be skipped when a default exists.

Common mistakes

1. **Declaring but never calling** — the function sits ready and nothing happens, so the beginner thinks the code "did not work". A function runs only where its name is called, like `salam('Ali');` inside main.
2. **Arguments in the wrong order** — `area(3, 5)` vs `area(5, 3)` gives the same area, but `greet` style data (name, age) swapped gives nonsense or a type error. Positional parameters follow the declaration's order, always.
3. **Forgetting return in a value function** — a function declared `int` that reaches its end without `return` is an error in Dart. If the function promises an answer, every path through it must return one.

Practice

1. Write a function `int add(int a, int b)` that returns the sum, and a `void` function `showLine()` that prints a line of dashes. Use both in main to print a small "bill": dashes, two items with prices, dashes, then the total from `add`.
2. Convert this idea into an arrow function: a function that takes Celsius and returns Fahrenheit — formula: `c * 9 / 5 + 32`. Test it with 37 (expected answer: 98.6).

Practice quiz

1. What does void mean in void salam(String name)?

- The function returns a number
- The function gives nothing back, it only does something
- The function is empty and does nothing
- The function runs forever

Answer: The function gives nothing back, it only does something

2. What does return do inside a function?

- It prints the value and keeps running
- It ends the function immediately and gives the value back
- It restarts the function
- It deletes the function

Answer: It ends the function immediately and gives the value back

3. What does an arrow function like double area(double l, double w) => l * w; do?

- It needs braces and the word return
- Its one expression is the answer, with no braces and no return word
- It runs only once, ever
- It is a comment

Answer: Its one expression is the answer, with no braces and no return word

---

Watch the video lesson: https://youtu.be/nKf3N-7CFXc

Read online: https://scopetreasure.com/notes/dart-functions · More free tools and notes at https://scopetreasure.com