ScopeTreasure Courses

DART COURSES · LESSON D.9 · CODE SCOPE SERIES

Null Safety — Khali Cheez Ka Khauf Khatam

Why Dart refuses to crash on empty values: String? , the ?. and ?? shortcuts, if-checks, and the dangerous ! — used safely.

Introduction

The most famous crash in programming history is the "null error" — a program reaching for a value that is not there, like opening a box you expected to hold a name and finding it empty. Dart was rebuilt so this crash almost cannot happen: the language forces you to *declare* which boxes may be empty, and then to handle the empty case before it can hurt you. This system is called **null safety**, and it is the last big Dart idea before Flutter. *(Roman Urdu: null matlab "kuch nahi hai" — Dart aap ko pehle se bata deta hai ke konsa dabba khali ho sakta hai.)*

Key points

- `null` is Dart's word for "no value — the box is empty." It is not zero, it is not an empty text `''` — it is *nothing at all*.
- A normal type like `String` is a **promise: never null**. Writing `String name = null;` is an error, and Dart stops you *before the program runs*. Most languages discover this crash at run time, in front of the user; Dart discovers it while you type.

  (Roman Urdu: String wali cheez kabhi khali nahi ho sakti — yeh Dart ka wada hai.)
- Add a `?` to make emptiness legal: `String? nickname;` — "this box *may* be empty." The `?` is you honestly telling Dart: handle with care.
- `?.` is the safe call: `city?.length` means "if city has a value, give its length; if it is null, the whole answer is null — no crash." The chain stops quietly instead of exploding.
- `??` gives a **fallback**: `city ?? 'Unknown'` means "use city if it has a value, otherwise use 'Unknown'". You met it in the Map lesson (D.7); it is the polite way to replace emptiness.
- **The if-check:** inside `if (x != null) { ... }`, Dart *knows* x is safe and lets you use it as a normal value. This is called promotion — your check taught the compiler.
- `!` is the dangerous override: `name!` shouts "I swear this is not null!" If you are wrong, the program crashes on the spot. Use it only when you are truly sure — and prefer `?.`, `??` and if-checks, which can never crash.

Code examples

Example 1 — A box that may be empty, and one that may not

```
void main() {
  String name = 'Ali';
  String? nickname = null;

  print(name); // Ali
  print(nickname); // null — koi crash nahi
}
```

What it does: `name` is a promised String and holds text. `nickname` is a `String?` — the question mark makes `null` a legal value, so printing it simply shows the word `null`.

What you will see: two lines — `Ali` and `null`. Now delete the `?` and run again: Dart errors *before running*, telling you a normal String cannot be null. That early error is null safety doing its job — the bug died in your editor, not in a user's phone.

Example 2 — The two safe shortcuts: ?. and ??

```
void main() {
  String? city;

  print(city?.length); // null — ?. ne crash roka
  print(city ?? 'Unknown city'); // Unknown city

  city = 'Sahiwal';
  print(city?.length); // 7
  print(city ?? 'Unknown city'); // Sahiwal
}
```

What it does: While `city` is empty, `?.` returns null instead of crashing on `.length`, and `??` quietly swaps in the fallback text. After a value is assigned, the same two lines pass the real value through untouched.

What you will see: four lines — `null`, `Unknown city`, `7`, `Sahiwal`. Same code, two situations, zero crashes. Compare this with older languages, where the first line alone would kill the app — this is why Flutter apps feel solid.

Example 3 — The if-check: Dart learns from your check

```
void main() {
  String? message = 'Assalam o Alaikum';

  if (message != null) {
    // Yahan Dart ko yaqeen hai: message khali nahi
    print(message.length);
  } else {
    print('No message');
  }
}
```

What it does: The `if` asks the question once. Inside the true branch, Dart *promotes* message to a safe String, so plain `message.length` is allowed — no `?` needed. If message were null, the else branch would run instead.

What you will see: one line — `17`, the length of the message. Set message to `null` and run again: now `No message` prints. The check, not luck, decides — and Dart verifies both branches are safe before running either.

Quick recap

- `null` = no value at all (not 0, not '').
- Plain types (`String`) promise never-null; add `?` (`String?`) to allow emptiness honestly.
- `?.` = safe call (null in → null out, no crash). `??` = fallback value.
- Inside `if (x != null)`, Dart treats x as safe — promotion.
- `!` overrides the safety and *can* crash. Reach for `?.`, `??` and if-checks first.

Common mistakes

1. **Adding ? everywhere "just in case"** — a `?` on every type makes code messy and forces checks everywhere. Only values that can genuinely be empty (not-yet-filled, maybe-missing) deserve the `?`. Most variables should keep the never-null promise.
2. **Using ! to silence an error you do not understand** — `name!` does not fix the emptiness; it just removes the guard. If the value really is null, the app crashes at that exact line. When Dart complains, the honest fixes are `?.`, `??` or an if-check.
3. **Confusing null with empty text** — `''` is a real String of length 0; `null` is no String at all. `city?.length` gives `0` for the first and `null` for the second. Forms and user input produce empty text often — null rarely — so check for the one your code will actually meet.

Practice

1. Declare `String? fatherName;` (empty) and print a line that shows 'Not given' when it is empty and the name when it is set — using `??`. Then assign a name and run again.
2. Write a small program with `int? age;` that prints 'Age unknown' if age is null, and 'Adult' or 'Minor' (from Lesson D.4's ternary) if it has a value. Test both situations.

Practice quiz

1. What does String? nickname; allow that String nickname; does not?

- Longer text
- The box may be empty, null is legal
- Numbers as well as text
- Nothing, they are identical

Answer: The box may be empty, null is legal

2. What does city?.length do when city is null?

- It crashes
- It gives null quietly, no crash, the safe call
- It gives 0
- It gives an empty string

Answer: It gives null quietly, no crash, the safe call

3. What does city ?? 'Unknown' give when city is null?

- null
- 'Unknown', the fallback value
- An error
- An empty string

Answer: 'Unknown', the fallback value

Watch the video lesson: https://youtu.be/dq06NNxx6I8

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