Null Safety Flutter

Null safety prevents errors that result from unintentional access of variables set to null.

For example, if a method expects an integer but receives null, your app causes a runtime error. This type of error, a null dereference error, can be difficult to debug.

With sound null safety, all variables require a value. This means Dart considers all variables *non-nullable*. You can assign values of the declared type only, like int i=42. You can never assign a value of null to default variable types. To specify that a variable type can have a null value, add a ? after the type annotation: int? i. These specific types can contain either a null *or* a value of the defined type.

Sound null safety changes potential **runtime errors** into **edit-time** analysis errors. With null safety, the Dart analyzer and compilers flag if a non-nullable variable has either:

* Not been initialized with a non-null value
* Been assigned a null value. These checks allows you to fix these errors *before* deploying your app.

## Introduction through examples

With null safety, none of the variables in the following code can be null:

// With null safety, none of these can ever be null.

var i = 42; // Inferred to be an int.

String name = getFileName();

final b = Foo();

To indicate that a variable might have the value null, just add ? to its type declaration:

int? aNullableInt = null;

Null safety principles

Dart supports null safety using the following two core design principles.

Non-nullable by default. Unless you explicitly tell Dart that a variable can be null, it’s considered non-nullable. This default was chosen after research found that non-null was by far the most common choice in APIs.

Fully sound. Dart’s null safety is sound, which enables compiler optimizations. If the type system determines that something isn’t null, then that thing can never be null. Once you migrate your whole project and its dependencies to null safety, you reap the full benefits of soundness—not only fewer bugs, but smaller binaries and faster execution.