**FLUTTER**

* Flutter is a framework built by Google for developing applications in Android, iOS, web
* Flutter is built using Dart programming language.
* In flutter, we combine widgets to make a complete app.

Flutter vs React Native

* React native uses a middleware (JS) so its performance is slow.
* Flutter doesn’t use any middleware.
* Flutter has hot-reload functionality which enables, when any changes in code it will be reflected in application.
* It also has UI widgets

**DART**

* Dart uses hot reload feature in flutter.
* Hot reload is a tool - Reflects code changes in the app **without restarting** it. You can instantly see UI updates or logic changes while keeping the app state intact.
* Dart uses two major ways for program compilation, the Just-In-Time and the Ahead-of-Time method.
* A **JIT compiler (Just – In – Time)** converts program source code into machine code just before program execution. So initial runtime is slow. Initially, performance is also slower.
* An **AOT Compiler (Ahead Of Time)** converts program source code into machine code. It is used while deploying. The result is fast and runtime is better since machine code is generated before it is executed.

COMMENTS:

* // - Single line comment
  + // This is a single line comment.
* /\*…\*/ Multi – Line comment
* /\* These are

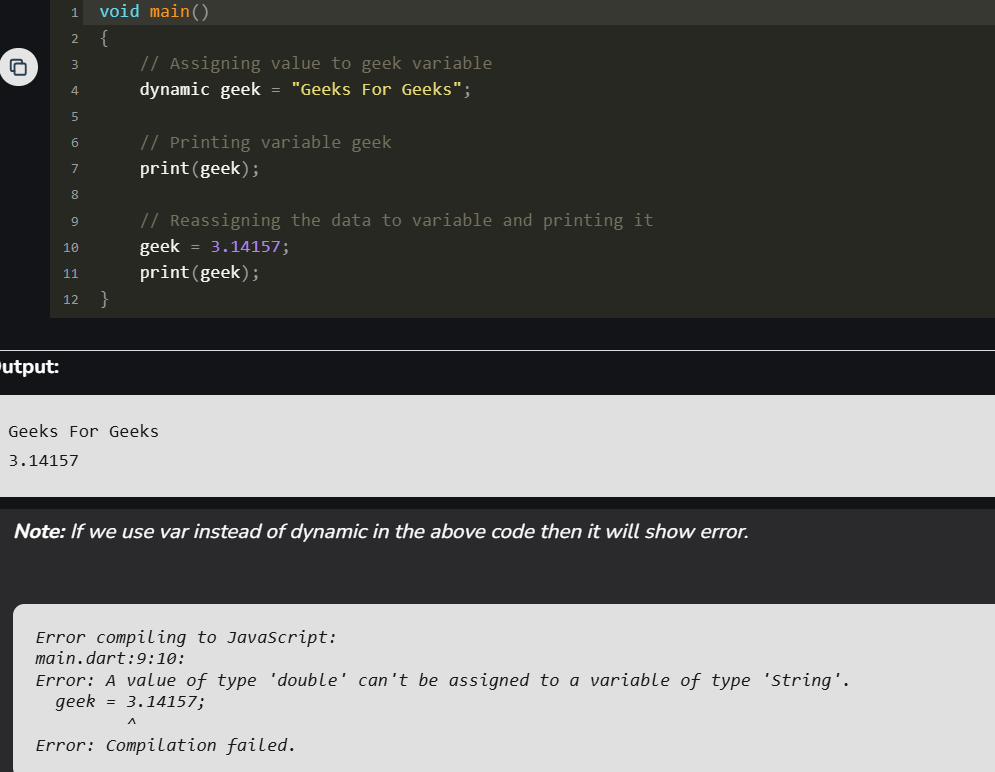
multiple line of comments

\*/

* /// Documentation Commenturns trueis even
* /// Returns true
* /// if n is even

VARIABLES:

* *SYNTAX : type variable1\_name, variable2\_name, variable3\_name,..*
* Dart supports type-checking, it means that it checks whether the data type and the data that variable holds are specific to that data or not.
* Types: **Static, dynamic, final/ const**
* Dynamic variable – SYNTAX **dynamic variable\_name.** Similar to var.
* Var vs dynamic :
  + Variable declared with **dynamic**, the datatype is assigned to it during runtime
  + Variable declared with **var,** datatype is assigned right after assigning the value to it.



* Final/ const– once a variable is defined using final, const keyword then its value can’t be changed in the entire code

**SYNTAX:** // Without datatype  
final variable\_name

Const var\_name  
  
// With datatype  
final data\_type variable\_name

Const data\_type var\_name

* Final – value assigned during runtime, suitable for dynamic or runtime value
* Const – value assigned during compile time, suitable for immutable, static values.
* EX: const time = DateTime.now(); 🡪 gives error, since it must be assigned during runtime.
* In Dart, by default a **variable can’t be assigned Null** value till it is defined that the variable can store Null value in it. This to avoid cases where user assign null value in Dart.
* To assign null, append **‘?'** to the type of the variable.

EXAMPLE: void main() {

int? a;

a=**null**;

print(a); 🡪 null

}

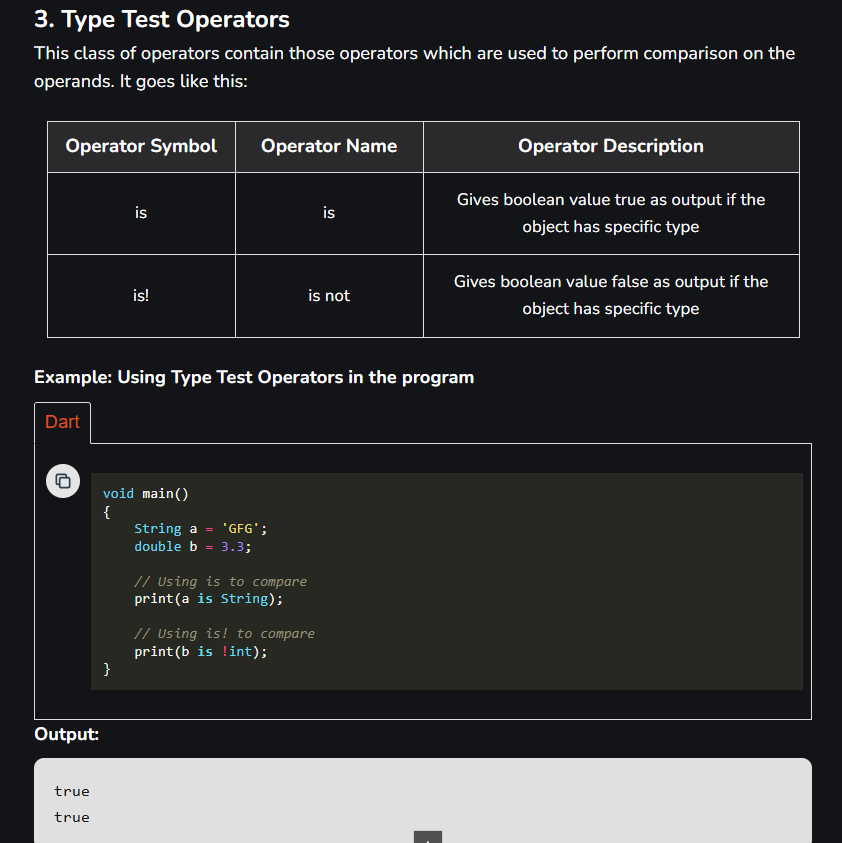
DATA OPERATORS:

* == operator can’t be used to check whether the objects are same or not. To check objects use **identical().**
* **ASSIGNMENT OPERATOR:**
  + = 🡪 Equal to
  + ??= 🡪 Assign value only if it is null;
  + EX: var sum = 7+5;

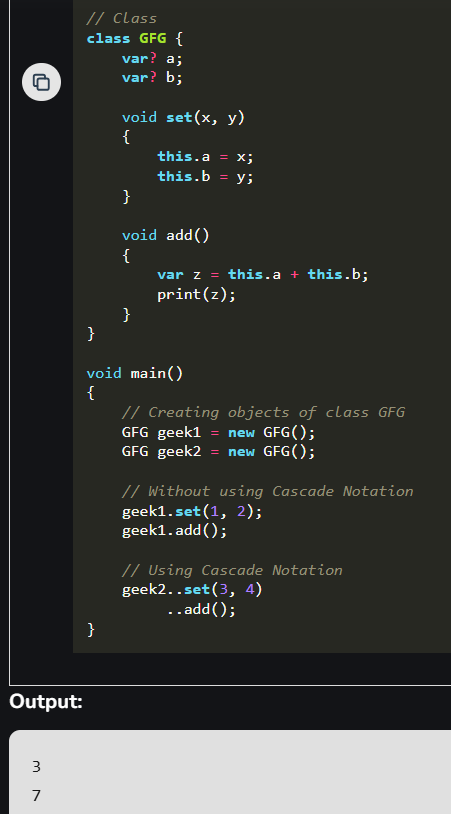
var d; //initially d is null;

d??=sum; //d=12;

d??=a-b; //d=12, since previously d=12 not null. So retains prev value



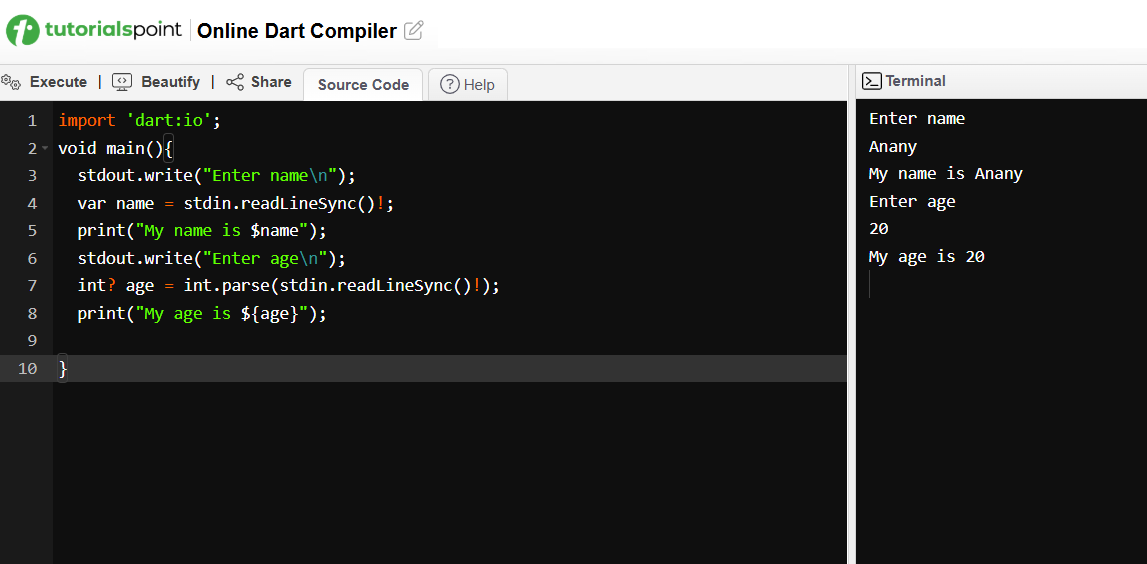
* .. 🡪 cascading operator is used to perform multiple methods on same object.



DART – STD INPUT OUTPUT:

* In Dart, you can take standard input from the user through the console via the use of .**readLineSync()** function.
* To take input from the console you need to import a library, named **dart:io** from libraries of Dart.

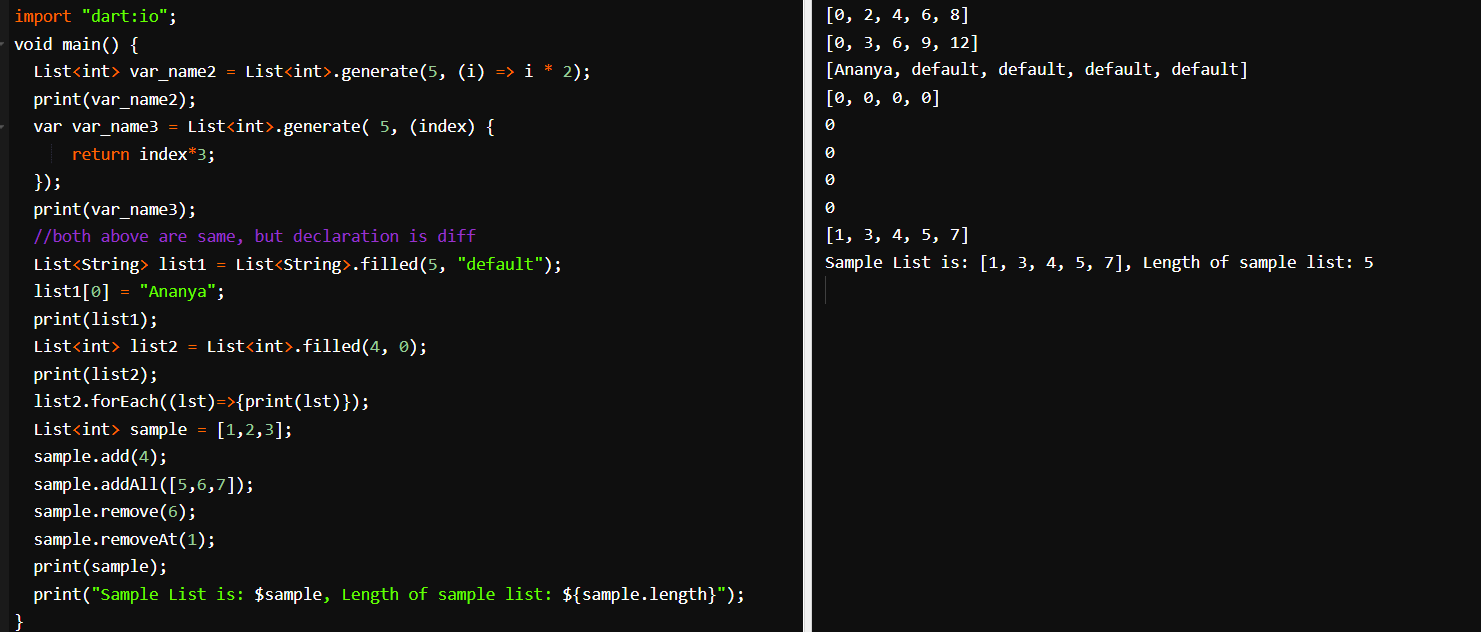




* int? n = int.parse(stdin.readLineSync()!);
  + stdin.readLineSync() 🡪 always returns **String?** Type
  + int? n 🡪 tells n can be either null or integer
  + ! 🡪 tells n will not be null, if null throw err.
  + Int.parse() 🡪 converts string to integer. If valid integer is not given it will show format exception error
* For printing we can use, **stdout.write() or print()**
* The print() statement brings the cursor to next line while stdout.write() don’t bring the cursor to the next line, it remains in the same line.

DART – DATA TYPES:

* **Number** – int, double, num (it can store either int or double)
* int.parse() 🡪 converts string into int. if given string is decimal, it shows format exception error.
* Num.parse() 🡪 converts string to int/ double. If given string is decimal it converts into double value.
* Both num.parse() and int.parse() throw a FormatException if the input string cannot be parsed into a valid number (e.g., "abc").
* **String str\_name**
* **bool var\_name**
* **List 🡪** similar to arrays. Used to represent a collection of objects
* **Variable size list:**
  + List<int> var\_name1 = [];
  + List<int> var\_name2;
  + var emptyList = <String>[];
* **Fixed Size List:**
  + List<int> var\_name1 = List<int>.filled(size,0);
    - Creates a list and initializes all with same value.
  + List<int> var\_name1 = List<int>.generate(size, (index)=> index\*2);
    - 2nd argument inside generate is a generator function. It takes only one argument mostly index. Depends on size, the generator function runs those many times.
    - Creates a list and generate each element based on index.

****

* **Map 🡪** it is a key and value pair. Keys, values maybe any type.
* **Declaring empty map:**
  + Map? map\_name;
  + Map<key\_datatype, value\_datatype>? map\_name;
  + var map\_name = new Map();
* **Declaring map with elements inside it:**
  + Map m1 = {k1:v1; k2:v2};
  + Map<String, String> m2 = {“k1”:”v1”; “k2”:”v2”};
  + var m3{k1:v1; k2:v2}