Dart Programming - Dart Basics

Variables, Built-in types, Operators, Control flow

1. Variables:

Variables are used for storing data.

Example: double num1; // the num1 is variable that is being stored as a return type ‘double’. This means that any value that will be assigned to num1 must be a floating-point number.

num1 = 12.32;

Other return types include ‘int’ for integers, ‘String’ for strings, ‘bool’ for Boolean values (true/false) – this aspect will be further explained later.

Variables can be reassigned unless they are declared as final or const.

final String name = ‘Joshua’; //the value for variables declared with final can be set only once and it is determined at runtime.

const String sexBirth = ‘male’; //the value for variables declared with const is immutable and set at compile-time.

If the type is unknown or will change in the future, the variable can be declared as dynamic.

dynamic myMind = 4; //integer

myMind = ‘four’; //now a string

From Dart 2.12, variables are non-nullable by default unless they are marked by ?.

String nullableName = null; //nullable

String nonNullableName = ‘Joshua’ //cannot be null

late is used when a variable will be initialized later.

late double num1;

num1 = 6.08;

2. Built-in types

a. Numbers

int: for integer values e.g., 10, -60

double: floating-point numbers (as discussed earlier).

b. Strings

String represents text wrapped with single (‘) or double (“) quotes. String interpolation can be done with $ or ${expression}.

String message = ‘Hello, $name’;

String result = “sum: ${2 + 3}”;

c. bool is used for representing true or false

bool isCorrect = true;

d. Lists

used for ordered collections. The values do not need to be unique.

List<int> numbers = [1, 2, 3, 3]; //1 is the 0^{th} value and the rest follow suit

e. Sets

These are unordered collections of unique items

Set<String> colors = {‘red’, ‘blue’};

colors.add(‘green’); //adds only if unique

f. Maps

Used for declaring key-value pairs, that is, Map<k, v> or {}.

Map<String, int> scores = {‘Alisson’: 90, “Bola”: 98};

var map = {‘key1’: ‘value1’};

g. Runes

They are used for emojis or special characters. They represent Unicode code points for strings.

Runes emoji = Runes(“\u{1f600}”);

3. Operators In Dart

There are various operators used for performing operations on variables and values in Dart.

a. Arithmetic Operators:

+, -, *, /, ~/ (integer division), % (modulus).

int result = 10 + 5;

b. Relational Operators:

==, !=, >, <, >=, <=

bool isEqual = (5==5); //true

c. Logical Operators

&& (and), || (or), ! (not)

bool result = (x > 0 && y < 10);

d. Bitwise Operators:

& (and), | (or), ^(xor), ~(not), <<(left shift), >> (right shift).

int result = 5 & 3 ; //1

e. Assignment operators:

=, +=, -=, *=, /=, ??= (assign if null).

x +=2; // x = x + 2

f. Conditional (ternary) operator:

condition ? expr1 : expr2

String status = age >= 18 ? "adult" : "minor";

g. null-aware operators:

?.(conditional access), ?? (default value), ??= (assign if null)

String? name;

print(name ?? "Guest");

h. Type test operators:

is, is! (check type)

if (obj is String) print(“It is a String”);

i. Cascade operator (..):

Allows multiple operations on the same object.

var list = [1, 2]..add(3)..add(4) // [1,2,3,4]

4. Control flow

These are constructs to control the flow of program execution.

a. Conditional statements:

```
if (age >= 18) {
  print("Adult");
} else if (age >= 13) {
  print("Teen");
} else {
  print("Child");
}
```

b. Switch Statement:
This is used to set multiple conditions based on a single value

```
String grade = "A";
switch (grade) {
  case "A":
    print("Excellent");
    break;
  case "B":
    print("Good");
    break;
  default:
    print("Unknown");
}
```

c. Loops:

i. for Loop

```
for (int i = 0; i < 5; i++) {
```

```
  print(i); // prints 0 to 4
 }
```

ii. for-in Loop

```
var list = [1, 2, 3];
for (var item in list) {
  print(item);
}
```

iii. while Loop

```
int i = 0;
while (i < 5) {
  print(i);
  i++;
}
```

iv. do-while loop

```
int i = 0;
do {
  print(i);
  i++;
} while (i < 5);
```

d. Break and continue:

“break” is used for exiting a loop or switch while “continue” skips the current iteration and continues with the next.

```
for (int i = 0; i < 5; i++) {
  if (i == 3) break; // stops at 3
  print(i);
}
```

e. Assert:

This is used for debugging. It throws an error if the condition is false and is only utilized in debug mode.

```
assert(age >= 0, “Age cannot be negative”);
```

f. Exception Handling

try, catch, finally are used for handling errors.

```
try {
  var result = 10 ~/ 0; // division by 0 **// Added semicolon**
} catch (e) {
  print("Error: $e");
} finally {
  print("cleanup done");
}
```

- specific exceptions can be caught

```
catch (e) {
  if (e is IntegerDivisionByZeroException) {
    print("cannot divide by zero");
  }
}
```

- Exceptions can also be manually thrown

```
throw Exception(“Something went wrong”);
```

Summary

Variables are used for storing data with optional types. They support null safety and can include final/const for immutability.

Dart is endowed with built-in types (int, double, String, bool, list, set, Map, Runes, and Symbol).

The following operators: Arithmetic, relational, logical, bitwise, assignment, null-aware, and cascade operators are used for concise operations.

For control flow: Conditional (if, switch), loops (for, while, do-while), exception handling and break/continue for program flow control.