Beginning Programming with Dart

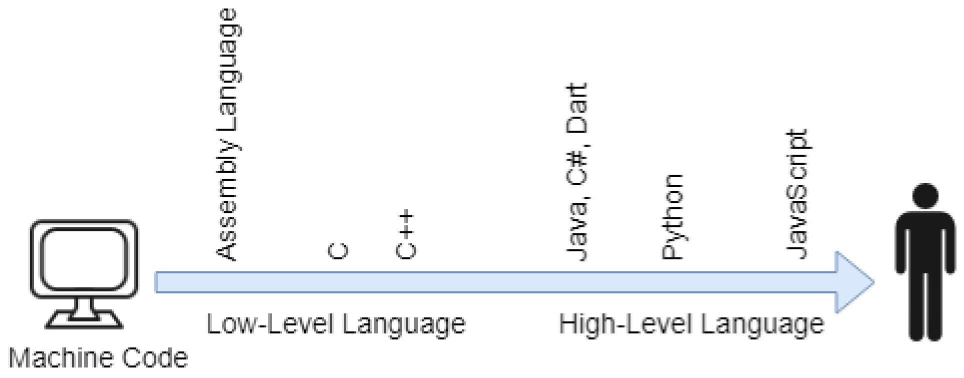
Sharad Ghimire Follow
Mar 2 · 15 min read ★

Dart is created by Google, and its primary purpose is to leverage C-based languages like C++, C#, and Java. It is a general-purpose programming language that is fast in compile time, concise and easy to learn. Dart is purely object-oriented, optionally typed, a class-based language which has excellent support for functional as well as reactive another.

These instructions are maybe 5 lines or maybe 1000, or maybe be even 500 million, each of which is telling our computer system to do something very tiny, but specific, something like 'divide 10 by 2'. And that's why programming is interesting (at least for me) because of the way we take a bigger idea or problem and break it down into these smaller tiny pieces, and start building from the ground up. Okay, that might be a little abstract, so, let's consider a small real-life example of programming.

Let's imagine, you are all alone in your house and you are hungry but don't know how to cook. Suddenly you remembered that you have a Ramyun noodle in your kitchen. So, you called your mother asking her how to make a Ramyun Noodle. She gave you some step by step guides on how to cook it. So, as you can see, cooking is a journey from point A which is unpacking those noodles to point B which is pouring it down in a bowl, and thanks to your mother who broke it down into simpler steps and helped you so that you can easily understand. Now, as you already noticed, sequence here plays a vital role. If by any change if you mixed the execution flow of those steps, for example, if you put noodles on top of the stove before putting in the cooking pan, you will totally create something else 😅.

So with programming, we are giving steps to the computer, obviously not to cook but to perform a specific task. It's breaking apart a more complex problem, into its smallest individual instructions and then using a programming language to write those instructions. Just like English, programming language also has some syntax we need to follow so that they understand exactly what we want them to do. Just like a sentence in English, the programming language has **statements**. Although these statements may different types of CPU. Below is the diagram showing different types of languages that are closer to understand for CPU and harder for the human being and vice versa.



Why Dart?

I came across this language when I had to develop an e-commerce application for my client called BestBikes. It was a simple e-commerce application for buying and selling second-hand bikes for customers. The client wanted a web application for customers and a mobile application for the seller. The web application was a major part of the project while the mobile application was not, it was just for the seller so that they can take the pictures of bikes easily and upload them. So, we decided to use Flutter for it. Flutter is the mobile development framework also developed by Google which uses Dart as its programming language and its very fast for prototyping. As I came from Java and JavaScript background, learning Dart was really easy and filled with excitement for me. It's easy to use, fast and the learning curve was very less compared to other languages like C# and Java. And I really believe it is the future of cross-platform development as we can develop web, mobile and even server-side applications using Dart.

Core Syntax

Here is the simple Hello World program and technically a complete dart program.

```
// main() is the entry point of our dart program
main() {
    //print function to print the text to console
   print("Hello World");
}
```

This simple piece of code is self-explanatory. Execution of a dart program begins with a call to a function called `main()` .

Variables and Data Types

In every program we write, we have to keep track of some pieces of information. For example, in simple Flappy Bird game, we need to store the score of a current player, high score, etc. This is a simple piece of data and we create variable to store them. Variables are simply a block of memory in our system that we give our custom name that can be recalled later, and then we change the value we need. Moreover, there is another term we need to know called **datatype**. Data types are simply a type of data. Some common data types are integers, floating point, strings and arrays. They can also be more specific like date, booleans and timestamps. However, data types in dart are objects, therefore their initial value is by default `null` literary a zero. We will cover that later. Let's create some variable and learn how to use them. Dart has special support for following data types:

- Number (includes `int` and `double`)
- Strings

Note: Dart is a statically typed language meaning it guarantees that a variable of one type cannot produce a value of another type.

```
main(){
  // It is inferred as integer automatically at runtime
  var age = 10;
  // OR
  int age = 10;
  int hexaValue = 0xEADEEAE;
  double percentage = 13.0;
  bool isStudent = true;
}
```

Let's define the above syntax:

- `//This is a comment` A single line comment which is useful for notes to programmers. These comments are not executed in our final program.
- `10` A number literal. `Sharad Ghimire` A string literal. Both quotes are supported in dart. `'...' or "..."`
- `int` Integer data types can store all those integer number values which are numbers without decimal points.
- `var` A way to declare a variable without specifying its type.
- `main()` This is a special function provided by dart which is required. This is the top-level function where our application execution starts.
- `bool` Boolean is a special data type which contains only two values i.e `true` and

```
//Long String
String s4 = 'Lorem Ipsum is simply dummy text of the printing.'
            'standard dummy text ever since the 1500s, when an'
            'five centuries, but also the leap into electronic'
            'remaining essentially unchanged.';

// String Interpolation
String name = 'Sharad';
String message = 'My name is' + name;
String messsage = 'My name is $name';
String messsage = 'My name\'s length is ${name.length}';
```

- `$variable OR ${expression}` String interpolation: This meaning including a variable or expression's string equivalent inside a string literal. An **expression** is simply a combination of one or more variables, operators, and functions that computes to produce another value.
- `'What\'s'` This is called an escape character. When we have to write a single quote inside a string literal with a single quote, we use it as `\` .

Arithmetic Operator	Meaning
+	Addition (Also used for string concatenation)
-	Subtraction Operator
*	Multiplication Operator
/	Division Operator
%	Remainder Operator
Unary Operator	Meaning
++	Increment Operator (increments value buy 1)

Logical Operators	Meaning
||	OR (true if either of the boolean expression is true)
&&	AND (true if all boolean expressions are true)

Different Types of Operator in Dart

- Dart does has operator precedence, i.e some operators are treated as more important than others. For example, `*` is used first then `+` operator. Similar to Mathematics. We can make sure some operator evaluate first than others using parentheses. `(100 – 20)*2` .
- `x += 10` This simply means whatever the value of the score, add 10 to it. `x = x + 10;` This operation is so frequent that there is a shortcut way of doing it, i.e `+=` . Similarly, we have `+=, -=, *=, /=` .
- `x++` If we want to add 1 to a value we can simplify it by `x++ or x += 1` instead of writing the full statement.

Working with Conditional Code

Till now, we have been writing simple one line code. All the code we have written execute from top to bottom. What if we want them to execute if certain condition meets. For example, let’s get back to that Ramyun noodle example. What if your mother told you to put some onions in there and you don’t have any? Are you gonna wait for onions till someone actually brought it and carried it to you, or you gonna ignore onions and proceed to the next step in the recipe? Proceed right. Here you made some decision based on some condition. As same in programming, we may have to take decision-based on some calculation. For example, if the player’s current score is greater than a high score, show “You have achieved high score”. That's called Conditional Programming.

`if (condition) { ... }` If some condition is true, then do whatever you wanna do that goes between two braces `{}` .

```
// if else if  statement
var marks = 70;

if(marks >= 90 && marks < 100){
    print("Something");
} else if(marks >= 80 && marks < 90){
    print("Something eLse ");
} else if(marks >= 70 && marks < 80){
    print("Anything ");
} else if(marks >= 60 && marks < 70){
    print("Nothing");
} else {
    print("Invalid!");
}
```

`if(condition1){...} else if(condition2) {...} else {...}` These means do the first block of code if `condition1` is true, the second block if `condition2` is true and if both are not true then do the last block of code.

```
int a = 2;
int b = 3;
a < b ? print(a) : print(b); // Prints a

int smallNumber =  a < b ? a : b;

String name = null;
String nameToPrint = name ?? "Guest User";
print(nameToPrint); // Prints "Guest User";
```

```
void main() {
  String name = 'Sharad';
  switch (name) {
    case: 'Sharad':
      print("You are Sharad!");
      break;

    case: 'Pramish':
      print("You are Pramish!");
      break;

    default:
      print("Your name is not in my system!");
  }
}
```

Sometimes there's may a specific situation where it's not the only option for checking a condition maybe we're checking a variable for a specific value. We can do it using if else condition but will become long pretty soon. In that case, `switch case` is the better option.

Writing Control Statement

Let's say we write a simple program to print the name of the current user. Then again we figured out we need to do it again so we copied it one more time. Now, again we need to print that user name 100 times. What would we do? Do we copy that line of code 100 times? In these scenarios, the control statement comes into play. We can also hear people call it Loops or Iterations. All our loops have conditions that control how long we need to loop and when to stop.

```
/// Do loop
```

```
    print value;
    value++;
}
// 1 2 3 4 5

// Infinite Loop
do {
    print('Value');
    value++;

    if(value == 3){
        print('value is 3');
        continue; // Continue the loop
    }

    if(value > 5){
         print('value is greater than 5');
         break;  // jumps out of the loop
    }

} while(true);

// For Each
List people = ['Sharad', 'Ram', 'Hari'];

// Starting value, range, increment
for(int i = 0; i < people.length; i++){
    print("Person at ${i} is ${person[i]}");
}

people.forEach((String person) {
    print(person);
});
```

- `do{ ...} while(condition);` Do loop will always run its block first while the `while(condition){ ... }` loop evaluates first before running.
- `break;` Break jumps out of the loop and stops executing that loop. While `continue;`

After writing statements and conditional code, our code starts to become long and dirty and harder to understand. So, we break them apart into smaller chunks called functions or methods in dart. These make our code easy to understand and also reusable. A function is simply a chunk of code which we then wrap up using braces and then call them whenever we want. To define them, functions are a collection of statements grouped together to perform an operation. All functions in Dart returns a value. If no return value is specified the function return `null`. Specifying return type is optional but is recommended as per code convention.

```
void main(){
    print(findArea(4, 5));
    findPerimeter(4,5);
}

int findPerimeter(int length, int breadth){ // Required Parameter
    return length + breadth;
}

void findArea(int length, int breadth){
    print(length * breadth);
}

findArea(int length, int breadth){
    // By default, if no return value is specified, function
     returns null
}
```

- `int findPerimeter()` This is a function definition that returns `int`. If the function returns nothing we should write `void` as a return type. If we have to call this function we simply write the function name with parentheses which means calling

```
// Anonymous Functions (internal functions)
() { print('Hello'); } // Nothing happens because its not invoked

List people = [ 'Sharad', 'Ghimire']; // We will cover List later
people.forEach(print);
people.forEach(() {
    print(name);
});
```

- Use fat arrow for single line functions and remove `return` keyword and `{ }`. *Just One Expression function.*
- Anonymous functions are functions that are dynamically declared at runtime. They're called **a**nonymous functions because they aren't given a name in the same way as normal functions. If the function is only used once, or a limited number of times, it may be syntactically lighter than using a named function. Like we are printing all people using an anonymous function.
- There are two types of parameters in Dart: Required parameters and Optional Parameters. Under optional parameter, there are another three types called optional positional parameters, optional named parameters and optional default parameters. Named parameter prevents errors if there are a large number of parameters.

```
void main(){
  printCities("Dang", "Tulsipur", "Ghorahi"); // Must have all 3 arg
  printCountries("Nepal", "India"); // Will get null in s3
  print(findVolume(2, h: 10, b: 3)); // Sequence does not matter
  var result = findArea(length: 2);
}

// Required Parameters
```

Get one more story in your member preview when you sign up. It's free.

Sign up with Google

Sign up with Facebook

Already have an account? Sign in

Collections

A collection is an idea of storing values just like variables but it can store multiple values and all those values are contained in one named variable. It is a great way of keeping data together that belong together. For example, names of countries, or breed of dogs, and so on. In Dart, List and Sets are part of the core library so we don't have to import those. For others, we need to import from another package. Importing concept will be explained later.

```
import 'dart:collection';

//enum
enum colors { red, green , blue }

main(List<String> arguments) {
  print(colors.values); // [colors.red, colors.green, colors.blue]
  print(colors.red);

  List test = [1, 2, 3, 4]; // Fixed length list
  print(test.length);
  print(test[0]);  // Gives item at index 0
  print(test.elementAt(2)); //3

  // Growable List of generic type
  List things = new List();
  things.add(1);
  things.add('cats');
  things.add(true);

  List<int> numbers = new List<int>();
  //new creates a new object in memory
  numbers.add(1);

  Set<int> numbers = new Set<int>();
  numbers.add(1);
  numbers.add(2);
```

```
    Map people = { 'firstname': 'Sharad', 'lastname': 'Ghimire' };
    print(people.keys); // (firstname, lastname)
    print(people.values); // (Sharad, Ghimire)
    print(people['firstname']); // Sharad

    Map<String, String> people = new Map<String, String>();
    people.putIfAbsent('firstname', () => 'Sharad');
  }
```

- **Enum**: Enums are simply a list of constants. When we need a predefined list of values we use enums. For example, Names of days, or months, etc. In dart, we cannot put enum inside the main function. It is the simplest of the collection.
- **List**: List is an indexable collection of values with a length. The list has a zero-based index. There are two types of list, Fixed length and Growable list. `List<int> numbers = new List<int>();` Creating a variable type of generic of int (List that only has int).
- **Set**: Set can store certain values, without any particular order, and no repeated values.
- **Queue**: Queue can store ordered, has no index, and add and remove operation can be done from the start and end. It is not a part of the standard collection so we need to import it. `import 'dart:collection';` .
- **Map**: Map can store key/value pairs. They are composed of a collection of (key, value) pairs, such that each possible key appears at most once in the collection. They can add items and we do not need to know their index and just need to know keys which can be any datatype (esp. String and int).

Note: List is the most basic and most used collection so we will heavily use it. Other

```
// Error is a program failure
// Exception are errors that  can be handled

int age;
int years = 7;
print(age * years); //Unhandled exception: NoSuchMethodError: The
method '*' was called on null.

// Try Catch Finally
try {
    int age;
    int years = 7;
    print(age * years);
}
on NoSuchMethodError { // Catch specific error
    print('Sorry thats not gonna happen');
}
catch(e) {
    print("There was an error: ${e.toString()}");
}
finally {  // Clean up
   print("Compete");
}

// Throwing Exception
try {
   int age;
   int years = 7;
   if(years != 7) throw new Exception('Years must be 7'); // Custom
exception
   if(age == null) throw new NullThrowError();
   print(age * dogyears);
}
on NullThrownError{
    print("The value was null!!");
}
on NoSuchMethodError {
    print('Sorry no such method!');
}
catch(e) {
```

network failure or not available, and release the resources in the finally block like close that database.

- We can also catch a specific exception by using `on` keyword.

Conclusion

You are now a beginner programmer in Dart Language. This is just the first part of the 3 part series. In the second part, we will go deeper into Classes and Generics and also File Systems. And finally, in the last advanced part, we will discuss Asynchronous Programming, Functional Programming, Reactive Programming and Database Programming and much more. For now, have a good weekend.

. . .

Read this story later in Journal.

Wake up every Sunday morning to the week's most noteworthy Tech stories, opinions, and news waiting in your inbox: Get the noteworthy newsletter >