Functions

Functions are the building blocks of readable, maintainable, and reusable code. A function is a set of statements to perform a specific task. Functions organize the program into logical blocks of code. Once defined, functions may be called to access code.

```
String sayHelloWorld() {
    return "Hello, World!";
}
void main(){
sayHelloWorld();
}
```

Function with parameters

```
void main(){
  var sum = add(10,20);
  print("Sum Of Given No. Is : ${sum}");
}

int add(int n1, int n2){
    int result;
    result = n1+n2;
    return result;
}
```

Arrow Function

```
void main() => print(add(1,3));

int add(int n1, int n2){
  int result;
  result = n1+n2;
  return result;
}
```

Classes

A **class** is used in object-oriented programming to describe one or more objects. It serves as a template for creating, or instantiating, specific objects within a program. ... While the syntax of a **class definition** varies between programming languages, **classes** serve the same purpose in each language.

Pillars of Object Oriented Programming:

1) Inheritance
2) Encapsulation
3) Abstraction
4) Polymorphism

1) Inheritance

The ability of creating a new class from an existing class. Inheritance is when an object acquires the property of another object. Inheritance allows a class (subclass) to acquire the properties and behavior of another class (super-class). It helps to reuse, customize and enhance the existing code. So it helps to write a code accurately and reduce the development time.

Dart

2) Encapsulation

Encapsulation means wrapping up data and member function (Method) together into a single unit i.e. class. Encapsulation automatically achieve the concept of data hiding providing security to data by making the variable as private and expose the property to access the private data which would be public.

3) Abstraction

Abstraction is the process of showing only essential/necessary features of an entity/object to the outside world and hide the other irrelevant information.
In programming language we achieve the abstraction through public and private access modifiers and a class. So in a class make things (feature) which we want to show as public and thing which are irrelevant make them as private so they won't be available to the outside world.

Dart

4) Polymorphism

Polymorphism is when you use one block of code, you change the version of the code being used based on the inputs being giving to it. So to make it a bit clearer, different classes can be used with the same interface but can provide its own implementation of that interface.

Example: Real life example of Polymorphism is mobile phone. It is a single object but it can be used for making calls, listening music, sending mails, taking pictures, etc (different forms).

Classes Example

```
void main() {
  var car1 = Car();
   car1.name = "Prius";
   car1.disp();

   var car2 = Car();
   car2.name = "Civic";
   car2.disp();
}

class Car {
   String name = "";
   void disp() {
      print("This car is ${name}")  ;
   }
}
```

Class Inheritance

Dart supports the concept of Inheritance which is the ability of a program to create new classes from an existing class. The class that is extended to create newer classes is called the parent class/super class. The newly created classes are called the child/sub classes.

A class inherits from another class using the ‘extends’ keyword. Child classes inherit all properties and methods except constructors from the parent class.

```
void main() {
   var cirr =  Circle();
   cirr.calarea();
    cirr.cir();

   var sqr =  Square();
   sqr.calarea();
    sqr.sq();
}

class Shape {
   void calarea() {
      print("calling calc area defined in the Shape class");
   }
}

class Circle extends Shape {
  void cir(){
    print("Circle Class");
  }
}

class Square extends Shape {
  void sq(){
    print("Square Class");
  }
}
```