What is dart programming language?

Dart is a versatile, object-oriented programming language developed by Google. It is designed to be easy to learn and use, and it offers several features that make it suitable for various programming tasks, especially in building modern applications.

Features of Dart Programming

**Object Oriented:**

Dart is an object-oriented language, meaning it uses classes and objects to organize code. This approach helps in creating reusable and modular code.

**Strongly Typed:**

Dart is statically typed, which means that types are checked at compile-time. This helps catch errors early in the development process.

**Asynchronous Programming:**

Dart supports asynchronous programming using Future and async/await, which is essential for building responsive applications that perform tasks like network requests or file operations without blocking the main thread.

**Just-In-Time (JIT) and Ahead-Of-Time (AOT) Compilation**:

Dart can be compiled both Just-In-Time (JIT) during development for faster iteration and Ahead-Of-Time (AOT) for optimized performance in production. This dual compilation approach helps in balancing development speed and runtime performance.

**Hot Reload**

With frameworks like Flutter, Dart supports hot reload, which allows developers to see changes in the code immediately without restarting the application. This feature greatly accelerates development and testing.

**Cross-Platform Development**:

Dart is the language behind Flutter, Google’s UI toolkit for building natively compiled applications for mobile, web, and desktop from a single codebase. This allows developers to write once and deploy across multiple platforms.

**Rich Standard Library**:

Dart comes with a rich standard library that provides a wide range of functionality, from collections and asynchronous programming to file I/O and HTTP requests.

**Null Safety**:

Null Safety in simple words means a variable cannot contain a ‘null’ value unless you initialized with null to that variable. With null safety, all the runtime null-dereference errors will now be shown in compile time.

Example Code: main()

{

int num;

print(num);

}

Compile Time Error: main.dart:5:10: Error: Non-nullable variable 'num' must be assigned before it can be used.

print(num);

^^^

**Use of Dart Programming Language**

**Mobile Development**:

Dart is primarily known for its use with Flutter to build cross-platform mobile applications for both iOS and Android.

**Web Development**:

Dart can be used to build web applications. It can be compiled to JavaScript, allowing Dart code to run in web browsers.

**Desktop Applications**:

With Flutter, Dart is also used to create desktop applications for Windows, macOS, and Linux.

**Server-Side Development**:

Although less common, Dart can be used for server-side programming with frameworks like aqueduct (though development on aqueduct has ceased, other frameworks and libraries are available).

Data type in Dart Programming Language:

1. Number
2. String
3. Boolean
4. Lists
5. Sets
6. Maps
7. Runes
8. Nullability
9. Dynamic.
10. Custom Size.
11. Type Inference

**What is List in Dart?**

In Dart, arrays are represented by the List class. There are several ways to create and initialize lists in Dart, depending on your needs. Here’s a comprehensive guide to different types of list creation in Dart:

**Check Datatype in dart:**

Example: void main()  
{  
 int num = 10;  
  
 print('Data type: ${num.runtimeType}');  
}

**Types of List creation in dart**

1. Empty List Creation.
2. List with Initial Elements.
3. Fixed-Length List.
4. List with Different Initial Values.
5. List from Another Iterable.
6. List with Typed Elements.
7. List with Mixed Types.
8. Growable List vs Fixed-Length List.

**Sets in dart**

In Dart programming, a Set is a collection of unique items. Unlike lists or arrays, sets do not allow duplicate elements and are not ordered. This means that the elements in a set are not indexed, and their order may not be preserved.

There are two ways to create sets in dart

1. var variable\_name = <variable\_type>{};
2. set <variable\_type> variable\_name = {};

**Dart Object-Oriented Concepts**

Dart is an object-oriented programming language, and it supports all the concepts of object-oriented programming such as classes, object, inheritance, mixin, and abstract classes. As the name suggests, it focuses on the object and objects are the real-life entities.

Note: The Object-oriented programming approach is used to implement the concept like polymorphism, data-hiding, etc.

Note: The main goal of oops is to reduce programming complexity and do several tasks simultaneously.

**Class**

**In dart** classes are defined as the blueprint of the associated objects. A Class is a user-defined data type that describes the characteristics and behaviour of it. To get all properties of the class, we must create an object of that class.

**Object**

An object is a real-life entity such as a table, human, car, etc. The object has two characteristics - state and behaviour. Let's take an example of a car which has a name, model name, price and behavior moving, stopping, etc. The object-oriented programming offers to identify the state and behavior of the object.

**Types of Constructors**

There are three types of constructors in Dart as given below.

* Default Constructor or no-arg Constructor
* Parameter Constructor
* Named Constructor

**Dart this Keyword**

The this keyword is used to refer the current class object. It indicates the current instance of the class, methods, or constructor. It can be also used to call the current class methods or constructors.

**this** keyword represents an implicit object pointing to the current class object. It refers to the current instance of the class in a method or constructor. The **this** keyword is mainly used to eliminate the ambiguity between class attributes and parameters with the same name. When the class attributes and the parameter names are the same this keyword is used to avoid ambiguity by prefixing class attributes with *this* keyword. **this** keyword can be used to refer to any member of the current object from within an instance method or a constructor

**Uses of this Keyword**

1. It can be used to refer to the instance variable of the current class
2. It can be used to make or initiate current class constructor
3. It can be passed as an argument in the method call
4. It can be passed as an argument in the constructor call
5. It can be used to make a current class method
6. It can be used to return the current class Instance

**Dart super Keyword**

The super keyword is used to denote the instant parent class object of the current child class. It is used to invoke superclass methods, superclass constructor in its child class. The super keyword's main objective is to remove the confusion between parent class and subclass with the same method name. It is also used to refer to the superclass properties and methods.

**Inheritance**

Dart supports inheritance, which is used to create new classes from an existing class. The class that to be extended is called parent /superclass, and the newly created class is called child/subclass. Dart provides extends keyword to inherit the properties of parent class in child class.

**Types of Inheritance:**

1. **Single Inheritance:** This inheritance occurs when a class inherits a single-parent class.
2. **Multiple Inheritance:** This inheritance occurs when a class inherits more than one parent class. **Dart doesn’t support this.**
3. **Multi-Level Inheritance:** This inheritance occurs when a class inherits another child class.
4. **Hierarchical Inheritance:** More than one classes have the same parent class.

There are two types of methods in Dart:

1. **Instance Method**

A method that can be accessed by using the instance of class is called instance methods. The instance methods can be no arguments or with arguments. The instance method can access by instance variable this keyword.

**Syntax:**

return\_type method\_name(<list of arguments>)

{  //statement(s)

}

1. **Class Method**

The class method is declared with the static keyword. It can be accessed by using the class name instead of the class object. These methods are common to all instances of that individual class. The static methods only can access the static variables.

In Dart, class methods are static methods defined using the static keyword. They belong to the class itself rather than an instance of the class. This means you can call these methods without creating an instance of the class.

**Syntax:**

static return\_type method\_name(){    //statement(s)

 }

**What is Polymorphism?**

The polymorphism is a combination of the two Greek words **poly,** which means **many** and morph means **morphing into different forms or shapes**. Together, polymorphism means the same entity can be used in various forms. In the programming aspect, the same method can be used in different classes. This technique makes programming more intuitive and more accessible.

Different method to approach polymorphism

1. Method Overriding.
2. Abstract Classes and Methods.
3. Interfaces.
4. Duck Typing.

**Method Overriding**

When we declare the same method in the subclass, which is previously defined in the superclass is known as the method overriding. The subclass can define the same method by providing its own implementation, which is already exists in the superclass. The method in the superclass is called **method overridden,** and method in the subclass is **called method overriding**.

**Rules of Method overriding in Dart**

The few rules of method overriding are given below. These points must be kept in mind while declaring the same method in subclass.

1. The overriding method (the child class method) must be declared with the same configuration as the overridden method (the superclass method). The return type, list of arguments and its sequence must be the same as the parent class method.
2. The overriding method must be defined in the subclass, not in the same class.
3. The static and final method cannot be inherited in the subclass as they are accessible in their own class
4. The constructor of the superclass cannot be inherited in a subclass.
5. A method that cannot be inherited, then it cannot be overridden.

**Dart Abstract Classes**

Abstract classes are the classes in Dart that has one or more abstract method. Abstraction is a part of the data encapsulation where the actual internal working of the function hides from the users. They interact only with external functionality. We can declare the abstract class by using the abstract keyword. There is a possibility that an abstract class may or may not have abstract methods.

**Rules for Abstract classes:**

The rules of the abstract are given below.

1. An abstract class can have an abstract method (method without implementation), or not.
2. If there is at least one abstract method, then the class must be declared abstract.
3. The object of the abstract class cannot be created, but it can be extended.
4. An abstract keyword is used to declare the abstract class.
5. An abstract class can also include normal or concrete (method with the body) methods.
6. All abstract methods of parent class must be implemented in the subclass.

**Syntax:**

1. abstract class ClassName {
2. // Body of abstract class
3. }