1 What is flutter-who developed when

Flutter is open source frame work devlp by google for building beautiful and  natively complied multi platform application using single code base  May 2017

2  What is Dart?

Dart is a general-purpose, object-oriented programming language with C-style syntax. It is open-source and developed by Google in 2011. The purpose of Dart programming is to create a frontend user interfaces for the web and mobile apps. It is an important language for creating Flutter apps. The Dart language can be compiled both AOT (Ahead-of-Time) and JIT (Just-in-Time.

3 ) What are the best editors for Flutter development?

 Android Studio

 Visual Studio

 Xcode

4)What is pubspec.yaml file?

It is the project's configuration file that will use a lot during working with the Flutter project. It allows you how your application works. It also allows us to set the constraints for the app. This file contains:

Project general settings such as name, description, and version of the project.

Project dependencies.

Project assets (e.g., images, audio, etc.).

5)Why does the first Flutter app build take so long?

When you build the Flutter app the first time, it will take a longer time. It is because the Flutter built the device-specific APK or IPA file. Thus, the Gradle and Xcode are used to build the file, taking a long time.   Apk- android application p[ackage    Ipa -ios appstore pakage

6) What is the Flutter architecture?Flutter has a three-layered architecture:-

* Upper Layer: The upper layer consists of the Dart programming language along with the widgets, animations, illustrations, customizations, etc.
* The middle layer or the Flutter Engine: This layer deals with text display, formatting, layout, etc.
* Bottom Layer or the built-in service: This layer is for managing plugins or packages.

7) What build modes are available in Flutter?

 Debug Mode- This mode is for debugging apps on a device or a simulator.

1. Profile Mode- Some debugging abilities are available, along with an analysis of app's performance in testing rounds.
2. Release Mode- This mode is used when deploying the app. This mode is for faster performance. You cannot debug or edit the code in this mode.

8) [Top Apps Made with Flutter Framework](https://www.intelivita.com/blog/apps-made-with-flutter/#top-apps-made-with-flutter-framework)

[Google Ads](https://www.intelivita.com/blog/apps-made-with-flutter/#google-ads)      Google Pay      Take your seat     Lunching      Pairing

9) What are widgets in Flutter?                                                                                                 Widgets in Flutter are nested with each other to get the final design of the app. Widgets are used to code mobile phone applications.This means the widget is the base on which the app is coded.

10)What are the limitations of Flutter?                                                                                                   Lack of third-party libraries                                                                                                                    Larger release size                                                                                                                                  Flutter works with Dart language which is not so advanced as C# or C-Sharp and JavaScript.Not so user-friendly for iOS developers as it is developed by Google and it takes time to resolve the issues for apple devices.

11)What is the difference between main() and runApp() functions in Flutter?main () function came from Java-like languages so it's where all program started, without it, you can't write any program on Flutter even without UI.runApp() function should return Widget that would be attached to the screen as a root of the Widget Tree that will be rendered.

12)Write difference between Hot reload and Hot restart

**Hot Reload**

* It performs very fast as compared to hot restart or default restart of flutter.
* If we are using the state in our app then hot reload will not change the state of the app.
* We perform hot reload by using key ctlr+\.

**Hot Restart**

* It is slower than hot reload but faster than the default restart.
* It doesn’t preserve the state of our it starts from the initial state of our app.
* We perform hot restart using ctrl+shift+\

13 types of compilation techniques?1. Jit - just in time   fast in compile time    slow in execution time It helps developer eg hot reload        2. Aot- ahead of time     slow in compile time    fast in execution time It helps user

14 What is *Fat Arrow Notation* in Dart and when do you use it?  =>

The fat arrow syntax is simply a shorthand for returning an expression and is similar to (){ return expression; }.

The fat arrow is for returning a single line, braces are for returning a code block.

Only an expression—not a statement—can appear between the arrow (=>) and the semicolon (;). For example, you can’t put an *if* statement there, but you can use a *conditional* expression

15)What are Null-aware operators?

* Dart offers some operators for dealing with values that might be null.
* One is the ??= assignment operator, which assigns a value to a variable only if that variable is currently null:

    int a; // The initial value of a is null.

a ??= 3;

    print(a);

* Another null-aware operator is ??, which returns the expression on its left unless that expression’s value is null, in which case it evaluates and returns the expression on its right:

print(1 ?? 3); // <-- Prints 1.

print(null ?? 12); // <-- Prints 12.

Null safety's main goal is to have control into WHERE, HOW & WHEN  null can flow through a program

**After null safety:**

1.types in your code are non-nullable by default, meaning that variables can’t contain null unless you say they can. With null safety,

Eg:   int a;  int a = 4;  int? a;

class Car {

String carName = "Aston Martin";

}

void main() {

  Car cars;

  print(cars.carName);

}   o/p:Error: Non-nullable variable 'cars' must be assigned before it can be used.

class Car {

String carName = "Aston Martin";

}

void main() {

  Car cars = Car();

  print(cars.carName);

}o/p:Aston Martin

2.   **runtime** null-dereference errors turn into **edit-time** analysis errors.  
 It helps to fix the error in the fastest way.

3.  Sound null safety is available in Dart 2.12 and Flutter 2.

4.  Implicit downcasting is removed.

Eg: function1(String s) {}

          function2(Object d) {

        function1(d); // it will give error bcz type is not initialized Implicitly downcasts are removed. print(d.toUppercase()); }

function1(String s) {}

  function2(Object d) {

function1(d as String);   // d` is promoted to `String`

print(d.toUppercase()); }

5 The null assertion operator (!)

If you’re sure that an expression with a nullable type isn’t null, you can use a [null assertion operator](https://dart.dev/null-safety/understanding-null-safety#null-assertion-operator) (!) to make Dart treat it as non-nullable. By adding ! just after the expression, you tell Dart that the value won’t be null, and that it’s safe to assign it to a non-nullable variable.<https://dart.dev/codelabs/null-safety>

**6Late modifier:** In Dart, we use the late keyword to declare variables that will be initialized later. These are called non-nullable variables as they are initialized after the declaration. Hence, we use the late keyword

 ref:<https://stackoverflow.com/questions/68965626/why-do-we-need-to-add-late-modifier-while-declaring-the-variables>

.

<https://dev.to/no2s14/understanding-var-dynamic-const-final-in-dart-1han>-important

Var

One of many ways, and the simplest way, to define a **variable** in Dart is using the var key word. var message = 'Hello, World'; This example creates a variable called message , and also initializes the variable with a String value of Hello, World . Now, you can access that value by referring to the message variable.

**Dynamic**

dynamic: can change TYPE of the variable, & can change VALUE of the variable later in code. var: can't change TYPE of the variable, but can change the VALUE of the variable later in code.

Note:

* **dynamic**: can change TYPE of the variable, & can change VALUE of the variable later in code.
* **var**: can’t change TYPE of the variable, but can change the VALUE of the variable later in code.
* **final**: can’t change TYPE of the variable, & can’t change VALUE of the variable later in code.

**Types of cases**

Camel case       helloWorld  We have to use Camel case in function name in flutter

Snake case       hello\_world

Pascal case HelloWorld We have to use Camel case in class name in flutter

    what is flutter

* Flutter is a free and open-source mobile UI framework created by Google and released in May 2017.It allows you to create a native mobile application with only one codebase. This means that you can use one programming language and one codebase to create two different apps (for iOS and Android).

two important parts of flutter

* An SDK (Software Development Kit): A collection of tools that are going to help you develop your applications. This includes tools to compile your code into native machine code (code for iOS and Android).
* A Framework (UI Library based on widgets): A collection of reusable UI elements (buttons, text inputs, sliders, and so on) that you can personalize for your own needs.

* advantages of using flutter.
  + Reduce Code Development:

Cross-platform Development:

Live and Hot Reloading:

Similar to Native App performance:

Good Community Support:

Little/Minimal Code:

Documentation:

Expressive and Flexible UI:

* What are the Flutter widgets?
  + A Flutter app is always considered as a tree of widgets. Whenever you are going to code for building anything in Flutter, it will be inside a widget. Widgets describe how your app view should look like with their current configuration and state. When you made any alteration in the code, the widget rebuilt its description by calculating the difference of previous and current widget to determine the minimal changes for rendering in the app's UI.
  + Widgets are nested with each other to build the app. It means your app's root is itself a widget, and all the way down is a widget also. For example, a widget can display something, can define design, can handle interaction, etc.
* What are the best editors for Flutter development?
  + Android Studio
  + Visual Studio
  + IntelliJ IDEA
  + Xcode

**What are the data types in the Dart language?**

* Strings: String values are represented by the keyword String.
* Booleans: A Boolean value is represented by the bool keyword.
* Lists: Predefined List classes are used to describe lists.
* Maps: Predefined Map classes are used to describe Maps.
* The Dynamic Type: The dynamic keyword is explicitly used as a type annotation.

**What is type-checking in Dart?**

Type checking in Dart is the process of testing and enforcing the constraints of types at either compile-time (i.e., statically) or runtime (i.e., dynamically).

**9. What is the use of typedef in Dart?**

In Dart, Typedef is used to generate a function's user-defined identity (alias). That identity can be used in place of the process in the program code. The function's parameters are defined using typedef. It can also provide the variable with a process.

**10. Does Dart have a syntax for declaring interfaces?**

Interface declaration does not have a syntax in Dart. Dart considers class definitions to be interfaces. To use an interface, classes should use the implements keyword. It is required that the implementing class offer a concrete implementation of all of the implemented interface's functions.

**What are Streams in Dart?**

* The longer-term and Stream classes define asynchronous programming in Dart.
* A stream can be thought of as a series of asynchronous occurrences. It's similar to an asynchronous Iterable in that instead of receiving a future event once you invite it, the stream informs you that an occasion has arrived.
* Streams are formed in various methods, but they have always been used the same way; the asynchronous loop is one.

**12. Which editor is used to enable breakpoint and step-by-step debugging?**

Breakpoints and step-by-step debugging are available in the WebStorm editor. At the place where the breakpoint is set, the program will crash. This capability is similar to what you'd find in a Java or C# program. From the WebStorm Editor, you may observe variables, traverse the stack, and step over and into method and function calls.

**Does Dart Support Comment?**

Dart can support remarks. There are different types of comments:

* Single-line comments( // )
* Multi-line comments ( /\*\*/ )

**16. What Are The Various Types Of Operators In Dart?**

There are many different types of operators in Dart:

* Arithmetic Operators
* Equality and Relational Operators
* Type test Operators
* Bitwise Operators
* Assignment Operators
* Logical Operators

**17. Is Dart easier than JavaScript?**

Dart is around two times as quick as JavaScript. Dart is a type-safe programming language that can be compiled using both AOT and JIT compilers. Dart is a flexible programming language that may be used in various tasks. Dart is a scripting language similar to Javascript and is simple to learn if you already know Javascript.

**18. Does Dart support overloading?**

Dart doesn't support function overloading in any way. Only within the class where they are declared can static methods be accessed without the class name as a prefix.

**What are various string functions in Dart?**

Various string functions are provided:

|  |  |
| --- | --- |
| String methods | Description |
| toLowerCase() | It is a function that turns all string characters to lowercase. |
| toUpperCase() | It turns all string characters in this to uppercase. |
| trim() | removes all whitespace from the string. |
| compareTo() | It is a function that compares one object to another. |

**. What are the platforms used in Dart?**

Dart's compiler technology allows you to run code in a variety of ways, including:

* Native platform: Dart features a Dart VM with a just-in-time (JIT) compilation and an ahead-of-time (AOT) compiler to produce machine code for programs targeting mobile and desktop platforms.
* Web platform: Dart offers both a development time compiler (dartdevc) and a production time compiler (dartprod) for web projects (dart2js). Dart is translated into JavaScript by both compilers.

**What is run in Dart?**

A Dart runtime is required to execute code, regardless of which platform you use or how you create it. This runtime handles the following essential tasks:

* Dart has a managed memory model, which means that unneeded memory is reclaimed by a garbage collector (GC).
* Enforcing the Dart type system: While most type checks in Dart are static (at build time), there are a few dynamic type checks (runtime). The Dart runtime, for example, uses type check and cast operators to enforce vigorous checks.
* Isolates are managed by the Dart runtime, which is in charge of the primary isolate (where code generally runs) and any secondary isolates that the app creates.

The Dart runtime is included automatically inside self-contained executables on native platforms and is part of the Dart VM provided by the dart run command.

**What are Getters and Setters?**

The class methods getter and setter interpret data in class fields. A setter is used to set the class field’s data to some variable, whereas a getter is used to read or get the data of the class field.

Getter method

    It's used to get a specific class field and save it in a variable. The default getter function is available in all classes, although it can be changed explicitly.

Setter method

It's used to put data into a variable that the getter method has returned. A default setter function exists for all classes, but it can be explicitly overridden. Using the set keyword,

**What Is Method Overriding In Dart?**

In Dart, method overriding happens when a child class attempts to override the method of the parent class. When a child class extends a parent class, it gains full access to the parent class's methods and thereby overrides the parent class's methods. It's accomplished by redefining a method from the parent class.

When you need to conduct different functions for each child class, this technique comes in handy because we can simply re-define the content by overriding it.

* Only the child class, not the parent class, can override a method.
* Except for the information present inside the method, which can and cannot be identical, both the child and parent class methods should be perfect copies, from name to parameter list.
* The child class cannot override a method that is designated final or static in the parent class.
* Because parent class constructors cannot be inherited, the child class cannot override them.

**What is a constructor in Dart?**

When an object is created in a program, constructors are used to initialize it. However, the default function Object() { [native code] } will not be built and will be disregarded if you do so. Dart constructors have the same name as the class and no return type.

**29. What is polymorphism in Dart?**

Dart supports polymorphism. Polymorphism allows a group of objects to share the same interface yet have various implementations. Let's add a new concrete class called Plane to the mix. You can use polymorphism to create Blimp and Plane and apply them to the Aircraft variable. You can use their shared attributes because they are both descendants of Aircraft. Each object instance has its own set of class properties and instance values and output that is exclusive to each object's class instance.

**30. What is the Dart final?**

In Dart, the last keyword defines immutable constants or objects. The main difference between the final and const keywords is that absolute is a runtime constant, which means its value can be assigned at runtime rather than at compile-time like the const keyword.

**31. What is a double dot in Dart?**

If you want to call many methods on the same object, you can avoid repeating the same target.

**32. What is encapsulation in Dart?**

Encapsulation is a technique for hiding property or method while still allowing access to it within the scope of the function. Unlike other object-oriented programming languages, Dart encapsulates data at the library level rather than the class level.

**33. What is an optional parameter in Dart?**

Optional parameters are those that are not required to be specified when invoking a function. After the essential parameters, optional parameters must be stated. Optional parameters can also have a default value that is utilized if the function call does not provide one.

**What are the types of Inheritance in Dart?**

* Single Inheritance: This inheritance occurs when a class inherits from a single parent class.
* Multiple Inheritance: This inheritance occurs when a class inherits from more than one parent class. This is not supported by Dart.
* Multi-Level Inheritance: This inheritance occurs when a class inherits another child class.
* Hierarchical Inheritance: More than one class has the same parent class under hierarchical inheritance.

**37. What are the Snapshots in Dart?**

The Dart relies heavily on snapshots. Snapshots are files that include objects and other runtime information.

Script snapshots

Snapshot files can be created from Dart programs. All of the program code and dependencies are prepared and ready to run in these folders. This enables quick start-ups.

Full snapshots

The Dart core libraries can be compiled into a snapshot file, allowing them to be loaded quickly. The core libraries are prebuilt snapshots that are loaded at runtime in most standard distributions of the main Dart

Object snapshots

Dart is an asynchronous programming language. For concurrency, it takes advantage of isolates. Because these are message-passing workers, a mechanism to serialize a message is required. This is accomplished by creating a snapshot from a particular object, which is then passed to another isolate for deserialization.

**What is rune in Dart?**

**Ans:** In Dart, rune is an integer representing Unicode code point.

**What is the use of this keyword in Dart?**

**Ans:** In Dart, this keyword refers to the current instance of the class.

void main() {

Car c1 = new Car('E1001');

}

class Car {

String engine;

Car(String engine) {

this.engine = engine;

print("The engine is : ");

}}

**What are the types of list in Dirt?**

**Ans:** There are two types of list in Dirt that are given below:

1. Fixed Length List : (length fixed)
2. Growable List: (Length can change at runtime.

**Explain an interface in dart with an example?**

*Unlike other languages dart does not have any interface keyword, You can implement it with an abstract keyword*

abstract class Joker{

 void makePeopleLaugh();

}

class Clown implements Joker{

 void makePeopleLaugh() {

   // Here is where the magic happens

 }

}

class Comedian implements Joker{

 void makePeopleLaugh() {

   // Here is where the magic happens

 }

}

**How do you catch an error in the Dart program?**

*Unlike other languages, dart try and catch syntax are different*

try {

// ...

} on SomeException catch(e) {

//Handle exception of type SomeException

} catch(e) {

//Handle all other exceptions

}

**What is a constructor? Name types of constructor in dart?**

*A constructor is a special function of the class that is responsible for initializing the variables of the class.*

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

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

**What are “async”, “await ”and “Future”?**

*In dart,* ***Async*** *means that this function is asynchronous, and you might need to wait a bit to get its result.*

***Await*** *literally means — wait here until this function is finished and you will get its return value.*

***Future*** *is a type that “comes from the future ” and returns a value from your asynchronous function. It can complete with success(.then) or with an error(.catchError)*

**What are the differences between Object and dynamic?**

*dynamic is not a type; it just disables type checking. The object is the ‘union’ of all non-nullable types, type checking rules still apply.*

*With dynamic*

// a 'dynamic' variable can be assigned value of any type

dynamic a = 2;

// assign 'dynamic' value to any variable and code checker will not complain

int b = a;

// even when there is a bug

String c = a;

*With Object*

// It is OK to assign a 'int' value to an 'Object' variable, because 'int' is a subtype of 'Object'

Object a = 2;

// will get type error: "A value of type 'Object' can't be assigned to a variable of type 'int'"

int b = a;

// typecast is required when assign a 'Object' value to a variale of one of its subtypes.

int c = a as int;

**What is assert in dart?**

*assert statements are useful for debugging a dart project. It is used mainly in development mode. assert takes one expression and checks if it is true or false. If it is true, the program runs normally and if it is false, it stops the execution and throws one error called AssertionError.*

**4. What is Cascade notation in dart?**

Cascades (.., ?..) allow you to make a sequence of operations on the same object. In addition to accessing instance members, you can also call instance methods on that same object.

var paint = Paint()

 ..color = Colors.black

 ..strokeCap = StrokeCap.round

 ..strokeWidth = 5.0;

**5. What is the spread operator in dart?**

*Dart supports the* [***spread operator***](https://dart.dev/guides/language/language-tour#spread-operator)*, which allows inserting multiple elements into a collection.*

*Instead of calling add() or addAll(), You can simply use … operator to insert multiple records*

return [

   ...someVariable.toList(),

   anotherObject,

   ...anotherListOfObjects

];

**6. What is “mixin”? When do we use it?**

*Mixins are very helpful when we want to share behavior across multiple classes that don’t share the same class hierarchy, or when it doesn’t make sense to implement such behavior in a superclass*

**7. What is isolate?**

*To achieve concurrency dart, make use of isolate. Isolate is similar to multithread in other languages.*

**Does dart uses VM, if so, how?**

*Yes, Dart VM is being used when you build your mobile app. Dart VM has two different operation modes a JIT one and an AOT one.*

*In the* ***JIT*** *mode, Dart VM is capable of dynamically loading Dart source, parsing it, and compiling it to native machine code on the fly to execute it. This mode is used when you develop your app and provides features such as debugging, hot reload, etc.*

*In the* ***AOT*** *mode, Dart VM does not support dynamic loading/parsing/compilation of Dart source code. It only supports loading and executing precompiled machine code. However even precompiled machine code still needs VM to execute, because VM provides a runtime system that contains garbage collector, various native methods needed for dart:\* libraries to function, runtime type information, dynamic method lookup, etc. This mode is used in your deployed app.*

How much will you rate yourself in Dart?

When you attend an interview, Interviewer may ask you to rate yourself in a specific Technology like Dart, So It's depend on your knowledge and work experience in Dart.

2. What challenges did you face while working on Dart?

This question may be specific to your technology and completely depends on your past work experience. So you need to just explain the challenges you faced related to Dart in your Project.

3. What was your role in the last Project related to Dart?

It's based on your role and responsibilities assigned to you and what functionality you implemented using Dart in your project. This question is generally asked in every interview.

4. How much experience do you have in Dart?

Here you can tell about your overall work experience on Dart.

5. Have you done any Dart Certification or Training?

It depends on the candidate whether you have done any Dart training or certification. Certifications or training are not essential but good to have.

What’s the difference between a function expression and a function declaration?

A function expression is a function that is assigned to a variable, while a function declaration is a function that is declared as a standalone entity. Function expressions can be anonymous, while function declarations cannot.

What happens if two classes contain constructors with the same arguments?

If two classes contain constructors with the same arguments, the Dart compiler will throw an error.

 How can you call multiple constructors for a class?

You can call multiple constructors for a class by using the keyword “this”. For example, if you have a class with two constructors, one that takes an int aclass MyClass {

MyClass(int i) { … }

MyClass(String s) { … }

}

void main() {

var myClass = new MyClass(42); // calls the first constructor

var myClass = new MyClass(“foo”); // calls the second constructor

}

nd one that takes a String,

 Is it possible to assign values from a list to individual variables? How?

Yes, it is possible to assign values from a list to individual variables. This can be done using the spread operator. The spread operator allows you to take the values from a list and spread them out into individual variables. For example, if you have a list of numbers called myList, you could use the spread operator to assign the values of myList to three individual variables like this:

var a, b, c;

[a, b, c] = myList;

16. What’s the difference between lists and maps?

Lists are ordered collections of values, while maps are unordered collections of key-value pairs.

In which situations should we use generics?

Generics are used when you want to create a class or function that can work with multiple data types. This allows you to write code that is more flexible and can be reused more easily.

18. What are mixins?

Mixins are a way to reuse a class’s code in multiple other classes. This is especially useful in Dart, since Dart does not support multiple inheritance. To use a mixin, you first create a class that contains the code you want to reuse, then you “mix in” that class to any other classes that should have access to that code.

* **1. How do you reduce widget rebuild?**
* The state of a widget changes when you rebuild it. This is beneficial because it allows the user to see the UI reflect state changes. However, recreating sections of the user interface that do not need to change is wasteful.  
  There are various things you can do to limit unnecessary widget rebuilding. The first step is to divide a huge widget tree into smaller individual widgets, each with its build process. Use the const constructor whenever possible to convince Flutter that the widget does not need to be rebuilt.  
  Keep a stateful widget's subtree as minimal as possible. If a stateful widget requires a widget subtree, construct a custom widget for the stateful widget and provide it with a child parameter.

Different types of streams in Dart

There are two types of streams in Dart, single subscription streams and broadcast streams.

Single Subscription Streams

* The events within a larger whole are delivered sequentially with single subscription streams.
* They are used for events that are in received matters or for reading a file.
* There is only one listener throughout the sequence to trigger the event, else the event won’t be triggered.

Broadcast Streams

* Initially, the event needs to be subscribed to by the listener, then only these streams deliver events to their subscribers and subscribers can immediately start listening to events.
* There are several listeners to events simultaneously. Besides, one can also listen to the event again even after canceling a previous subscription.

**How is whenCompleted() different from then() in Future?**

**Ans:** whenComplete will fire a function either when the longer term completes with a mistake or not, while .then returns a replacement Future which is completed with the results of the decision to onValue (if this future completes with a value) or to onError (if this future completes with an error)

.whenCompleted is that the asynchronous equivalent of a "finally" block.