1. **Introduction of Dart Programming language.**

Dart is a modern, high-performance, object-oriented programming language developed by Google. It is designed for building fast, scalable, and highly performant applications across multiple platforms, including mobile, web, desktop, and backend systems. Dart is the language behind the Flutter framework, which powers cross-platform applications with a single codebase.

1. **Key Features**

* Object Oriented Language
* Strong and Sound Type
* Cross-Platfrom
* Null Safety
* Fast Performance
* AOT( Ahead of time) & JIT( Just in Time ) Compilation
* Asynchronous Programming

1. **Where Dart is Used?**

1. Mobile App Development (with Flutter)

Dart powers the Flutter framework, enabling developers to build cross-platform mobile applications.

Popular apps like Google Pay, Reflectly, and eBay Motors are built with Dart and Flutter.

2. Web Development

Dart can be compiled to JavaScript, making it suitable for building high-performance web apps.

3. Desktop Applications

Using Flutter, Dart can build native desktop applications for Windows, macOS, and Linux.

4. Backend Development

Dart can be used as a backend language, e.g., with the Shelf package for building web servers or APIs.

1. **Advantages of Dart**

* Unified Development: Single language for UI, backend, and web development.
* Hot Reload: Fast iteration during development, especially with Flutter.
* Null Safety: Minimizes runtime errors caused by null references.
* Great Tooling: Integrated tools for debugging, testing, and profiling.
* Easy Learning Curve: Familiar syntax similar to JavaScript, Java, or C#.

1. **Understanding Programming Fundamentals**
2. **Data Types In Dart**

In Dart, data types are used to define the type of data a variable can hold. Dart is a statically typed language, which means every variable has a type that is checked at compile-time. However, with features like var and dynamic, Dart provides flexibility in declaring variables.

**Types of Data Types in Dart.**

1. **Numbers**

Used for numeric values:

- int: For integers ( whole numbers).

- double: For decimal or floating-point numbers.

**Example:**

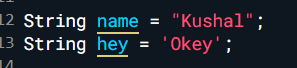
****

1. **Strings**

Used for text or sequences of characters. Strings are enclosed in single (‘) or

double quote(“).

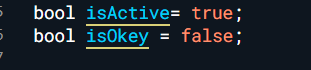
**Example:**



1. **Boolean**

Represents **true** or **false** values, used for logical operations.

**Example:**

****

**Note : There are many more data types in which we will study under the collections in dart.**

1. **Variables**

In Dart, variables are used to store and manage data in memory. A variable is essentially a named container that holds a value, which can be updated or retrieved during the program's execution.

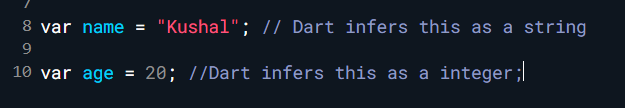
**Declaring Variables in Dart.**

1. **Using var**

 Automatically infers the type of the variable based on the assigned value.

 The variable’s type cannot change after type is inferred.

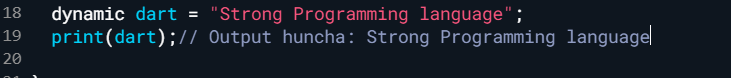
**Example:**



1. **Using dynamic**

The dynamic keyword allows a variable to hold values of any type and its type can change at runtime. Use dynamic only when necessary, as it skips compile-time type checking.

**Example:**

****

1. **Using final and const**

**Final :** A variable whose value can only be set once but is initialized at runtime.

**Const :** A variable that is a compile-time constant.

**Example:**



**C. Operators in Dart.**

                Operators in Dart are symbols or keywords used to perform operations on values and variables. They allow you to manipulate data, perform arithmetic, compare values, and control the flow of a program.

**Types of operators in Dart.**

* + 1. **Arithmetic Operators**
* Addition    “  + ”
* Subtraction    “ - “
* Multiplication “ -\*“
* Division“ /“
* Integer Division“ ~/ “
* Modulus ( Remainder ) “ %“

**Example :**

**ii. Relational (Comparison) Operators**

* Equal to   “  == ”
* Not Equal to    “ != “
* Greater Than “ >“
* Less than“ < “
* Greater than or equal to “ >=“
* Less than or equal to “ <=“

**Example :**

**iii. Logical Operators**

- Logical AND “  &&  ”

* Logical OR “  ||  ”
* Logical NOT “  !  ”

**Example :**

**iv. Assignment Operators**

* Assign “  =  ”
* Add and Assign “  +=  ”
* Subtract and assign “ -= ”
* Multiply and assign “  \*=  ”
* Divide and Assign “  /=   ”
* Modulus and Assign “  %=  ”

**Example :**

**v. Bitwise Operators**

* Bitwise AND “  &  ”
* Bitwise OR“  `  ”
* Bitwise XOR “ ^ ”
* Bitwise NOT“  ~  ”
* Left Shift “  <<   ”
* Right Shift “ >>  ”

**Example :**

**vi. Conditial (Ternary) Operators**

* condition ? Exp1 : Exp2

                     If condition is true, evaluate expr1; otherwise, evaluate expr2.

**Example:**

**vii.  Null aware Operators**

 -“  ??  ” (If the left- hand operand is null, return the right -hand operand)

* “  ??=  ” ( Assign value only if the variable is null)
* “ ?.  ” ( Calls a method or accesses a property if the object is not null.)

**Example :**

**viii.  Type Test Operators**

* “  is  “ ( Checks if an object is of a specific type ).
* “  Is! “ ( Checks if an object is not of a specific type ).
* “  as. “ (Typecast operator)

**Example :**

**xi.  Cascade Operator ( . . )**

- Allowing Chaining operators on the same object.

**Example :**