Chapter 1: Introduction to Dart

What is Dart?

- **Dart** is an **open-source, object-oriented programming language** developed by **Google**.
- It is mainly used for **building cross-platform mobile apps**, **web apps**, and **desktop applications**.
- Dart is the **primary language for Flutter**, which is used to create Android and iOS apps with a single codebase.

Key Features of Dart

1. **Object-Oriented** → Supports classes, objects, inheritance, etc.
2. **Cross-Platform** → One code works on Android, iOS, web, and desktop.
3. **Compiled Language** → Dart code is compiled into fast native code.
4. **Null Safety** → Helps prevent null value errors in programs.
5. **Rich Standard Library** → Provides built-in support for lists, maps, strings, etc.

First Dart Program

Example:

```
void main() {
  print('Hello, Dart!');
}
```

Explanation:

- `void main()` → Every Dart program starts with the **main()** function.
- `print()` → Used to display output on the console.
- `'Hello, Dart!'` → Text (string) displayed on screen.

Output:

```
Hello, Dart!
```

What is a Variable?

A **variable** is a **named storage location** that holds a value.
It is used to store data like numbers, text, or boolean values.

Example:

```
void main() {
  var name = 'Neeraj';
  var age = 23;
  print('Name: $name');
  print('Age: $age');
```

```
}
```

Output:

```
Name: Neeraj
Age: 23
```

Explanation:

- `var` keyword is used to declare a variable.
- Dart automatically detects the **data type** (String, int, etc.) from the value.
- `$name` is used for **string interpolation** (to print variable value inside a string).

Rules for Naming a Variable

1. The name must **start with a letter or underscore (_)**.
   Example: `name`, `_count`
   Invalid: `1value`, `@name`
2. It can contain **letters, digits, and underscores**.
   Example: `user_name`, `age1`
3. Variable names are **case-sensitive**.
   `name` and `Name` are different.
4. Avoid using **Dart keywords** as variable names.
   Example: `var class = 'test';` (invalid)
5. Choose **meaningful names** for readability.
   Example: `userAge` is better than `ua`.

Data Types in Dart

What are Data Types?

A **data type** defines the kind of value a variable can store.
It helps Dart know what operations are allowed on that variable.

Common Data Types in Dart

Data Type	Description	Example
`int`	Stores integer (whole) numbers	`int age = 23;`
`double`	Stores decimal numbers	`double price = 99.99;`
`String`	Stores text (characters/words)	`String name = 'Neeraj';`
`bool`	Stores boolean values (true/false)	`bool isActive = true;`
`var`	Automatically detects the data type	`var city = 'Lucknow';`
`dynamic`	Can hold any type of value, even change type later	`dynamic value = 10; value = 'Hello';`

Data Type	Description	Example
`List`	Collection of multiple values (like arrays)	`List<int> numbers = [1, 2, 3];`
`Map`	Stores key-value pairs	`Map user = {'name': 'Neeraj', 'age': 23};`

Examples of Each Data Type

```
void main() {
  // Integer
  int age = 23;

  // Double
  double height = 5.9;

  // String
  String name = 'Neeraj';

  // Boolean
  bool isStudent = true;

  // Var - Dart decides type automatically
  var country = 'India';

  // Dynamic - Type can change
  dynamic value = 50;
  value = 'Fifty'; // valid because dynamic allows changing type

  // List (similar to array)
  List<String> fruits = ['Apple', 'Banana', 'Mango'];

  // Map (key-value pairs)
  Map<String, dynamic> student = {
    'name': 'Neeraj',
    'age': 23,
    'isStudent': true
  };

  // Printing values
  print('Age: $age');
  print('Height: $height');
  print('Name: $name');
  print('Is Student: $isStudent');
  print('Country: $country');
  print('Value: $value');
  print('Fruits: $fruits');
  print('Student Info: $student');
}
```

Output

```
Age: 23
Height: 5.9
Name: Neeraj
Is Student: true
Country: India
```

```
Value: Fifty
Fruits: [Apple, Banana, Mango]
Student Info: {name: Neeraj, age: 23, isStudent: true}
```

Tip: Difference Between `var` and `dynamic`

Feature	`var`	`dynamic`
Type fixed after first assignment	Yes	No
Can change data type later	No	Yes
Compile-time type checking	Yes	No

Example:

```
var x = 10;
x = 'Hello'; // Error (type fixed as int)

dynamic y = 10;
y = 'Hello'; // Valid
```

Operators in Dart

What are Operators?

Operators are **symbols** that perform operations on variables and values.
Example: +, -, *, /, ==, etc.

Types of Operators in Dart

Operator Type	Description
Arithmetic Operators	Perform mathematical operations
Relational Operators	Compare two values
Logical Operators	Combine multiple conditions
Assignment Operators	Assign or modify values
Unary Operators	Work on a single operand (like ++, --)
Conditional Operator	Short form of if-else
Type Test Operators	Check data type (is / is!)

1. Arithmetic Operators

Operator	Description	Example
+	Addition	`a + b`
-	Subtraction	`a - b`
*	Multiplication	`a * b`
/	Division (returns double)	`a / b`

Operator	Description	Example
~/	Integer Division (returns int)	a ~/ b
%	Modulus (remainder)	a % b

Example:

```
void main() {
  int a = 10, b = 3;

  print('a + b = ${a + b}');
  print('a - b = ${a - b}');
  print('a * b = ${a * b}');
  print('a / b = ${a / b}');
  print('a ~/ b = ${a ~/ b}');
  print('a % b = ${a % b}');
}
```

Output:

```
a + b = 13
a - b = 7
a * b = 30
a / b = 3.3333333333333335
a ~/ b = 3
a % b = 1
```

2. Relational (Comparison) Operators

Operator	Description	Example
==	Equal to	a == b
!=	Not equal to	a != b
>	Greater than	a > b
<	Less than	a < b
>=	Greater than or equal to	a >= b
<=	Less than or equal to	a <= b

Example:

```
void main() {
  int a = 10, b = 20;

  print(a == b);  // false
  print(a != b);  // true
  print(a > b);   // false
  print(a < b);   // true
  print(a >= b);  // false
  print(a <= b);  // true
}
```

3. Logical Operators

Operator	Description	Example
`&&`	Logical AND	`(a > 5 && b < 10)`
`		`
`!`	Logical NOT	`!(a > b)`

Example:

```
void main() {
  bool x = true;
  bool y = false;

  print(x && y); // false
  print(x || y); // true
  print(!x);     // false
}
```

4. Assignment Operators

Operator	Example	Meaning
`=`	`a = 5`	Assign value 5 to a
`+=`	`a += 2`	a = a + 2
`-=`	`a -= 2`	a = a - 2
`*=`	`a *= 2`	a = a * 2
`/=`	`a /= 2`	a = a / 2
`~/=`	`a ~/= 2`	a = a ~/ 2

Example:

```
void main() {
  int a = 10;

  a += 5;
  print(a); // 15

  a -= 3;
  print(a); // 12

  a *= 2;
  print(a); // 24

  a ~/= 4;
  print(a); // 6
}
```

5. Unary Operators

Operator	Description	Example
`++a`	Pre-increment	Increase first, then use
`a++`	Post-increment	Use first, then increase

Operator	Description	Example
`--a`	Pre-decrement	Decrease first, then use
`a--`	Post-decrement	Use first, then decrease

Example:

```
void main() {
  int a = 5;
  print(++a); // 6 (increment then print)
  print(a++); // 6 (print then increment)
  print(a);   // 7
}
```

6. Conditional (Ternary) Operator

Syntax:

```
condition ? expr1 : expr2
```

If condition is **true**, executes `expr1`; otherwise `expr2`.

Example:

```
void main() {
  int age = 18;
  String result = (age >= 18) ? 'Eligible' : 'Not Eligible';
  print(result);
}
```

Output:

```
Eligible
```

7. Type Test Operators

Operator	Description	Example
`is`	True if object is of specified type	`x is int`
`is!`	True if object is **not** of specified type	`x is! String`

Example:

```
void main() {
  var name = 'Neeraj';
  var number = 23;

  print(name is String); // true
  print(number is! String); // true
}
```

Summary:

Arithmetic → `+ - * / %`

Relational → `== != > < >= <=`

Logical → `&& || !`

Assignment → `= += -= *= /=`

Conditional → `?:`

Type Test → `is`, `is!`

Chapter 4: Strings in Dart

What is a String?

A **String** is a sequence of characters used to represent text.
In Dart, strings are enclosed in **single quotes (' ')** or **double quotes (" ")**.

Example:

```
void main() {
  String name = 'Neeraj';
  String greeting = "Hello, Dart!";

  print(name);
  print(greeting);
}
```

Output:

```
Neeraj
Hello, Dart!
```

String Interpolation

You can insert variable values inside a string using **$variableName** or **${expression}**.

```
void main() {
  String name = 'Neeraj';
  int age = 23;

  print('My name is $name and I am $age years old.');
  print('Next year I will be ${age + 1}.');
}
```

Output:

```
My name is Neeraj and I am 23 years old.
Next year I will be 24.
```

Most Used String Methods in Dart

Method	Description	Example	Output
`length`	Returns number of characters	`'Hello'.length`	5
`toUpperCase()`	Converts to uppercase	`'dart'.toUpperCase()`	DART
`toLowerCase()`	Converts to lowercase	`'DART'.toLowerCase()`	dart
`contains()`	Checks if substring exists	`'hello'.contains('he')`	true
`startsWith()`	Checks if string starts with text	`'flutter'.startsWith('f')`	true
`endsWith()`	Checks if string ends with text	`'flutter'.endsWith('r')`	true
`substring()`	Extracts part of a string	`'flutter'.substring(0, 3)`	flu
`replaceAll()`	Replaces all occurrences	`'good day'.replaceAll('good', 'nice')`	nice day
`trim()`	Removes leading & trailing spaces	`' hello '.trim()`	hello
`split()`	Splits string into list	`'a,b,c'.split(',')`	[a, b, c]

Example:

```
void main() {
  String text = '  Hello Dart Language  ';

  print(text.trim());
  print(text.toUpperCase());
  print(text.contains('Dart'));
  print(text.replaceAll('Dart', 'Flutter'));
  print(text.split(' '));
}
```

Output:

```
Hello Dart Language
  HELLO DART LANGUAGE
true
  Hello Flutter Language
[, , Hello, Dart, Language, , ]
```

Lists in Dart

What is a List?

A **List** is an **ordered collection** of elements — similar to arrays in other languages.

Example:

```
void main() {
  List<String> fruits = ['Apple', 'Banana', 'Mango'];
```

```
  print(fruits);
}
```

Output:

```
[Apple, Banana, Mango]
```

Most Used List Methods

Method	Description	Example	Output
`add()`	Adds an element	`fruits.add('Orange')`	[Apple, Banana, Mango, Orange]
`addAll()`	Adds multiple elements	`fruits.addAll(['Grapes', 'Guava'])`	[Apple, Banana, Mango, Grapes, Guava]
`remove()`	Removes a specific element	`fruits.remove('Banana')`	[Apple, Mango]
`removeAt(index)`	Removes by position	`fruits.removeAt(0)`	[Banana, Mango]
`length`	Returns total elements	`fruits.length`	3
`contains()`	Checks if item exists	`fruits.contains('Mango')`	true
`indexOf()`	Finds position of an element	`fruits.indexOf('Mango')`	2
`sort()`	Sorts the list	`fruits.sort()`	[Apple, Banana, Mango]
`reversed`	Reverses the list	`fruits.reversed.toList()`	[Mango, Banana, Apple]
`clear()`	Removes all elements	`fruits.clear()`	[]

Example:

```
void main() {
  List<String> fruits = ['Apple', 'Banana', 'Mango'];

  fruits.add('Orange');
  fruits.remove('Banana');
  fruits.sort();

  print(fruits);
  print('First Fruit: ${fruits.first}');
  print('Total Fruits: ${fruits.length}');
}
```

Output:

```
[Apple, Mango, Orange]
```

```
First Fruit: Apple
Total Fruits: 3
```

Maps in Dart

What is a Map?

A **Map** stores data in **key-value pairs** (like a dictionary in Python).

Example:

```
void main() {
  Map<String, dynamic> student = {
    'name': 'Neeraj',
    'age': 23,
    'isActive': true
  };

  print(student);
}
```

Output:

```
{name: Neeraj, age: 23, isActive: true}
```

Most Used Map Methods

Method	Description	Example	Output
`keys`	Returns all keys	`student.keys`	(name, age, isActive)
`values`	Returns all values	`student.values`	(Neeraj, 23, true)
`length`	Total number of pairs	`student.length`	3
`addAll()`	Add multiple entries	`student.addAll({'city': 'Lucknow'})`	adds new key-value
`remove()`	Removes a key	`student.remove('age')`	removes age
`containsKey()`	Checks if key exists	`student.containsKey('name')`	true
`containsValue()`	Checks if value exists	`student.containsValue('Neeraj')`	true
`clear()`	Removes all entries	`student.clear()`	{}
`forEach()`	Iterates over map	`student.forEach((k,v)=>print('$k:$v'))`	prints all pairs

Example:

```
void main() {
  Map<String, dynamic> student = {
    'name': 'Neeraj',
    'age': 23,
    'course': 'MCA'
  };

  student['city'] = 'Lucknow';  // add new key
  student.remove('age');        // remove key

  print(student);
  print('Keys: ${student.keys}');
  print('Values: ${student.values}');

  student.forEach((key, value) {
    print('$key : $value');
  });
}
```

Output:

```
{name: Neeraj, course: MCA, city: Lucknow}
Keys: (name, course, city)
Values: (Neeraj, MCA, Lucknow)
name : Neeraj
course : MCA
city : Lucknow
```

Quick Summary:

Concept	Meaning
String	Text values (e.g., name = 'Neeraj')
List	Ordered collection (e.g., [1, 2, 3])
Map	Key-value pairs (e.g., {'name':'Neeraj'})