Dart Programming Language

Dart is an open-source, object-oriented, general-purpose programming language developed by Google in 2011. It is optimized for building fast, cross-platform apps (mobile, web, desktop, backend). The most popular framework that uses Dart is Flutter Framework.

Dart Syntax Basics :

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

Types of Variables in Dart :

- Float data type is not used in dart

1) int

Represents whole numbers without a decimal point.

```
void main()
{
  int age = 21;
  print(age);    // 21
  print(age.runtimeType);    // int
}
```

2) double

Represents floating-point numbers (numbers with a decimal point).

```
void main()
{
  double pi = 3.14;
  print(pi);   // 3.14
  print(pi.runtimeType);    // double
}
```

3) num

A parent type for both int and double. It allows variables to hold either integers or decimal values without changing type.

```
void main()
{
  num x = 10;
  num y = 3.5;
  num sum = x + y;
  print(sum);  // 13.5
  print(sum.runtimeType); // double
}
```

4) string

Used to store sequences of characters (text). Strings can use interpolation ($variable) to include values inside text.

```
void main()
{
  String name = "Chandini";
  print(name);   // Chandini
  print(name.runtimeType); // string
}
```

5) bool

Represents a logical value, either true or false.

```
void main()
{
  bool isLoggedIn = true;
  print(isLoggedIn);  // true
  print(isLoggedIn.runtimeType); // bool
}
```

6) var

Automatically infers the type at compile time based on the initial value. Once assigned, the type cannot be changed.

```
void main()
{
  var message = "Hello";
```

```
  print(message);                                                    // Hello
  message = 10;       // Error, type fixed as String, cannot be changed to int
  print(message);   // Hello
  print(message.runtimeType);                                        // string
}
```

7) dynamic

Allows flexibility by deciding the type at runtime. The variable's type can change later, but it loses compile-time safety.

```
void main()
{
  dynamic data = "Hi";
  print(data);                                                       // Hi
  data = 100;                                                        // This is valid
  print(data);  // 100
  print(data.runtimeType);                                           // int
}
```

Boolean Example with Conditions

```
void main()

{

  int age = 18;

  int requiredAge = 21;

  bool canVote = age >= requiredAge;

  bool isTeenager = age >= 13 && age <= 19;

  print(canVote);   // false

  print(isTeenager); // true

}
```

This example shows how a bool in Dart stores logical values (true/false) based on conditions.

Floating-Point Precision Issue

```
void main()
```

```
{
  double a = 0.1;
  double b = 0.2;
  print(a + b); // 0.30000000000000004
}
```

- **Explanation**:
  Computers store numbers in binary (base-2), but some decimal values like 0.1 and 0.2 don't have an exact binary representation. Instead, they are stored as very close approximations. When these approximations are added, the result is slightly off from the expected 0.3.
- **Why this happens**:
  - 0.1 becomes something like 0.1000000000000000555…
  - 0.2 becomes something like 0.2000000000000000111… Adding them gives 0.3000000000000000444… instead of 0.3.

Solution: Format the Output

```
void main() {
  double a = 0.1;
  double b = 0.2;
  print((a + b).toStringAsFixed(1)); // 0.3
}
```

- **Explanation**:
  .toStringAsFixed(n) rounds the number to n decimal places and returns it as a string. This way, the output looks correct (0.3) instead of showing the tiny precision error.