Dart Programming Language

Dart is a modern, general-purpose programming language developed by Google. It is optimized for building fast, multi-platform applications — especially mobile apps, but also web, desktop, and backend.

1) **Variables:** Name assigned to the memory location.

Types of variables in dart:

1) num: Supertype for both integer and floating point numbers.

   Two types:
   int: whole numbers without floating point.

   Ex: void main()
   ```
   {
     int x=10;
     print(x);  //10
     print(x.runtimeType);  //int
   }
   ```

   double: number with floating point.

   Ex: void main()

   ```
   {
     double x=10.33;
     print(x);  //10.33
     print(x.runtimeType);  //double
   }
   ```

2) bool: In Dart, bool is the Boolean data type, which can store only two values:

- true
- false

Ex: void main()

```
{
    int a = 10;
    int b = 5;
   bool isGreater = a > b;
```

```
    bool isEqual = a == b;
    print(isGreater); // true
    print(isEqual);   // false
}
```

3) dynamic: dynamic declares a variable whose type is not checked at compile time. The variable can hold any type, and its type can change during runtime.

```
Ex: void main()
{
    dynamic a = "Hello";
    print(a);       // Output: Hello
    a = 100;
    print(a);       // Output: 100
    a = true;
    print(a);       // Output: true
}
```

4) var: var declares a variable with type inference. The Dart compiler infers the variable's type from the initial value, and after that, the type is fixed and cannot change.

```
Ex: void main()
{
    var name = "Alice";  // Dart infers String type
    print(name);             // Output: Alice
    // name = 123;          //  Error: Can't assign int to a String variable
}
```

5) String: sequence of characters.

```
Ex: void main()
{
    String name= "john";  // john
}
```

NOTE:

1) float datatype is not used in dart.

2) void main()

```
{
    num a = 0.1;   or   double a=0.1;
    num b = 0.2;   or   double b=0.2;
    num sum = a + b;  // Result: 0.30000000000000004 due to floating-point precision
}
```

Floating-point numbers like 0.1 and 0.2 cannot be represented exactly in binary, causing tiny precision errors.

Solution: Use .toStringAsFixed() beacuse it converts a number to a string representation with a fixed number of decimal places.

- It rounds the number to the specified decimal places.
- Returns a string, not a number.
  Ex: void main()
  {
  num a=0.1;
  num b=0.2;
  num s=a+b;
  print(s.toStringAsFixed(2)); //0.30
  }