Dart Data Types and Variables - Complete Explanation | Flutter Tutorial

In this tutorial, we'll dive deep into Dart's data types and variables, essential concepts in Dart programming, especially when developing with Flutter. We'll break down what variables and data types are, their significance, and how to use them effectively.

Table of Contents

4. Common Data Types in Dart
    - Integers
    - BigInt
    - Double
    - String
    - Boolean
    - Collections
5. Variable Scope and Null Safety
6. Inline Variable Declaration
7. Working with BigInt and Large Numbers
8. Handling Double and Num Data Types
9. Boolean Data Types in Conditional Logic
10. Dynamic and Var Data Types (Upcoming Tutorial)

1. Introduction to Variables

A **variable** in Dart is a named space in memory that stores a value. This value can be retrieved, modified, and used multiple times throughout a program. Variables are essential for managing and accessing data efficiently during runtime.

Example:

```
int age = 25; // 'age' is a variable storing an integer value of 25.
```

2. Understanding Data Types in Dart

Data types define the kind of data a variable can hold. Dart is a strongly typed language, meaning each variable must have a specific type. These types can be integers, strings, booleans, etc.

Common Data Types:

- **int:** For integer values.
- **double:** For floating-point numbers.
- **String:** For text.
- **bool:** For true/false values.

- **BigInt:** For large integer values.
- **List, Map:** For collections of data.

3. Declaring and Initializing Variables

Declaration

Declaring a variable involves specifying the type and the variable name.

```
int score; // Declaring an integer variable named 'score'.
```

Initialization

Initialization means assigning an initial value to the variable.

```
score = 100; // Initializing 'score' with the value 100.
```

Combined Declaration and Initialization

You can declare and initialize a variable in one line.

```
int score = 100; // Declares and initializes 'score' in one step.
```

4. Common Data Types in Dart

Integers

`int` is used for storing whole numbers.

```
int age = 30;
```

BigInt

When an integer exceeds the standard range, use `BigInt`.

```
BigInt largeNumber = BigInt.parse('12345678901234567890123456789O');
```

Double

`double` is for floating-point numbers or numbers with decimals.

```
double price = 19.99;
```

String

`String` is used for text.

```
String name = "John Doe";
```

Boolean

`bool` is used for true/false values.

```
bool isLoggedIn = false;
```

Collections

Dart offers `List` and `Map` for handling collections of data.

```
List<int> numbers = [1, 2, 3];
```

```
Map<String, String> user = {'name': 'Alice', 'email': 'alice@example.com'};
```

5. Variable Scope and Null Safety

Variables have a scope, which is the context in which they are accessible. Dart also supports null safety, meaning variables must be assigned a value before they are used.

Example:

```
void main() {
  int x; // Declared but not initialized.
  print(x); // Error: Non-nullable variable 'x' must be assigned before it can be used.
}
```

To make a variable nullable, use the `?` symbol.

```
int? x; // 'x' can be null.
```

6. Inline Variable Declaration

You can declare and initialize a variable in one step, known as inline declaration.

Example:

```
String greeting = "Hello, World!";
```

7. Working with BigInt and Large Numbers

When dealing with numbers larger than the typical integer range, use `BigInt`.

Example:

```
BigInt bigValue = BigInt.parse('98765432101234567890');
print(bigValue); // Outputs: 98765432101234567890
```

8. Handling Double and Num Data Types

`double` is used for numbers with decimals, and `num` can hold both `int` and `double` values.

Example:

```
double pi = 3.14;
num anyNumber = 5; // Can be an int or double.
```

9. Boolean Data Types in Conditional Logic

Boolean variables are crucial in control flow, such as if-else statements.

Example:

```
bool isActive = true;

if (isActive) {
  print("The account is active.");
} else {
  print("The account is inactive.");
}
```

10. Dynamic and Var Data Types (Upcoming Tutorial)

Dart also includes `var` and `dynamic` for more flexible variable declarations. We'll explore these in the next session.

Conclusion

Understanding variables and data types is crucial in Dart programming. They allow you to manage data effectively and write more efficient code. In the next tutorial, we'll dive deeper into `var` and `dynamic` types, helping you understand when and how to use them. Happy coding!

This article covers the fundamental concepts of variables and data types in Dart, equipping you with the knowledge to start coding effectively in Flutter. Feel free to experiment with the examples provided to reinforce your understanding.