Introduction to Dart Programming Language

An Overview with Examples

Dr. Hammoudeh Alamri

What is Dart?

- Dart is an open-source, general-purpose programming language developed by Google. It is optimized for building web, server, and mobile applications.

Why Use Dart?

1. Easy to learn
2. Optimized for mobile apps (Flutter)
3. Strong typing with flexibility
4. Efficient performance on multiple platforms

Dart Syntax Overview

- Dart syntax is similar to other C-style languages such as Java and JavaScript.

- Example:

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

Variables and Data Types

- Dart supports various data types like:

- int
- double
- String
- bool
- var (dynamic typing)

- Example:

```
int age = 30;
String name = 'Alice';
bool isStudent = false;
```

Data Type: int

- The 'int' data type is used for whole numbers.

- Example:

```
int age = 33;
int year = 2025;
```

- Values can be positive or negative.

Data Type: double

- The 'double' data type is used for floating-point numbers.

- Example:

```
double height = 5.9;
double price = 49.99;
```

Data Type: String

- The 'String' data type is used for text or sequences of characters.

- Example:

```
String name = 'John';
String greeting = 'Hello, Dart!';
```

Data Type: bool

- The 'bool' data type is used for true/false values.

- Example:

```
bool isStudent = true;
bool isLoggedIn = false;
```

Data Type: var and dynamic

- In Dart, both var and dynamic are used to declare variables, but they behave differently in terms of **type inference, type safety, and flexibility**.

1. Type Inference:

•**var**: When you use var, Dart infers the type of the variable at compile time based on the assigned value. Once the type is inferred, it cannot be changed.

•Example:

```
var name = 'Alice'; // Dart infers 'String'
name = 'Bob'; // This is fine, because 'Bob' is also a String.
name = 42; // Error: A value of type 'int' can't be assigned to a variable of type 'String'.
```

•**dynamic**: A variable declared with dynamic is dynamic-typed, meaning that it can hold values of any type, and its type can change during runtime without causing compile-time errors.

•Example:

```
dynamic data = 'Alice'; // Initially, a String
data = 42; // Now, it's an int
data = true; // Now, it's a bool
```

2. Type Safety:

- **var:** Provides type safety after the initial type inference. If the variable is inferred as a specific type (e.g., String), it will enforce that type for future assignments, preventing accidental assignment of incompatible types.
- **dynamic:** Does not provide type safety. You can assign any type of value to a dynamic variable, and no compile-time checks will be performed. This can lead to runtime errors if you're not careful.
- Example:

```
dynamic value = 'hello';
print(value.length); // This works fine because it's a String.

value = 100;
print(value.length); // Runtime error! 'int' doesn't have a 'length' property.
```

3. Usage Scenario:

- var: Use var when you want Dart to infer the type and keep the variable type consistent throughout its use. It provides the benefits of type safety while avoiding the need to explicitly declare types.
  - Ideal for most cases where the type is predictable and should remain the same.
- dynamic: Use dynamic when you explicitly need a variable to hold different types of values throughout its lifecycle or when working with APIs, legacy code, or situations where the type isn't known ahead of time.
  - Example use case: Handling JSON responses, where the structure or types might not be consistent.

4. Explicit Declaration of dynamic:

- **var:** If you assign a value to var, Dart infers the type, and it can't change later.
- **dynamic:** If you declare a variable as dynamic, it explicitly allows you to change its type during runtime.

You can explicitly declare a variable as dynamic to make it flexible:

```
dynamic value;
value = 'text';  // First it's a String
value = 42;      // Then it's an int
```

Data Type: var

- The 'var' keyword is used for dynamically typed variables. The type is inferred at runtime.

- Example:

```
var age = 30;
var name = 'Alice';
```

Operators in Dart

- Dart supports typical operators:

- Arithmetic: +, -, *, /

- Comparison: ==, !=, <, >

- Logical: &&, ||

- Assignment: =, +=, -=, etc.

List Types in Dart

- Lists are a collection of ordered elements.
- Dart provides several types of lists:

- Fixed-length list

- Growable list

- Unmodifiable list

Fixed-Length List

- A fixed-length list cannot change its size after creation.

- Example:

```
List<int> numbers = List.filled(5, 0);
numbers[0] = 10;
```

Growable List

- A growable list can expand or shrink in size.

- Example:

```
List<String> names = [];
names.add('Alice');
names.add('Bob');
```

Unmodifiable List

- An unmodifiable list cannot be changed after it's created.

- Example:

```
List<String> fruits = List.unmodifiable(['apple', 'banana']);
```

Accessing List Elements

- List elements are accessed by index.

- Example:

```
List<String> colors = ['red', 'green', 'blue'];
print(colors[0]); // Output: red
```

Modifying List Elements

- You can modify elements in a growable or fixed list.

- Example:

```
List<int> nums = [10, 20, 30];
nums[1] = 40;
```

List Methods

- Dart provides useful list methods such as:

- add()

- remove()

- insert()

- clear()

- Example:

```
List<String> items = [];
items.add('item1');
items.remove('item1');
```

Maps in Dart

- Maps are a collection of key-value pairs.

- Example:

```
Map<String, String> capitals = {
'USA': 'Washington, D.C.',
'France': 'Paris'
};
```

Control Flow Statements

- Dart supports typical control flow statements:

- if, else

- switch

- for, while, do-while

- Example:

```
if (age > 18) {
print('Adult');
} else {
print('Not an Adult');
}
```

Looping in Dart

- Dart supports looping constructs like:

- for loop

- while loop

- Example:

```
for (int i = 0; i < 5; i++) {
    print(i);
}
```

Functions in Dart

- Functions in Dart are declared using the 'void' or data type keyword.

- Example:

```
void sayHello() {
print('Hello!');
}
```

- Functions can also return values:

```
int add(int a, int b) {
return a + b;
}
```

Conclusion

Dart is a versatile language suitable for web, server, and mobile applications. Its integration with Flutter makes it a great choice for modern app development.