# Basic Codes of Dart Language

## 1. Variables

Used to store data values. Declared using `var`, `final`, or `const`.  
  
Example:  
```dart  
void main() {  
 var name = "John"; // Variable that can change.  
 final age = 25; // Immutable variable, set once.  
 const pi = 3.14; // Compile-time constant.  
}```

## 2. Data Types

Common types include `int`, `double`, `String`, `bool`, and `List`.  
  
Example:  
```dart  
void main() {  
 int number = 10;  
 double price = 99.99;  
 String greeting = "Hello";  
 bool isDartEasy = true;  
 List<int> numbers = [1, 2, 3];  
}```

## 3. Functions

Defined using the `returnType functionName(parameters)` syntax.  
  
Example:  
```dart  
int add(int a, int b) {  
 return a + b;  
}  
  
void main() {  
 print(add(3, 5)); // Output: 8  
}```

## 4. Control Flow Statements

Conditional statements like `if`, `else`, and loops like `for`, `while`.  
  
Example:  
```dart  
void main() {  
 int number = 10;  
  
 if (number > 5) {  
 print("Greater than 5");  
 } else {  
 print("5 or less");  
 }  
  
 for (int i = 0; i < 3; i++) {  
 print(i); // Output: 0, 1, 2  
 }  
}```

## 5. Classes and Objects

Dart supports Object-Oriented Programming (OOP) with `class` and `object`.  
  
Example:  
```dart  
class Person {  
 String name;  
 int age;  
  
 Person(this.name, this.age);  
  
 void greet() {  
 print("Hello, my name is \$name.");  
 }  
}  
  
void main() {  
 Person john = Person("John", 25);  
 john.greet(); // Output: Hello, my name is John.  
}```

## 6. Null Safety

Dart prevents null errors by requiring nullable types to be explicitly declared.  
  
Example:  
```dart  
void main() {  
 String? nullableString = null; // Nullable type.  
 print(nullableString); // Output: null  
}```

## 7. Collections

Includes `List`, `Set`, and `Map`.  
  
Example:  
```dart  
void main() {  
 List<int> numbers = [1, 2, 3]; // Ordered collection.  
 Set<int> uniqueNumbers = {1, 2, 2, 3}; // Unique items.  
 Map<String, int> ages = {"Alice": 30, "Bob": 25}; // Key-value pairs.  
  
 print(numbers[0]); // Output: 1  
 print(uniqueNumbers); // Output: {1, 2, 3}  
 print(ages["Alice"]); // Output: 30  
}```

## 8. Asynchronous Programming

Uses `Future` and `async`/`await` for async tasks.  
  
Example:  
```dart  
Future<String> fetchData() async {  
 await Future.delayed(Duration(seconds: 2));  
 return "Data loaded";  
}  
  
void main() async {  
 print("Fetching...");  
 String data = await fetchData();  
 print(data); // Output: Data loaded  
}```

## 9. Exceptions

Handle errors using `try`, `catch`, and `finally`.  
  
Example:  
```dart  
void main() {  
 try {  
 int result = 10 ~/ 0; // Throws an exception.  
 } catch (e) {  
 print("Error: \$e");  
 } finally {  
 print("Execution completed.");  
 }  
}```

## 10. Arrow Functions

Simplified syntax for single-expression functions.  
  
Example:  
```dart  
int square(int x) => x \* x;  
  
void main() {  
 print(square(4)); // Output: 16  
}```