**By: Eng. Elias Al-Showaiter**

## [Dart Programming Language] - Lecture [12]

# [Exercises part 3, miscellaneous topics]

### 1. Introduction:

Although we have covered most of the fundamentals ideas in dart that well help us have a good start in flutter. There is a few topics we need to cover to be ready 100%. These topics are null safety, async/await, const and final, and cascade notation. We will explore these topics with a single example.

Q. You are working on a Dart project that involves managing product orders. The task requires you to integrate the following concepts into a single code example:

- **Null Safety**: Some product fields may or may not have values.
- **Async and Await**: Orders should be processed asynchronously, simulating a network delay.
- **Const and Final**: Some variables should remain immutable after initialization.
- **Cascade Notation**: Use the cascade operator (..) to simplify method calls on the same object.

Using the following data model, write a Dart program that:

1. Creates a list of products, some with nullable fields.
2. Creates an order using this list of products.
3. Calculates the total price of the order.
4. Defines a const map for shipping options.
5. Processes the order asynchronously with a delay.
6. Uses final for a variable that stores the order status.
7. Applies cascade notation in at least one part of the code.

```dart
class Product {
  final int id;
  final String name;
  final double price;
  final bool? isAvailable; // Nullable field

  Product(this.id, this.name, this.price, {this.isAvailable});
}

class Order {
  final int orderId;
  final List<Product> products;
  bool isProcessed = false;

  Order(this.orderId, this.products);

  double calculateTotalPrice() {
    return products.map
              ((product) => product.price).reduce((a, b) => a + b);
  }

  void markAsProcessed() {
    isProcessed = true;
  }

  Future<void> processOrder() async {
    await Future.delayed(Duration(seconds: 2)); // Simulate network delay
    markAsProcessed();
    print('Order ${orderId} processed.');
  }
}

void main() async {
  // Step 1: Creating a list of products, some with nullable fields
  final products = [
    Product(1, 'Laptop', 1200.0, isAvailable: true),
    Product(2, 'Phone', 800.0, isAvailable: false),
    Product(3, 'Keyboard', 50.0), // Nullable field omitted
  ];

  // Step 2: Create an order using the list of products
  final Order order = Order(101, products);
```

```
  // Step 3: Calculate total price using a cascade notation
  final Order totalPrice = order..calculateTotalPrice();
  print(
      'Total price for order ${order.orderId}:
                        \$${order.calculateTotalPrice()}');

  // Step 4: Create a const map of shipping options (immutable)
  const Map<String, double> shippingOptions = {
    'standard': 5.0,
    'express': 10.0,
  };

  print('Available shipping options: $shippingOptions');

  // Step 5: Process the order asynchronously
  await order.processOrder();

  // Step 6: Use final to store some calculated value (immutable after
initialization)
  final String orderStatus = order.isProcessed ? 'Processed' : 'Pending';
  print('Order status: $orderStatus');
}
```