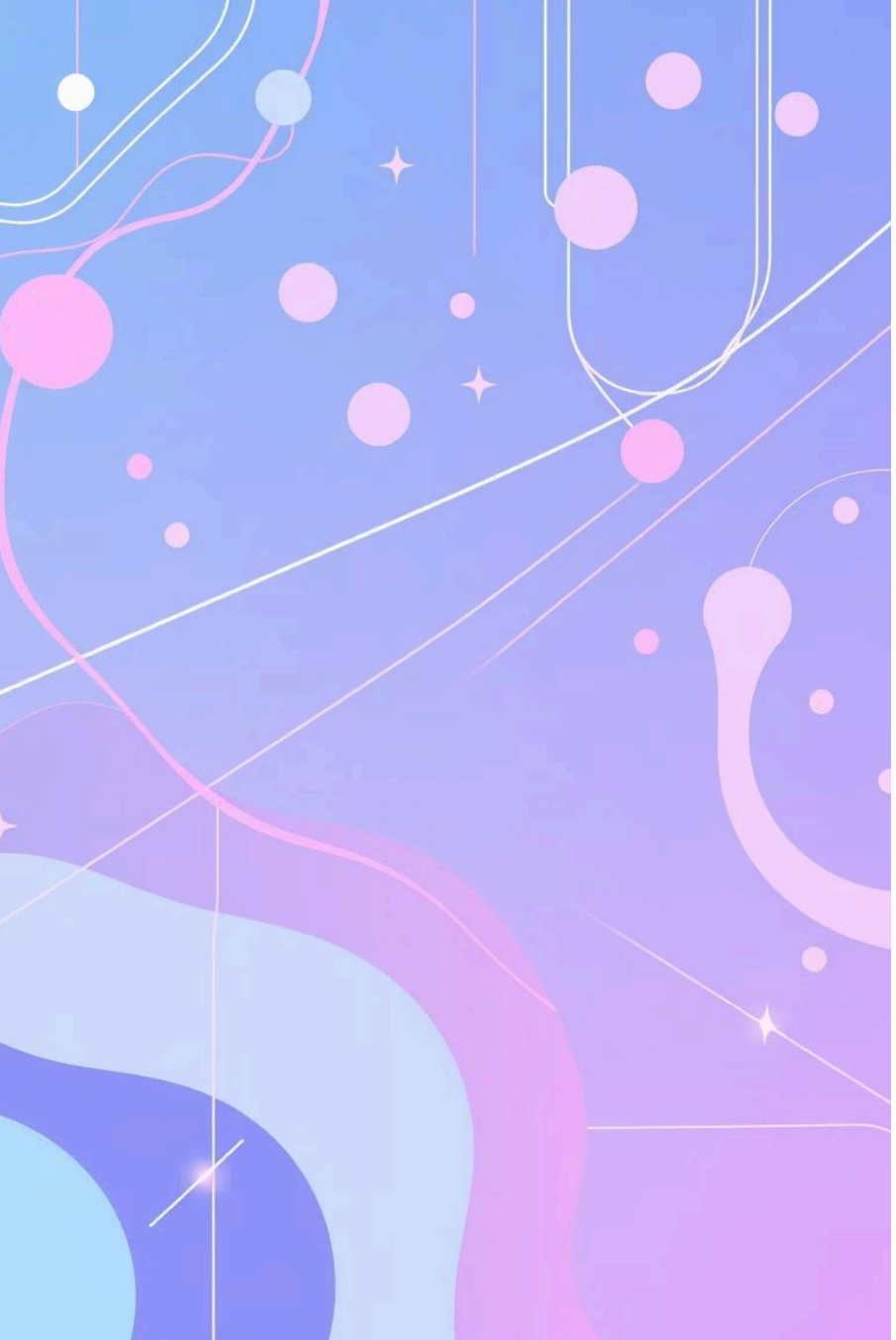


Dart Session Two:

. Collections

. Functions

. Null Safty

Non primitive _ Collections in Dart

List (fixed _growable)

An ordered collection of items, allowing duplicate values.

- Common Operations
- add(),
- remove(),
- list[index]

Set

An unordered collection of unique items.

Common Operations:

add(),

remove(),

contains()

Map

A collection that stores values as key-value pairs.

- Common Operations:
- map[key] = value,
- remove(),
- map[key]

Below are practical examples demonstrating the basic usage of each collection type:

List Examples

```
List<String> fruits = ['apple', 'banana'];
fruits.add('orange'); // Add item
print(fruits[0]); // Access 'apple'
fruits.remove('banana'); // Remove item
print(fruits); // Output: [apple, orange]
```

Set Examples

```
Set<int> numbers = {1, 2, 3};
numbers.add(4); // Add item
numbers.add(2); // Ignored, 2 already exists
print(numbers.contains(1)); // Check: true
numbers.remove(3); // Remove item
print(numbers); // Output: {1, 2, 4}
```

Map Examples

```
Map<String, int> ages = {'Alice': 30, 'Bob': 24};
ages['Charlie'] = 35; // Add item
print(ages['Alice']); // Access 30
ages.remove('Bob'); // Remove item
print(ages); // Output: {Alice: 30, Charlie: 35}
```

Task

List)

1) print first and last elemnt of [1, 2, 3, 4] .

2)on this List ["Ali", "Omar", "Sara"] add =>"Mona" And remove => " Omar"

Set)

1) remove duplicates of list [1,1,2,3,5,5]

map)

Create a Map that stores your **name**, **age**, and **university**.

- Print all values.
- Add a new key `"city"` with your city name.
- Print the updated Map.

For-In Loops in Dart

The basic syntax is straightforward: `for (var item in collection) { // do something with item }`.

For-in with Lists

```
void main() {
  List<String> fruits = ["Apple", "Banana", "Cherry"];
  for (var fruit in fruits) {
    print(fruit);
  }
}
```

For-in with Maps

```
void main() {
  Map<String, String> user = {
    "name": "Alice",
    "city": "NY"
  };
  for (var key in user.keys) {
    print("Key: $key");
  }
  for (var entry in user.entries) {
    print("Entry: ${entry.key}: ${entry.value}");
  }
}
```

Task

```
var products = { "Laptop": 15000, "Phone": 8000, "Headphones": 1200, "Keyboard": 600, };
```

find max price ;

Functions in Dart

Functions are reusable blocks of code that perform specific tasks.

Basic Function Declaration & Calling

```
void greet() {
  print("Hello, Dart!");
}

void main() {
  greet(); // Calling the function
}
```

Functions with Parameters and Return Values

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

void main() {
  int sum = add(5, 3);
  print("Sum: $sum"); // Output: Sum: 8
}
```

Functions with Named Parameters

```
String GreetPerson({String name = "Guest", String greeting = "Hello"}) {
  return "$greeting, $name!";
}

void main() {
  print(GreetPerson(name: "Charlie")); // Output: Hello, Charlie!
  print(GreetPerson(greeting: "Hi", name: "David")); // Output: Hi, David!
}
```

Task

1) function that take your your name , age , and display it .

2) Write a function **printEvenNumbers** that takes a **List of numbers** as prametar and prints all the even numbers.

Arrow Function Syntax

```
int multiply(int a, int b) => a * b;

String sayHello(String name) => "Hello, $name!";

void main() {
  print(multiply(4, 5)); // Output: 20
  print(sayHello("Eve")); // Output: Hello, Eve!
}
```

Anonymous Function?

```
var multiply = (int a, int b)

{

return a * b;

};

void main()

{

 var numbers = [1, 2, 3, 4];

numbers.forEach((n) { print(n * 2); });

}
```

Where Function in Dart

The `where` used on collections to filter their elements based on a specified condition.

It returns a new `Iterable`.

The basic syntax is: `collection.where((element) => condition)`.

Common use cases include filtering data from lists, searching for specific items, or preparing subsets of data for further processing.

Practical Examples

Converting to List using `.toList()`

Since `where` returns an `Iterable`, use `.toList()` to get a mutable list if needed.

```
  List<int> numbers = [1, 2, 3, 4, 5, 6];

  List<int>evenNum=numbers.where((n)=> n%2==0 ).toList();

print(evenNum);
```

Task

applay (**Arrow Function Anonymous Function, where)**

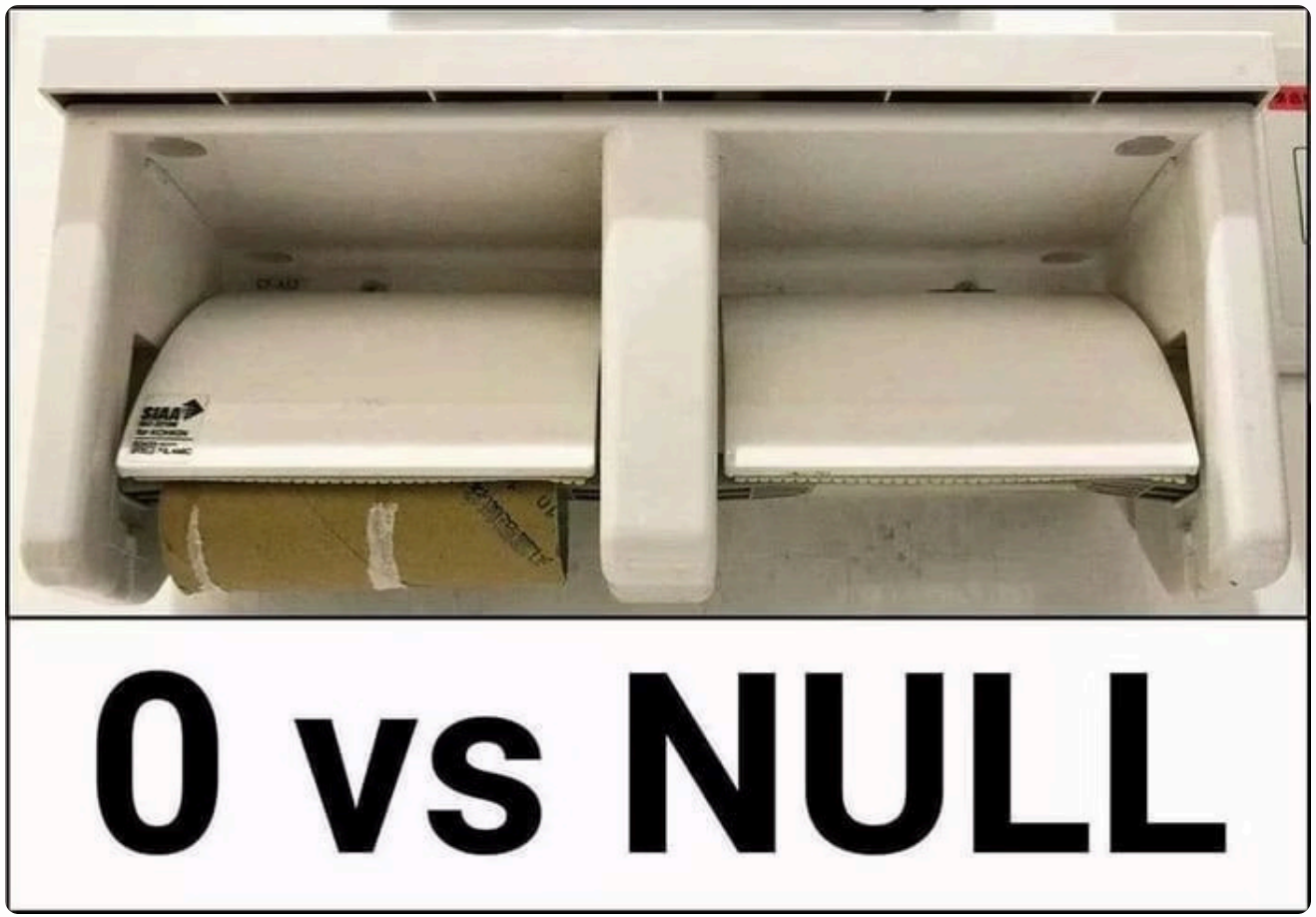
1) var names = ["Ali", "Omar", "Ahmed", "Sara", "Anwar"]; ==========> return the elemnts that only start with (A) .

2)var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]; =======> return numbers that grater than (5).

3)var students = Return the names of only the passed students

```
[
{"name": "Ali", "score": 45},
{"name": "Omar", "score": 55},
{"name": "Sara", "score": 70},
{"name": "Ahmed", "score": 90},
];
```

what is Null?



Problem with Null Values?

```
String name;
print(name.length);
```

The Problem with Null Values

```
String? getNameById(int id) {
  if (id == 1) {
    return 'karim';
  }
  return null; // Potentially returns null
}

void greetUser(int userId) {
  String userName = getNameById(userId); // Problem: userName could be null here!
  print('Hello, ${userName.toUpperCase()}'); // This line would crash if userName is null
}
```

Null Safety in Dart

Non-Nullable by Default

In Dart, variables are non-nullable by default.

```
String name = "Alice"; // Cannot be null
```

Nullable Types

```
String? city = null; // Can be null
```

Null Assertion Operator (!)

If you are certain that a nullable expression is not null at runtime, you can use the null assertion operator (!)

```
String? username = "John";
print(username!.length); // Assumes username is not null
```

The late Keyword

The late keyword is used for variables that will be initialized later but are guaranteed to be non-null before they are used. This is useful for circular dependencies or when initialization depends on a constructor argument.

```
late String description;
// ... later
description = "A detailed overview.";
```

TASK:

1. Write a function called `getProductPrice(String productName)`:
    - If `productName == "phone"` → return `10000`
    - Otherwise → return `null`

Null-check Operator (??)

if null then assign »

```
String? userName = null;
String displayName = userName ?? 'Guest';
print(displayName); // Output:

String? productPrice = '25.99';
String displayPrice = productPrice ?? 'N/A';
print(displayPrice); // Output:
```

الرمز	المعنى
?	nullable type
!	assert non-null
??	if null, use fallback

assignment

Task:

Write a Dart program that calculates the total price for a list of clothing items (T-Shirts & Pants) with quantity. If the total exceeds $80, apply a 10% discount.

Requirements:

- Each item has: `name`, `price`, `quantity`.
- Calculate total price per item and overall total.
- Apply 10% discount if overall total > $80.