جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

جامعة المستقبل
كلية العلوم – قسم الانظمة الطبية الذكية

Lecture: (1)

Subject: Dart Programming Language
Class: Third
Lecturer: Asst.Lec Mohammad Baqer Haleem

Dart Programming Language

By : Asst.Lec Mohammad Baqer Haleem

I. Introduction

Dart is a general purpose programming language developed by Google. You can use it to develop web, server, desktop and mobile applications for iOS and Android. It's easy to learn (especially if you read this book) and also has great tooling support.

II. Variables & Constants 🧮

`int n1 = 10;`	stores integers (whole numbers), without decimals, such as 123 or -123
`double n2 = 12.23;`	stores floating point numbers, with decimals, such as 19.99 or -19.99
`String n3 = 'myString';`	stores text, such as "Hello World". String values are surrounded by double quotes
`bool n4 = true;`	stores values with two states: true or false
`var n5 = 12;`	The var keyword says to Dart, “Use whatever type is appropriate.”

`dynamic n6 = '12112';`
`n5 = 12.12;`
`n5 = 12;`
	lets you assign any data type you like to your variable

`num n7 = 12.5;` `num n7 = 12;`	store both an integer and floating-point value
`const n10 = 10;`	constants when compile-time
`final n11 = n1 + n2;`	constants when runtime

III. List and Mapping 🧮

List

```dart
void main() {
 List<int> myList = [1, 3, 5, 1, 5, 8, 9];
 myList.forEach((element) {
   print(element);
 });
 myList.removeAt(5);
 myList.sort();
}
```

Mapping

```dart
void main() {
 Map<String, dynamic> student = {
   'id': 12323,
   'name': 'Phanith',
   'age': 20,
   'address': '123 Main St',
   'phone': '555-555-5555'
 };
 print(student['name']);

 //Modify the student map to add a new key/value pair
 student['name'] = 'John';

 //forEach loop to print out all the key/value pairs
 student.forEach((key, value) {
   print('$key: $value');
 });

 //Remove the phone key/value pair
 student.clear();
}
```

IV. Control Flow 🕹️

If else statement

```
const animal = 'Fox';
if (animal == 'Cat' || animal == 'Dog') {
     print('Animal is a house pet.');
} else {
     print('Animal is not a house pet.');
}
```

If else-if statement

```
const trafficLight = 'yellow';
var command = '';
if (trafficLight == 'red') {
 command = 'Stop';
} else if (trafficLight == 'yellow') {
 command = 'Slow down';
} else if (trafficLight == 'green') {
 command = 'Go';
} else {
 command = 'INVALID COLOR!';
}
print(command);
```

IV. Control Flow 🕹️

switch statement

```
const number = 3;
switch (number) {
 case 0:
   print('zero');
   break;
 case 1:
   print('one');
   break;
 case 2:
   print('two');
   break;
 case 3:
   print('three');
   break;
 case 4:
   print('four');
   break;
 default:
   print('something else');
}
```

Noted:

switch statement is usually faster than a series of if-else statements when there are many cases to check. This is because a switch statement uses a table lookup to determine which case to execute, while a series of if-else statements requires a linear search through each condition until a match is found.

IV. Control Flow 🕹️

While Loop

```
while (condition) {
  // loop code
}
```

Do-while Loop

```
do {
  // loop code
} while (condition)
```

For Loop

```
for(Initialization; condition; incrdecr) {
  // loop body
}
```

For-in Loop

```
List<int> myList = [1,2 ,1, 1, 4];
for(var num in myList){
      print("item: $num");
}
```

ForEach Loop

```
List<int> myList = [3,3 ,3, 4];
myList.forEach((element) {
      print("item: $element");
});
```

Noted:

forEach is that it is often faster and more efficient than using a traditional for loop, especially for large collections. This is because forEach uses an internal iterator that is optimized for performance, while a traditional for loop with an index variable may be slower due to the overhead of incrementing and checking the index variable on each iteration.

V. Function 👩🏽‍🚀

In programming, a function is a block of code that performs a specific task or set of tasks. Functions can be thought of as small, self-contained units of code that can be reused throughout a program.

V. Function 👩🏽‍🚀

Take parameters, Return values

```
int sum(int a, int b) ⇒ a + b;
void main() {
     int result = sum(5, 7);
}
```

Take parameters, No return values

```
void printMessage(String message) {
     print(message);
}
void main() {
     printMessage("Hello, world!");
}
```

No parameters, Return values

```
import 'dart:math';
int getRandomNumber() {
     final random = Random();
     return random.nextInt(10) + 1;
}

void main() {
     int randomNum = getRandomNumber();
}
```

No parameters, No return values

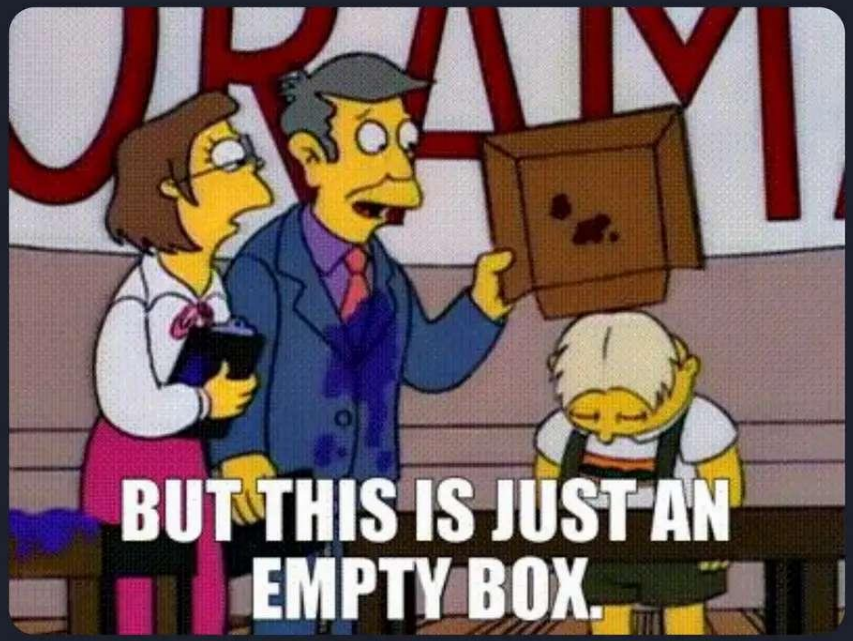
```
void printList(List<int> numbers) {
     for (final number in numbers) {
          print(number);
     }
}

void main() {
 final List<int> numbers = [1, 2, 3, 4, 5];
 printList(numbers);
}
```

VI. Nullability - Null Safety 📪

Null Safety in simple words means a variable cannot contain a ‘null’ value unless you initialized with null to that variable. With null safety, all the runtime null-dereference errors will now be shown in compile time.

```
final String msg1 = null;      Error Null can’t be assigned to a variable
final String? msg2 = null;     No error
```



VI. Nullability - Null Safety 📪

Bug

```
void crushMessage(String msg) {
     print("MESSAGE: $msg");
}
void main() {
     final String? name = "Hey, I'm a string!";
     crushMessage(name); Error Null can’t be assigned to a variable
}
```

Solution

```
void crushMessage(String msg) {
     print("MESSAGE: $msg");
}
void main() {
     final String? name = "Hey, I'm a string!";
     crushMessage(name!);
}
```