PRESENTATION TOPIC: FLUTTER

Presented by: Galib Mahmud

Software Developer Intern at The ICT Hub

Mohammadi Housing Ltd.

Mohammadpur,Dhaka.

Contact: 01581027072

what is flutter ?

Flutter is an UI toolkit and Framwork developed by Google published on May 2017.

It uses dart programming language which released on November 2013. It allows developers

for building cross-platform applications. Like android,ios, web,macOS,desktop etc. Many well Known

companies are using Flutter for a rich set of tools and features to build beautiful and high

performance applications such as : Google ,Alibaba , eBay ,BMW, Space X etc.

Uses of Flutter :

In todays era, Flutter is one of the most popular and powerful tools for app development. Businesses want app for Andriod, iOS, Web, and Desktop all from a single codebase. By allowing this, Flutter saves time, cost, and resources. Flutter is the first choice of many developers and customers because of its smoothness, beautiful interface, custom widgets etc .

Key Features and Advantages:

Cross-Platform Development:

Flutter enables developers to build applications for multiple platforms (iOS, Android, web, Windows, macOS, Linux) with a single codebase, reducing development time and effort.

Fast Development Cycle:

Flutter's hot reload feature allows developers to see changes to the UI in real-time, speeding up the development process.

Rich Widget Library:

Flutter provides a comprehensive set of pre-designed widgets, making it easier to build visually appealing and interactive user interfaces.

High Performance:

Flutter compiles code to native machine code (ARM or Intel) or JavaScript, resulting in fast and efficient applications.

Open Source and Free:

Flutter is an open-source framework, allowing developers to use it without licensing fees and benefit from a vibrant community.

Key Concepts:

Widgets:

The fundamental building blocks of Flutter applications, used to create the user interface.

Dart Programming Language:

Flutter uses Dart, a modern object-oriented programming language, for building applications.

Impeller:

Flutter's rendering engine, which provides high-performance rendering capabilities.

In summary, Flutter is a powerful and versatile framework that simplifies cross-platform app development, offering a rich set of tools and features to build beautiful and high-performance applications.

Install dart sdk:

need dart sdk (unzip file then paste into c drive copy bin path and set it into environment variable user path)

check it cmd and search dart -v

Install Flutter sdk :

need flutter sdk (unzip file to c drive and set environveriable path and search cmd flutter doctor)

Problems and solve:

1.(problem in open new flutter project sdk)

go to the folder and type

cmd and then

```
flutter create galib12
```

2.Increase the maximum number of file handle

Registry editors, H key local machine, system,current controll set, services

web clint, parameters, file size limit in Bytes,4294967295

or

cmd then: git reset --hard

```
C:\flutter\bin\cache\dart-sdk>git reset --hard
```

Common uses of dart :

01. Datatypes

```
void main() {
  print('Hello World');
  // Data types
  String name = 'Galib Mahmud';
  print(name);
  int age = 10;
  print(age);
  double height = 5.6;
  print(height);
  bool value = true;
  print(value);
  // List e


  List myList = [1, 2, 3, 4, 5];
  print(myList);
 // Map
  Map<String, dynamic> myMap = {'name': 'Galib', 'age': '24'};
  print(myMap);
  //set
  Set mySet = {1, 2, 3, 4, 5};
  print(mySet);
  //Rune
  final nameTwo = 'Hello';
  print(nameTwo.codeUnits);
// Runes (Unicode emoji)
  Runes input = Runes('\u{1f49b}'); // 💛
  print(String.fromCharCodes(input)); //
}
```

02.Dart Operators

Arithmetic Operator: + Add ,- Subtract,*multiply,/ Divide, % get the remainder

~/trancating division Operator,++ increment , -- decrement

Equality an drelational operator

Type test Operator

Bitwise operator

Assignment operator

Logical Operator

Conditional Operator

Cascade notation Operator

```
void main() {
  int a = 10;
  int b = 3;
  var ans = a ~/ b;
  print(ans);
 //Type test
  String name = 'Galib Mahmud';
  var result = name is String;
  print(result);
// Tarnary operator
  String colour = 'white';
  var result1 = colour == 'red' ? 'colour is red' : 'colour is not red';
  print(result1);
  String color = 'red';
  String result2;
  if (color == 'red') {
    result2 = 'colour is red';
  } else {
    result2 = 'colour is not red';
  }
  print(result2);
 // Null aware operator
  int? a = null;
  int b = 10;
  var result3 = a ?? b; // if a is null, return b
  print(result3);
// Spread operator
  List<int> list1 = [1, 2, 3];
  List<int> list2 = [4, 5, ...list1];
 print(list2);
// Null aware spread operator
  List<int>? list3;
  List<int> list4 = [6, 7, ...?list3];
  print(list4);
}
```

03.Conditional Statement

```
void main() {
  String connection = 'waiting';
  if (connection == 'connected') {
    print("connected");
  } else if (connection == 'waiting') {
    print("waiting");
  } else {
    print("Not connected");
  }
}
```

```
void main() {
  String connection = 'waiting';
  switch (connection) {
    case 'waiting':
      print('Waiting for connection');
      break;
    case 'connected':
      print('Connected');
```

```
     break;
    case 'disconnected':
      print('Disconnected');
      break;
    default:
      print('Unknown connection status');
  }

}
```

04.String Concatination & interpolation

```
void main() {
  // String concatination or addition
String a = 'we';
  String b = 'are';
  String c = 'lerning';
  String d = 'Dart';
 //Concatination
  print(a + b + c + d);
// Interpolation
  print("$a$b$c$d);
}
```

05.Compile time error and run time error

```
void main() {
  // String concatination or addition
 String a = 'we';
  String b = 'are';
  String c = 'lerning';
  String d = 'Dart';

  //Concatination
  print(a + b + c + d);

  // Interpolation
  print("$a$b$c$d)
  // syntex error or compile time error

  int a = 10;
  int b= 0;
  var result = a~/b;
  print (result);
// run time error

}
```

....
....

06.Final and cons

```
void main() {
  // finial korle change korte parbo na
  final int age = 50;
  // age=30; eita se nibe na
  print(age);

  //const onno kono variable r valu assign korte parbe na

  const double pi = 3.1416;
  // declear korar por compile time a memmory te aloocate hobe
  print(pi);
}
```

07.List

```
void main() {
  // finial korle change korte parbo na
  final int age = 50;
  // age=30; eita se nibe na
  print(age);

  //const onno kono variable r valu assign korte parbe na

  const double pi = 3.1416;
  // declear korar por compile time a memmory te aloocate hobe
  print(pi);
}
```

08.Enumeration

```
import 'dart:vmservice_io';
```

```
void main() {
  final gender = Gender.Female;
  switch (gender) {
    case Gender.Male:
      print("Gender is Male");
      break;

    case Gender.Female:
      print("Gender is Female");
      break;

    case Gender.Unknown:
      print("Gender is Female");
      break;

    default:
      print("Nothing Matched");
  }
}

enum Gender { Male, Female, Unknown }
```

09. Function

```
import 'dart:vmservice_io';

void main() {
```

```
// function, here main function is top level function
 addTwoNumbers(int a, int b) {
   return a + b;
 }

 // function call
 var result = addTwoNumbers(5, 10);
 print('The sum is: $result'); // Output: The sum is: 15

 // Returning a function
 exampleFunction() {
   return (int x, int y) {
     return x * y;
   };
 }

 // Calling the returned function
 var multiply = exampleFunction();
 var product = multiply(4, 5);
 print('The product is: $product'); // Output: The product is: 20

 //higher order function
 mainFunction() {
   // Function that takes another function as an argument
   void higherOrderFunction(int a, int b, Function operation) {
     var result = operation(a, b);
     print('The result is: $result');
   }
```

```
    // Passing a
  }
}
```

10.User input

```
import 'dart:io';
import 'dart:vmservice_io';

void main() {
  // User input

  print("Enter your name:");
  // var name = stdin.readLineSync();
  String? name = stdin.readLineSync();
  print("Hello, $name!");
}
```

11. Loop

```
void main() {
  List CotactList = ['0123456789', '0987654321', '1234567890', '9876543210'];

  // Print the list using a for loop
  for (int i = 0; i < CotactList.length; i++) {
    print(CotactList[i]);
  }
```

```
// For + break and continu
for (var i = 0; i < 10; i++) {
  if (i == 5) {
    continue; // Skip the rest of the loop when i is 5
  }
  if (i == 8) {
    break; // Exit the loop when i is 8
  }
  print(i);
}

// Print the list using a for-in loop
for (var contact in CotactList) {
  print(contact);
}

// for each loop
var myList = [
  {'Name': 'Name one'},
  {'Name': 'Name two'},
  {'Name': 'Name three'},
];
myList.forEach((element) {
  print(element['Name']);
});
//while loop
int i = 0;
while (i < CotactList.length) {
  print(CotactList[i]);
```

```
   i++;
  }

  // do while loop
  int j = 0;
  do {
   print(CotactList[j]);
   j++;
  } while (j < CotactList.length);
}
```

12. Null Safety

```
void main() {
  // Null safety
  String? name = null;
  convertStringIntoint('55');
}

convertStringIntoint(value) {
  print(int.parse(value));
}
```

13.Exception Handeling

```
void main() {
  // Exception Handeling -try, catch, strack- trace,finally
  // Formate Exception
  int result = int.parse('44s');
  print(result.runtimeType);
  try {
    int result = int.parse('44s');
    print(result.runtimeType);
  } catch (e) {
    print(e);
  }
  // strack trace or s print korle je library gula break hoiche oita dekhabe
  try {
    int result = int.parse('44s');
    print(result.runtimeType);
  } catch (e, s) {
    print(e);
    print(s);
  } finally {
    print('Finally block executed');
  }

  // Custom Exception
  try {
    throw MyException('This is a custom exception');
  } catch (e) {
    print(e);
  }
}
```

……………………………………………………………

……………………………………………………………

14.Asynchronous Programming

```
void main(){
  // Asynchronous function
print('line 1');
  print('line 2');
  Future.fetchData() async {



Future.delayed(Duration(seconds: 3), () => print('line 3'));
  }fetchData();
  print('line 4');
  print('line 5');

}
```

15.Class Object

```
class Example {
  int age = 10;
  String name = 'Galib Mahmud';
  myFunction() {
    print('Hello, My name is Galib Mahmud ');
  }

  myFunction2() {
    print('Opps ');
  }
}
imporrt 'galib.dart';
void main() {
  var obj = Example();
  print(obj.age);
  print(obj.name);
  obj.myFunction();
  obj.myFunction2();
}
```

16.Constractor

```
void main() {
  // Dart Constructor
  var obj = Example('Galib Mahmud');
}

class Example {
  String name;

  // Constructor with parameters
  Example(this.name) {
    print("This is my deafult constractor ");
    print(name);
  }
}
```

17.Static Keyword

```
void main() {
  // static keyword
  // static korle direct class r sathe somporko thake

  print(Galib.name);
// static na thakle .name diye acess korte partam
  Galib.displayInfo();
}

class Galib {
  static String name = "Galib";
  static int age = 25;

  static void displayInfo() {
    print("Name: $name, Age: $age");
  }
}
```

18. Inheritance

```
void main() {
  var son = Son();
  son.display();
  // This will call the Son's display method

  son.name();
}

class Father {
  int age = 50;
  name() {
    print("Father's Name Sazzad Hosain");
  }
}

class Son extends Father {
  int age = 20;

  @override
  void display() {
    print("Son's age is $age");
  }
}
```

19.Polimorphism

```
void main() {
  var son = Son();
  son.display();
  // This will call the Son's display method

  son.name();
}

class Father {
  int age = 50;
  name() {
    print("Father's Name Sazzad Hosain");
  }
}

class Son extends Father {
  int age = 20;

  void display() {
    print("Son's age is $age");
  }
// override method
  name() {
    print('GALIb');
  }
}
```

Basic Widget in Flutter:

1. Structural Widgets (Layout)

Used to organize the UI structure.

- `Container`
- `Row`
- `Column`
- `Stack`
- `Expanded`
- `SizedBox`
- `Padding`
- `Align`
- `Center`
- `Wrap`

---

2. Display Widgets (Visual/Styling)

Used to show visual content like text, images, icons.

- `Text`
- `RichText`
- `Image`
- `Icon`
- `FlutterLogo`
- `FadeInImage`
- `CircleAvatar`

---

3. Input Widgets (Interactive / Form Fields)

For user input and interaction.

- `TextField`
- `TextFormField`
- `Checkbox`
- `Radio`
- `Switch`
- `Slider`
- `DropdownButton`
- `Form`

---

4. Button Widgets

To trigger actions.

- `ElevatedButton`
- `TextButton`
- `OutlinedButton`
- `IconButton`
- `FloatingActionButton`
- `InkWell` (for custom tap detection)

---

5. Layout Control Widgets

Manage size, space, and alignment.

- `Flexible`
- `Spacer`
- `AspectRatio`
- `FittedBox`
- `LayoutBuilder`
- `ConstrainedBox`
- `FractionallySizedBox`

---

6. Navigation Widgets

Used for navigation and routing.

- `Navigator`
- `MaterialPageRoute`
- `PageView`
- `Drawer`
- `BottomNavigationBar`
- `TabBar` and `TabBarView`

---

7. Animation & Motion Widgets

Used for animations and transitions.

- `AnimatedContainer`

- `AnimatedOpacity`
- `Hero`
- `AnimatedBuilder`
- `FadeTransition`
- `ScaleTransition`
- `AnimatedSwitcher`

---

8. State Management Widgets

Used for rebuilding UI based on state changes.

- `StatefulWidget`
- `StatelessWidget`
- `InheritedWidget`
- `Provider`, `Bloc`, `Riverpod` (via packages)

---

9. Scaffold & App Structure

Used to build basic app layout.

- `Scaffold`
- `AppBar`
- `MaterialApp`
- `CupertinoApp`
- `Theme`

---

10. Scrollable Widgets

For scrollable content.

- `ListView`
- `SingleChildScrollView`
- `GridView`
- `PageView`
- `CustomScrollView`
- `Scrollbar`

---

11. Dialog & Popups

To show alerts or dialogs.

- `AlertDialog`
- `SimpleDialog`
- `BottomSheet`
- `SnackBar`
- `ModalBottomSheet`