1. Dart Introduction

Datr- Dart is a Client Optimized language(client side devloped) for fast apps on any platform(android , ios,linux,windows,mac)

~ Any platform menans cross platform

2. Environment Setup

3. Dart Hello World

~Dart Program

The main() function is a predefined method in dart
This method acts as the entry point to the application

```
@03_Dart_Hello_World.dart
    void main()
    {
        print("hello ");
    }
```

4. Dart Project Structure in Intellij Idea

1. .idea -> Intellij Ideas configuratio folder
2. bin -> Main function of Dart
3. lib -> Extra File store and connect main function
4. test -> for testing
5. .gitignore -> when we are push code into github then (.gitignore) use for remove unnecessary doc
6. CHANGELOG.md -> for Github
7. .iml -> auto genarated file
8. pubspec.yaml ->third party resourse (pub.dev)== official package repository for dart and flutter
9. read.md -> note and github push

5. Dart Kyeword and Syntax

~Dart Syntax:

1. Dart ignore Whitespace and Line Breaks (Space,new line,Tab)
2. Dart Case Sensitive (a,A are different)
3. Statement end with Semiclone
4. Comment in Dart
    1. Single Line comment (//) two backslash
    2. Multi Line comment (/*.....*/)

```
@05_Comment.dart
      void main()
       {
           print(hello world);//statement end with semiclone

           // print("hello world 1"); use single line comment

           /*
           print("hello 3");
           print("hello 3");
           print("hello 3");
           */

       }
```

6. Dart variable

~Variable: variable is used to store the value and reffer the memory location in computer

1. variable can not contain special character such as whitespace mathmatical symbol , unicode character and any kyeword
2. firt character of varable should be an alphabet(A-Z)(a-z).digit not allowed
3. Variable Case Sensitive
4. only underscore allow

```
@06_Dart_Variable.dart
  void main()
    {
      var x=10;
      var y=40;
      var z=x+y;
      print(z);
    }
```

7. Dart Data Type and Numbers

~Dart Data Type

1. Number
   1. Integer number(Non Fractional Number)
   2. Double Number(Fractional Number)

```
@07_Dart_DataType.dart
    void main()
   {
       var x=2;// non fractional number
       var y=3.0;// fractional number

       var z=x+y;
       print(z);// output:5.0 fractional number
   }
```

2. String
3. Boolean
4. Lists
5. Maps
6. Runes
7. Symbols

8. String and Boolean

~String(sequence of character)

[var str('hello')] [var str("hello")] both are correct

~ Boolean represent (True or False value)

```
@08_String_and_Boolean.dart
    void main()
    {
        // Example of String

        var str1= "Hello";
        var str2='Hello';
        print(str1);
        print(str2);

        // Example of Boolean
        var negative = false;
        bool positive = true ;
        print(negative);
        print (positive);
```

```
            }
```

9. Dart List

~ Dart list is a collection of ordered object(value).similar to array.
~ element in the list are separated by the comma and enclosed by square bracket[].
~We can Store any type of data into the list(string , int, double, boolean)
~ Index will Start by zero(0)

```
       @09_Dart_List.dart
          void main()
          {
              var list=['Dhaka', 0 , 3.1416, true];// store any type of data separate by comma

              print(list[0]); // index will start with Zero
          }
```

10. Dart Map

~ Map Type is used to store values in kye-value pairs
~ kye and value can be any type(kye/properties:value)
~ Map is defined by second bracket{}

```
       @10_Dart_Map.dart
               void main()
              {
                  var Person= {
                      'name':'khorsed',
                      "age":20,
                      'city':"Dhaka"
                  };
                  print (Person);// print all element of person
                  print(Person['age']);//print indevidual element
              }
```

11. Dart Operator

~Arithmetic Operator

1. Addition(+)
2. Subtraction(-)
3. Divide(/)
4. Multiplication(*)
5. Modulus(%)
6. Division(~/)
7. Unary Minus(-expr)

~Unary Operator

1. ++(prefix) increment
2. ++(postfix) increment
3. --(prefix) decrement
4. --(postfix) decrement

~Assignment Operator

1. = assign
2. (+=) add and assign
3. (-=) subtract and assign
4. (*=) Multiply and assign
5. (/=) Divide and Assign

```
        6. (~/) Divide and assign
        7. (%=) Mod and Assign

    ~Relational Operator
        1. (>) Greater than
        2. (<) Less than
        3. (<=) Less and Equal
        4. (>=) Greater and Equal
        5. (==) Equal
        6. (!=) Not Equal

    ~Type Test Operator
         1. (as) used for typecast
         2. (is)  it return true if the object has specified type
         3. (is!) it return true is the object has not specified type

    ~Logical Operator
         1. (&&) logical and
         2. (||) logical or
         3. ! logical not
    ~Bitwise Operator
         1. (&) Binary and
         2. (|) Binary Or
         3. (^) Binary XOR
         4. (~) Ones Compliment
         5. (<<) Shift left
         6. (>>) Shift right
```

12. Example of Arithmetic operator

```
   @12_Arithmetic_Operator.dart
      void main()
      {
          var a=10;
          var b=3;
          print(a+b);// output: 13
          print(a-b);// output:7;
          print(a*b);// output: 30;
          print (a/b);// output:3.33
          print(a%b);// output: 1
      }
```

13. Example of unary operator

```
    @13_Unary_Operator.dart
        void main()
        {
            var x= 10;
            print(++x);// output: 11

            var y= 10;
            print(y++);// output:10
            print(y++);// output:11

            var z=10;
            print(--z);// output: 9

            var a=10;
            print(a--);// output: 10
            print(a--);//output:9
        }
```

14. Dart Constant

~Dart Constant is defined as an immutable object(number that can not be change or reassigned)

~ Dart Constant define [final,const] kyeword

```
@14_Dart_Constant.dart
   void main()
   {
       final x=20; // use final kyeword
       print(x);

       const y=10;// using const kyeword
       print (y);
   }
```

15. List Properties

~Dart Builtin Properties

1.(.first) return first element of the list
2.(.last) teturn last element of the list
3.(.length) return list length
4.(.isEmpty) return True or false if list empty or not.
5.(.isNotEmpty) return True or False if list empty or not
6.(.reversed) return reversed value of the list
7.(.Single) return true or false if list have one element or not

```
@15_Dart_Properties.dart
    void main()
    {
        var city=['Dhaka','Rangput','Rajshahi','Bogura'];
        var result=city.length;// provide list size
        print(result);// output:4

        var result1=city.first;// provide first element of list
        print(result1);// output:Dhaka

        var result2=city.last;// provide last element of list
        print(result2);// output:Bogura

        var result3 =city.reversed;// provide reversed list
        print(result3);// output: Bogura,Rajshahi,Rangpur,Dhaka

        var result4=city.isEmpty;
        print(result4);// output:false

        var result5=city.isNotEmpty;
        print(result5);// output:true

        var result6=city.single;// It wil work when element number
one
        print(result6);// output:false
    }
```

16. Fixed length list growable list

~Fixed Length List

~The Fixed length lists are defined with the specified length

~length cannt be change

```
@16_Fixed_Length_List.dart
   void main()
   {
     const city=['Bogura','Nator'];
```

```
            print(city);
           }
```

~Groable List

~This List declared without specific size

~Size Can be Modified

@16_Groable_List.dart

```
    void main()
   {
       var City=['Hobigong','Munshigong','Jamalpur'];
       print(City);
       City.add('Cumilla');
       print(City);
   }
```

17. List insert items

~List Insert:Add element into the list by method

1. add() -> For Added only one element
2. addAll() -> For Added more then one element
3. insert() -> For Added one element on specific position
4. insertAll() ->For Added more then one element on Specific position

@17List_insert_items.dart

```
void main()
{
  var num=[1,2,3,4,5];
  num.add(6);// Add only one element
  print(num);// output:[1,2,3,4,5,6]

  num.addAll([7,8,9]);// added more than one elemnt
  print(num);// output:[1,2,3,4,5,6,7,8,9]

  num.insert(0,100);//added one element on specific position
  print(num);// output:[100, 1, 2, 3, 4, 5, 6, 7, 8, 9]

  num.insertAll(3,[10,20,30,40]);// added more than one element
  print(num);// output:[100, 1, 2, 10, 20, 30, 40, 3, 4, 5, 6,
7, 8, 9]

}
```

18. Dart list remove Update

~Dart element remove

1. (.removeLast)-> remove last element from dart list
2. (.removeAt)-> remove an element from specific index
3. (.remove)->remove specific value
4. (.removeRange)-> remove element using index range (start,end)

@18_Dart_list_remove_Update.dart

```
    void main()
  {
   var num=[10,20,30,40,50,60,70,80,90];

   //Dart list Updateing
   num[0]=1;// value update of 0th index
   num[1]=2;// value update of 1st index
   num[2]=3;// value update of 2nd index
   print(num);//output:[1, 2, 3, 40, 50, 60, 70, 80, 90]
```

```
        // Element remove from dart list
        num.removeLast();// remove last element from dart list
        print(num);// output:[1, 2, 3, 40, 50, 60, 70, 80]

        num.removeAt(2);//remove 2nd element from dart list
        print(num);// output:[1, 2, 40, 50, 60, 70, 80]

        num.remove(50);// remove specific value
        print(num);// output:[1, 2, 40, 60, 70, 80]

        num.removeRange(0,4);// remove element using index range
        print(num);// output:[70, 80]
      }
```

19. Dart set with Add and Add ALL

~Dart Set:Unordered Collection Of different values Same type
~ Doesnot allowed duplicate values
~ Set Must contain unique value

~ var variable_name=<generic name> {'Hobigang','Bogura'};

```
   @19_Dart_set_with_Add_and_Add_ALL.dart
       void main()
       {
         var cityset=<String>{'Dhaka','Barishal','Khulna'};
         print(cityset);// output:{Dhaka, Barishal, Khulna}

         cityset.add("dhaka");// added one element into the set
         print(cityset);// output:{Dhaka, Barishal, Khulna, dhaka}

        cityset.addAll({'CoxBazar','Rangpur'});// added multiple
element into the set
         print(cityset);// output:{Dhaka, Barishal, Khulna, dhaka,
CoxBazar, Rangpur}

       }
```

20. Set ElementAt() and Clear

~Access Specific one Element From Set using ElementAt() function

```
@20_Set_ElementAt()_and_Clear.dart
   void main()
   {
     // elementAt() fuction use
        var
mycityset=<String>{'Dhaka','Barishal','Bogura','Bhola'};
     print(mycityset.elementAt(3));

     // clear() function use
     mycityset.clear();// Clean all element of Dart Set
     print(mycityset);// output:{}

   }
```

21. Set properties

~Dart Builtin Properties
1.(.first) return first element of the list
2.(.last) teturn last element of the list
3.(.length) return list length
4.(.isEmpty) return True or false if list empty or not.
5.(.isNotEmpty) return True or False if list empty or not

```
6.(.reversed) return reversed value of the list
7.(.Single) return true or false if list have one element or not
8.(.hashcode) return hashcode for the corresponding object

@21_Set_properties.dart
   void main()
   {
    var city=<String>{'Dhaka','Nator','Bogura'};
    print(city.first); //output:Dhaka
    print(city.last);//output:Bogura
    print(city.length);//output:3
    print(city.isEmpty);//output:false
    print(city.isNotEmpty);//output:true
    print(city.single);//output:error massage if more than one
element
   }
```

22. Map add new Element

~To declare a map literal the kye value pairs are enclosed within the second brackets {} and separated by commas.

```
@22_Map_add_New_Element.dart

void main()
{
  var person={'name':'Khorsed','age':'20','city':'Bogura'};
  print(person);//output:{name: Khorsed, age: 20, city: Bogura}
  person['Country']='Bangladesh';
  print(person);//output:{name: Khorsed, age: 20, city: Bogura,
Country: Bangladesh}

 }
```

23. Map Constructor and Properties

~Declare the dart map using map constructor can be done in two ways.
   1. using map() constructor
   2. initialize the map

~Map builtin Properties
   1. keys
   2. values
   3. length
   4. isEmpty
   5. isNotEmpty

```
@23_Map_Constructor_and_Properties.dart
  void main()
  {
    var person= new Map();
    person['name']='Khorsed';
    person['age']='30';
    person['city']='Bogura';

    print(person);// output:{name: Khorsed, age: 30, city: Bogura}

    //(.keys)-> use to get all key
    print(person.keys);// output:(name, age, city)

    //(.values) -> use to get all values
    print(person.values);// output:(Khorsed, 30, Bogura)

    //(.length) -> use to get the length of Map
```

```
         print(person.length);// output:3

         // (.isEmpty)//if Map has no element then provide true
         print(person.isEmpty);// output:false

         // (.isNotEmpty) if Map has
         print(person.isNotEmpty);// output: true
       }
```

24. Map add Remove Clear

```
     ~24_Map_Add_Remove_Clear.dart
     void main()
     {
      var person ={
        'name':'Khorsed',
        'age':'23',
        'city':'Dhaka'

      };

      //Initial output
      print(person);//{name: Khorsed, age: 23, city: Dhaka}

       // (.addAll())-> used to add one or more element into the Map
       person.addAll({'Country':'Bangladesh'});
       print(person);//output:{name: Khorsed, age: 23, city: Dhaka ,
Country: Bangladesh}

       person.addAll({'gender':"male",'eduation':'bsc'});// added more
than one element into Map
       print(person);// output:{name: Khorsed, age: 23, city: Dhaka,
Country: Bangladesh, gender: male, eduation: bsc}

       // (.remove())-> used to remove only one element
        person.remove('gender');
        print(person);// output:{name: Khorsed, age: 23, city: Dhaka,
Country: Bangladesh, eduation: bsc}

        // (.clear())-> used to clear full map
        person.clear();
        print(person);// output:{}
    }
```

25. Dart Control Flow Statement Overview

~Dart control statements are used to control the flow of dart

1. Decision making statement
    1. if
    2. if-else
    3. if else if
    4. switch
2. Looping Statement
    1. for
    2. while
    3. do-while
3. Jump Statement
    1. Break
    2. Continue
    3. goto

26. If else Statement

```
@26_If_Else_Statement.dart
   void main()
    {
     var mark=39;
     if(mark>=80)
     {
       print("Result is A+");
     }
      else if(mark<80 && mark>=70)
      {
       print("Result is A--");
      }
      else if(mark<70 && mark>=60)
      {
       print("Result is B");
      }
       print("Result is C");
      }
      else if(mark<50 && mark>=40)
      {
       print("Result is D");
      }
      else if(mark<40)
      {
       print("Result is F");
      }

    }
```

27. Switch case Statement

```
@27_Switch_case_Statement.dart
   void main()
   {
    var X=5;
     switch(X)
     {
       case 0:
       {
         print("Zero");
         break;
       }
        case 1:
       {
         print("One");
         break;
       }
        case 2:
       {
         print("Two");
         break;
       }
        case 3:
       {
         print("Three");
        break;
       }
       default:
       {
       print("invalid");
       break;
       }
     }
   }
```

28. For Loop... Entry

```
     @28_For_Loop.dart
        void main()
        {
          for(var i=1;i<100;i++)
         {
           print("Hello World");// output:100 times "Hello World"
          }
        }
```

29. For in Loop Over List

~ For..in loop Slightly different from For Loop
~ Its work inside an array or list
~ Array/List er protek
~ For...IN loop syntax:

```
        for(var variable_name in variable_name)
        {
            //var oneAlphebet in AlphaList;for..in loop condition
        }
    @29_For_in_Loop_over_List.dart
        void main()
         {
           var AlphaList=['A','B','C','D','E','F','G'];
           for(var OneAlphabet in AlphaList)
           {
             print(OneAlphabet);
           }

         }
```

30. For In Loop Over Set and Json List

```
    @30For_In_Loop_Over_Set.dart
       void main()
        {
          var AlphabetSet={'A','B','C','D'};
          for(var i in AlphabetSet)
          {
            print(i);// output:A B C D
          }
        }
```

~Json List: Map Inside Array

```
      @30_For_in_Loop_OverJson_List.dart
      void main()
      {
          var productlist= [
            {'name':'Soap','Price':100},
            {'name':'suger','Price':200},
            {'name':'Milk','Price':40},
            {'name':'Fish','Price':100},

          ];
          for(var oneprduct in productlist)
          {
            print(oneprduct);// print full product list
            print(oneprduct['name']);// print only product name
            print(oneprduct['Price']);// print only product price
          }
      }
```

31.While and Do While Loop

```
    @31_While_Loop.dart
       void main()
      {
        var i=0;
       while(i<19)
        {
          print(i);
          i++;
        }
      }

    @31_Do_While_Loop.dart
         void main()
          {
              var i=0;
               do{
                print(i);
                i++;
               }while(i<20);
          }
```

32. Function Define and Call

~function: Group of code

1. Return Type: It can be any datatype(int,float,void)
2. func_name: Should be an appropriate and valid identifier.
3. Parameter_List: It denotes the list of the parameters which necessary when we call a function.
4. Return Value: A function return a value after complete it execution.

```
    @32_Function_Define_and_Call.dart\
       void  addTwoNumber()// functin define
       {
         var a=2;
         var b=3;
          print(a+b);
       }
       void main()
       {
         addTwoNumber();// Function call
       }
```

33. Passing Argument Inside Function

```
    @33_Passing_Argument_Inside_Function.dart

      void addTwoNum(int a,int b)
      // accept two number as a argument or parameter
      {
        print(a+b);// output:7
      }
      void main()
      {
        addTwoNum(3,4);// passing two integer number as a argument or
parameter
```

```
      }
```

34. Function Return and Main Function

~Main Function:
1. Top level Function of the dart
2. Most important and vital function of dart program
3. Execution start from Main() function
4. Main function can be used only one time in a program

@34_Function_Return_and_Main_Function.dart

```
int addTwoNum(int a,int b)
{
  var x=a+b;
  return x;// return value

}
void main()
{
  var result=addTwoNum(4, 4);// passing two argument
  print(result);// output:8
}
```

35. OOP concept and Class

~Dart is an Object Oriented Programming
1. Class
2. Object
3. Inheritance
4. Polymorphism
5. Interface
6. Abstract Class

~ Dart Class:
1. Dart classes are defined as the Blueprint of the Associated Object.
2. CLASS- User Defined data type that describes the (Characteristics and Behavior)
3. To get Properties of the class,we must create an object of that class.

@35_OOP_Concept_and_Class.dart

```
class Myclass{

}
void main()
{

}
```

36. Access Properties from Class

@36_MyClass.dart

```
  class MyClass{
  var MyName='Khorsed Alam';
  var Alphabet=['A','B','C','D'];
  addTwoNumber(int x,int y)
  {
    print(x+y);
  }
  addThreeNumber(int x ,int y,int z)
  {
    print(x+y+z);
  }
}
```

```
    @36_Access_Properties_from_Class.dart
      import '36_MyClass.dart';
      void main()
      {
          var obj = new MyClass();
          obj.addTwoNumber(10, 20);// output:30
          obj.addThreeNumber(10, 20, 30);// output:60
          print(obj.MyName);// output:Khorsed Alam
          print(obj.Alphabet);// output:[A, B, C, D]
          print(obj.Alphabet[0]);//output:A(first element of array)

      }
```

37. Access Static Properties from Class

```
    @37_Access_Static_Properties_From_Class.dart
      class MyClass{
       static var MyName='Khorsed Alam';
       static var Alphabet=['A','B','C','D'];
        static addTwoNumber(int a,int b)
       {
        print(a+b);
       }
       static addThreeNumber(int x,int y,int z)
       {
        print(x+y+z);
       }
    }
    @37_Access_Static_Properties_From_Class_Main_Function.dart
    import '37_Access_Static_Properties_From_Class.dart';
    void main()
    {
       print(MyClass.Alphabet);// output:[A, B, C, D]
       print(MyClass.Alphabet[1]);// output:B(element of 1th index)
       print(MyClass.MyName);// output:Khorsed Alam
       print(MyClass.addTwoNumber(3, 4));// output:7
       print(MyClass.addThreeNumber(2, 4,5));// output:11
    }
```

38. Details On Class Constructor

~A Constructor is a different type of function which created with same name as it class name.

1. Constructor has no Return Type
2. Constructor can have Parameter
3. Constructor execute automatically

```
      @38_Datails_On_Class_Constructor.dart
          class MyClass{
             MyClass(String msg)
             {
               print(msg);
             }
           }
      @38_Datails_On_Class_Constructor_Main_Function.dart
          import '38_Datails_On_Class_Constructor.dart';
          void main()
          {
             var obj=MyClass('Khorsed');//Parameter pass inside
constructor
          }
```

39. Dart This Kyeword

~This Keyword is used to refer the Current class Object

~This Keyword Indecate the Current instance of the Class,Methods,or Constructor.

```
    @39_Dart_This_Keyword.dart
      class MyClass{
       var num1=2;
       var num2=4;
       addTwoNumber()
       {
         var result=this.num1+this.num2;
          print(result);// output:6
       }
       MyFunction()
       {
         this.addTwoNumber();
       }
     }

    @39_Dart_This_Keyword_Main_Function.dart
        import '39_Dart_This_Keyword.dart';
        void main()
        {
          var obj=new MyClass();
          obj.addTwoNumber();
          obj.MyFunction();
        }
```

40. Inheritance Concept

~ Dart Inheritance is Defined as the Process of Deriving the Properties and Characteristics of another Class.

~Parant Class:A Class is Which Inherited by the Other Class is Called SuperClass or Parant Class.

~ Child Class: A Class Which Inherits Properties from other class is called Child Class or SubClass.

```
       @40_Inheritance_Concept.dart
         class Father{
         BaperTaka()
         {
           print("Total amount:1000000000");
         }
       }
       class Son extends Father
       {

       }
       void main()
       {

          var Fatherobj= new Father();
          Fatherobj.BaperTaka();
          // Inherite by Son
         var Sonobj= new Son();
         Sonobj.BaperTaka();
       }
```

41. Method Overriding

~ When we declare the same method in the subclass which is previously define in the SuperClass is known as the method Overriding

```
    @41_Method_Overriding.dart
         class Father{
          BaperTaka()
          {
            print("Total amount:10000000");
          }
        }
        class Son extends Father{
          BaperTaka()
          {
            print("Total Amount:4000000");
          }
        }
        void main()
        {
          var FatherObj= new Father();
          FatherObj.BaperTaka();

          var Sonobj= new Son();
          Sonobj.BaperTaka();

}
```

42. Dart Abstract

```
     ~Remove SuperClass Information
  @42_Dart_Abstract_Father.dart
     abstract class Father{
       BaperTaka()
       {
         print("Total Amount:500000");
       }
     }
   @42_Dart_Abstract_Son.dart
       import '42_Dart_Abstract_Father.dart';
      class Son extends Father{
        @override
        BaperTaka() {
           print("Total Amount:400000");
        }

      }
    @42_Dart_Abstract_Main_Function.dart
    import '42_Dart_Abstract_Son.dart';
     void main()
     {
       var Sonobj=new Son();
       Sonobj.BaperTaka();
     }
```

43. Dart Debugging

```
   ~Find out Mistake
   ~ Check code
   ~ Understand complex program flow
   ~ Work with complex action part by part
   ~ Improve code
   @43_Dart_Debugging.dart
```