DART COMMANDS COMPLETE DOCUMENTATION

```
/*DATA TYPES
 *int ,double
 int age=10;
 var age =10;
 strings
 String name="henry";
 var name ="henry";
 booleans
 lists/arrays
 maps
 runes
 Symbols
 */


/*STRING INTERPOLATION
 * 2,"john",4.5 are all literals
  var isCool=true;
  String s1="single";
  String s2="double";
  String s3='It\'s easy';
  String s4="Its's easy";
  String s5=" vhxjdhgvjjjjjj""jjjjgxcvfxvv" "vvvvvvvvvvvvvvv";//No need to add plus sign dart
automatically understands

  String name="ananya";
    print("my name is $name");
    print("characters is" + name.length.toString());
    alter
    print("my name number of characters is ${name.length}");

    Also for numbers:
    var l=10;
  var b=20;
  print("sum of $l and $b is ${l+b}");
```

```
* */


/*
 * CONSTANTS IN DART
 * final name="peter";
 * const PI =3.14;
 Final variable can only be set once and it is initialized only when accessed
 Const variable is implicitly final but it is compile time constant
 */

/*
 * CONDITIONAL IF ELSE
 * condition? exp1 :exp2
 *
 * exp1 ?? exp2
 * if exp1 is not null then evaluate exp1 else evaluate exp2
 *
 * */

/*
 LOOPS

 for(var i=0;i<4;i++){
 print("hello");
 }

 FOR -IN LOOP

list planetList ={"mercury"."venus","earth"};

for (String planet in planetList)
{
print(planet);
}
 * */



OPTIONAL PARAMETERS:
void main() {
  Cities("usa");
```

```
}
void Cities(String name1,[String name2]){
  print("Name1 is $name1");
  print("Name2 is $name2");}
```

NAMED PARAMETER:

```
void main() {
  var result=volume(2,h:7,b:5);
  print(result);
}
int volume(int l,{int b,int h})
{
  return l*b*h;
}
```

EXCEPTIONAL HANDLING:

- //catch clause is used when the type of error is unknown

```
try{
  int result = 12~/0;
  print(result);
  }catch(e){
    print("Cannot divide by zero");
  }
```

- //if type of exception is known then use ON clause:

```
try{
  int result = 12~/0;
  print(result);
  }on IntegerDivisionByZeroException{
    print("Cannot divide by zero");
  }
```

- //to trace the stack before throwing exception

```
try{
int result = 12~/0;
  print(result);
  }catch(e,s){
    print("Cannot divide by zero");
```

```
    print("Stack trace \n $s");
  }

}
```

CONSTRUCTOR:

The code in the constructor always executes before the class functions.

LAMBDA FUNCTION

```
void main() {

Function add=  (int a,int b){
   var sum =a+b;
   print(sum);
 };

 var multiply = (int num){
   return num*4;
 };
  Addnumbers(2,5);
}
 void Addnumbers(int a,int b)
 {
   var sum =a+b;
   print(sum);
 }
```

HIGHER ORDER FUNCTION

```
void main() {

Function add=  (int a,int b){
   var sum =a+b;
   print(sum);
 };

  somefun("hello",add);
}
 void somefun(String message,Function myfunction)
 {
   print(message);
   myfunction(2,4);
 }
```

```
Function task()
{
  Function mul = (int num) => num*4;
  return mul;
}
```

CLOSURE IS A FUNCTION WHICH CAN ACCESS TO THE PARENT SCOPE EVEN WHEN THE FUNCTION IS CLOSED.
TO MODIFY A VARIABLE IN A FUNCTION THAT IS DEFINED OUTSIDE THE FUNCTION

LIST

```
List<int> numberlist =List();
numberlist.add(77);
numberlist.remove(77);
numberlist.removeAt(41,2);//number and index
```

SET
Unordered collection of list
Hashset is the implementation of ul,we can include ony unique element

```
Set<String> countries =Set.from([“usa”,”india”])
countries.remove(“usa”);
countries.isEmpty;
countries.length;
countries.clear();
```

HASHMAP:
Key cannot be repeated but value can be repeated

```
void main() {
   Map<String,int> code={
     "usa" :1,
     "India":12,                                          //first method for map
     "pakistan":92;
   };


  Map<String,String> fruits =Map();                    //other method for map
    fruits["apple"]="red";
  fruits["banana"] ="yellow";

  fruits.containsKey("apple");
  fruits.update("apple",(value)=>"green");                    //different functions
  fruits.remove("apple");
  fruits.isEmpty;
  fruits.length;
```

```
  fruits.clear();

  print(fruits["apple"]);
  for(var key in fruits.keys)
  {
    print(key);
  }
  for(var value in fruits.values)
  {
    print(value);
  }

  fruits.forEach((key,value)=> print("key: $key and value :$value"));
}
```