Learn with Rabbil

DART SET

- The Dart Set is the unordered collection of the different values of the same type
- It has much functionality, which is the same as an array, but it is unordered.
- Set doesn't allow storing the duplicate values.
- Set must contain unique values.

ADD ELEMENT INTO SET

- The Dart provides the two methods add() and addAll() to insert an element into the given set.

```
void main() {
  var names = <String>{"James","Ricky", "Devansh","Adam"};
  names.add("Jonathan");
  print(names);
}
```

```
void main() {
  var names = <String>{"James","Ricky", "Devansh","Adam"};
  names.addAll({"A","B"});
  print(names);
}
```

ACCESS THE SET ELEMENT

- Dart provides the elementAt() method, which is used to access the item by passing its specified index position.

```
void main() {
  var names = <String>{"James","Ricky", "Devansh","Adam"};
  var x = names.elementAt(3);
  print(x);
}
```

DART REMOVE ALL SET ELEMENT

- We can remove entire set element by using the clear() methods.

```
void main() {
  var names = <String>{"James","Ricky", "Devansh","Adam"};
  names.clear();
  print(names);
}
```

Learn with Rabbil

DART SET PROPERTIES

Properties	Explanations
first	It is used to get the first element in the given set.
isEmpty	If the set does not contain any element, it returns true.
isNotEmpty	If the set contains at least one element, it returns true
length	It returns the length of the given set.
last	It is used to get the last element in the given set.
hashcode	It is used to get the hash code for the corresponding object.
Single	It is used to check whether a set contains only one element.

DART MAP

- Dart Map is an object that stores data in the form of a key-value pair.
- Each value is associated with its key, and it is used to access its corresponding value.
- Both keys and values can be any type.
- In Dart Map, each key must be unique, but the same value can occur multiple times.
- Dart Map can be defined in two methods.
- Using Map Literal
- Using Map Constructor

DART MAP USING MAP LITERAL

- To declare a Map using map literal, the key-value pairs are enclosed within the curly braces "{}" and separated by the commas.

```
void main() {
  var student = {'name':'Tom','age':'23'};
  print(student);
}
```

ADDING VALUE AT RUNTIME

- To declare a Map using map literal, the key-value pairs are enclosed within the curly braces "{}" and separated by the commas.

```
void main() {
  var student = {'name':' tom', 'age':23};
  student['course'] = 'B.tech';
  print(student);
}
```

USING MAP CONSTRUCTOR

- To declare the Dart Map using map constructor can be done in two ways.
- First, declare a map using map() constructor. Second, initialize the map.

```
void main() {
  var student = new Map();
  student['name'] = 'Tom';
  student['age'] = 23;
  student['course'] = 'B.tech';
  student['Branch'] = 'Computer Science';
  print(student);
}
```

MAP PROPERTIES

Properties	Explanation
Keys	It is used to get all keys as an iterable object.
values	It is used to get all values as an iterable object.
Length	It returns the length of the Map object.
isEmpty	If the Map object contains no value, it returns true.
isNotEmpty	If the Map object contains at least one value, it returns true.

MAP METHODS

Learn with Rabbil

- addAll() - It adds multiple key-value pairs of other. The syntax is given below.

```
void main() {
  Map student = {'name':'Tom','age': 23};
  print(student);
  student.addAll({'dept':'Civil','email':'tom@xyz.com'});
  print(student);
}
```

- Map.clear() - It eliminates all pairs from the map.

```
void main() {
  Map student = {'name':'Tom','age': 23};
  print(student);
  student.addAll({'dept':'Civil','email':'tom@xyz.com'});
  student.clear();
  print(student);
}
```

MAP METHODS

- remove() - It removes the key and its associated value if it exists in the given map.

```
void main() {
  Map student = {'name':'Tom','age': 23};
  print(student);
  student.remove('age');
  print(student);
}
```

Learn with Rabbil

DART CONTROL FLOW STATEMENT

The control statements or flow of control statements are used to control the flow of Dart program. In Dart, Control flow statement can be categorized mainly in three following ways.

- Decision-making statements
- Looping statements
- Jump statements

DART DECISION-MAKING STATEMENTS

The Decision-making statements allow us to determine which statement to execute based on the test expression at runtime. Dart provides following types of Decision-making statement.

- If Statement
- If-else Statements
- If else if Statement
- Switch Case Statement

IF STATEMENT

If statement allows us to a block of code execute when the given condition returns true.

```
void main() {
   var n = 35;
   if (n<40){
     print("The number is smaller than 40");
   }
}
```

IF-ELSE STATEMENTS

In Dart, if-block is executed when the given condition is true. If the given condition is false, else-block is executed.

```
void main() {
 var x = 20;
 var y = 30;
 if(x > y){
  print("x is greater than y");
 } else {
  print("y is greater than x");
 }
}
```

Learn with Rabbil

IF ELSE-IF STATEMENT

- Dart if else-if statement provides the facility to check a set of test expressions and execute the different statements.
- It is used when we have to make a decision from more than two possibilities.

```
void main() {
 var marks = 74;
 if(marks > 85)
 {
  print("Excellent");
 }
 else if(marks>75)
 {
  print("Very Good");
 }
 else if(marks>65)
 {
  print("Good");
 }
 else
 {
  print("Average");
 }
}
```

SWITCH CASE STATEMENT

- Dart Switch case statement is used to avoid the long chain of the if-else statement.
- It is the simplified form of nested if-else statement.

```
void main() {
 int n = 3;
 switch (n) {
  case 1:
   print("Value is 1");
   break;
  case 2:
   print("Value is 2");
   break;
  case 3:
   print("Value is 3");
   break;
  case 4:
   print("Value is 4");
   break;
  default:
   print("Out of range");
   break;
 }
}
```

Learn with Rabbil

DART LOOPING STATEMENTS

Dart Loop is used to run a block of code repetitively for a given number of times or until matches the specified condition.

- Dart for loop
- Dart for...in loop
- Dart while loop
- Dart do-while loop

DART FOR LOOP

The for loop is used when we know how many times a block of code will execute.

```
for(int i=0; i<=10; i++){
 print(i);
}
```

DART FOR... IN LOOP OVER LIST

The for...in loop is slightly different from the for loop. It only takes dart object or expression as an iterator and iterates the element one at a time.

```
var list1 = [10,20,30,40,50];
for(var i in list1) {
   print(i);
}
```

DART FOR... IN LOOP OVER MAP (JSON ARRAY)

```
var student = [
{'name':'Rain', 'age':36},
{'name':'Rupom', 'age':31},
{'name':'Rifat', 'age':18}
];

for(var i in student) {
   var name=i['name'];
   print(name);
}
```

DART FOR... IN LOOP OVER SET

```
void main() {
 var names ={"James","Ricky", "Devansh","Adam"};
 for(var i in names) {
    print(i);
 }
}
```

FUNCTION PARTS

return_type - It can be any data type such as void, integer, float, etc. The return type must be matched with the returned value of the function.

func_name - It should be an appropriate and valid identifier.

parameter_list - It denotes the list of the parameters, which is necessary when we called a function.

return value - A function returns a value after complete its execution.

```
int myFunc(int a, int b){
 int c;
 c = a+b;
 return c;
}

 void main() {
  var res=myFunc(10,20);
  print(res);
 }
```

FUNCTION PARTS

return_type - It can be any data type such as void, integer, float, etc. The return type must be matched with the returned value of the function.

func_name - It should be an appropriate and valid identifier.

parameter_list - It denotes the list of the parameters, which is necessary when we called a function.

return value - A function returns a value after complete its execution.

```
int myFunc(int a, int b){
  int c;
  c = a+b;
  return c;
}

void main() {
   var res=myFunc(10,20);
   print(res);
}
```

DEFINING A FUNCTION

- A function can be defined by providing the name of the function with the appropriate parameter and return type.
- A function contains a set of statements which are called function body.

```
myFunc(){


}
```

CALLING A FUNCTION

After creating a function, we can call or invoke the defined function inside the main() function body

```
myFunc(){

}
void main() {
 myFunc();
}
```

PASSING ARGUMENTS TO FUNCTION

When a function is called, it may have some information as per the function prototype is known as a parameter (argument).

```
myFunc(String Name){
 print(Name);
}
void main() {
 myFunc("RABBIL");
}
```

FUNCTION RETURN & RETURN TYPE

- It can be any data type such as void, integer, float, etc. The return type must be matched with the returned value of the function.

- A function returns a value after complete its execution.

```
int myFunc(int a, int b){
 int c;
 c = a+b;
 return c;
}
```

```
void main() {
 var res=myFunc(10,20);
 print(res);
}
```

Learn with Rabbil

THE MAIN() FUNCTION

- The main() function is the top-level function of the Dart.
- It is the most important and vital function of the Dart programming language.
- The execution of the programming starts with the main() function.
- The main() function can be used only once in a program.

```
void main() {


}
```