Dart - Loops

A looping statement in Dart or any other programming language is used to repeat a particular set of commands until certain conditions are not completed. There are different ways to do so. They are:

- for loop
- for... in loop
- for each loop
- while loop
- do-while loop

Control flow:

Control flow goes as:

1. initialization
2. Condition
3. Body of loop
4. Test expression

For Loops

For loop in Dart is similar to that in Java and also the flow of execution is the same as that in Java.

Syntax:

```
  for(initialization; condition; text expression){
     // Body of the loop
 }
```

for…in loop

For...in loop in Dart takes an expression or object as an iterator. It is similar to that in Java and its execution flow is also the same as that in Java.

Syntax:

```
 for (var in expression) {
   // Body of loop
}
```

for each ... loop

The for-each loop iterates over all elements in some container/collectible and passes the elements to some specific function.

Syntax:

```
 collection.foreach(void f(value))
```

while loop

The body of the loop will run until and unless the condition is true.

Syntax:

```
 while(condition){
    text expression;
    // Body of loop
}
```

do..while loop

The body of the loop will be executed first and then the condition is tested.

Syntax:

```
 do{
    text expression;
    // Body of loop
}while(condition);
```

Dart – Loop Control Statements (Break and Continue)

Dart supports two types of loop control statements:

1. Break Statement
2. Continue Statement

Break Statement:

This statement is used to break the flow of control of the loop i.e if it is used within a loop then it will terminate the loop whenever encountered. It will bring the flow of control out of the nearest loop.

Syntax:

```
break;
```

Example -

```
void main()
{
    int count = 1;

    while (count <= 10) {
        print("Hi, you are inside loop $count");
        count++;

        if (count == 4) {
            break;
        }
    }
    print("Hi, you are out of while loop");
}
```

Output:

```
Hi, you are inside loop 1
Hi, you are inside loop 2
Hi, you are inside loop 3
Hi, you are out of while loop
```

Continue Statement:

While the **break** is used to end the flow of control, **continue** on the other hand is used to continue the flow of control. When a continue statement is encountered in a loop it doesn’t terminate the loop but rather jump the flow to next iteration.

Syntax:

```
continue;
```