Control Flow

For-in loop for collections | Break and Continue usage | Nested loops (multiplication table)

Iterating Through Collections with for-in

Cleaner and more readable than a traditional for loop. Ideal when you don’t need the index of each element.

Syntax:

```
for (var element in collection) {
  // use element
}
```

Key points:

- The for-in loop is used to iterate over iterable collections like List, Set, or Map.values.
- It automatically goes through each element in sequence.

Controlling Loop Flow with break and continue

- **`break`** → Immediately exits the loop.
- **`continue`** → Skips to the next iteration of the loop.

Example:

```
void main() {
  for (var i = 1; i <= 5; i++) {
    if (i == 3) continue; // skip number 3
    if (i == 5) break; // stop the loop
    print(i);
  }
} // Output: 1 2 4
```

Using Nested Loops to Generate a Table - 1/2

- A nested loop is a loop inside another loop.
- Commonly used for tables or combinations (e.g., multiplication table).

Example:

```
void main() {
  for (var i = 1; i <= 3; i++) {
    for (var j = 1; j <= 3; j++) {
      print('$i × $j = ${i * j}');
    }
    print('---');
  }
}
```

Using Nested Loops to Generate a Table - 2/2

- The outer loop controls rows, and the inner loop controls columns.
- Useful for working with grids, charts, and matrices.

Example:

```
1 × 1 = 1
1 × 2 = 2
1 × 3 = 3
---
2 × 1 = 2
2 × 2 = 4
2 × 3 = 6
---
3 × 1 = 3
3 × 2 = 6
3 × 3 = 9
---
```