Looping Statements in TypeScript

Looping statements in TypeScript allow executing a block of code multiple times based on a condition. Below are the corrected syntax and examples for each loop.

1. while Loop

The while loop executes a block of code **as long as** the condition is true.

Syntax:

```
while (condition) {
   // Code to execute
}
```

Example:

```
let i: number = 1;
while (i <= 5) {
   console.log(i);
   i++;
}
```

Output:

```
1
2
3
4
5
```

- **The loop checks the condition before execution. If i <= 5, it runs; otherwise, it stops.**

2. do-while Loop

The do-while loop **executes the code at least once** before checking the condition.

Syntax:

```
do {
   // Code to execute
} while (condition);
```

Example:

```
let j: number = 1;
do {
   console.log(j);
   j++;
} while (j <= 5);
```

Output:

```
1
2
3
4
5
```

- **Executes once even if the condition is false.**

3. for Loop

The for loop is useful when the number of iterations is known.

Syntax:

```
for (initialization; condition; increment/decrement) {
   // Code to execute
}
```

Example:

```
for (let k: number = 1; k <= 5; k++) {
   console.log(k);
}
```

Output:

1

2

3

4

5

- **Includes initialization, condition check, and increment/decrement in a single line.**

4. break Statement

The break statement **stops the loop immediately** when a condition is met.

Syntax:

```
for (initialization; condition; increment/decrement) {
   if (someCondition) {
      break;
   }
   // Code to execute
}
```

Example:

```
for (let n: number = 1; n <= 10; n++) {
   if (n === 5) {
      break; // Exits the loop when n is 5
   }
   console.log(n);
}
```

Output:

1

2

3

4

- **The loop exits when n reaches 5.**

5. continue Statement

The continue statement **skips the current iteration** and moves to the next one.

Syntax:

```
for (initialization; condition; increment/decrement) {
   if (someCondition) {
      continue;
   }
   // Code to execute
}
```

Example:

```
for (let m: number = 1; m <= 5; m++) {
   if (m === 3) {
      continue; // Skips when m is 3
   }
   console.log(m);
}
```

Output:

```
1
2
4
5
```

- **Skips 3 and continues with the next iteration.**

Comparison: while Loop vs do-while Loop

Feature	while Loop	do-while Loop
Condition Check	Before execution	After first execution
Execution Guarantee	May not execute if the condition is false initially	Executes at least once

Example Demonstration:

```
let x: number = 5;

while (x < 5) {
   console.log("Inside while loop"); // Won't run because x is not < 5
}


do {
   console.log("Inside do-while loop"); // Runs once before checking condition
} while (x < 5);
```

Output:

```
Inside do-while loop
```

Conclusion

- ✅ **Use while** when you want to check the condition before execution.
- ✅ **Use do-while** when you need the loop to run at least once.
- ✅ **Use for** when you know how many times the loop should run.
- ✅ **Use break** to stop a loop early.
- ✅ **Use continue** to skip an iteration and proceed to the next one.

Lab Assignments

While loop

1. Write a program to calculate the sum of the first 10 natural numbers using a while loop.
2. Write a program to calculate the factorial of a given number using a while loop.
3. Write a program to reverse a given number using a while loop.
4. Write a program to check if a given number is a prime number using a while loop.
5. Write a program to find the largest digit in a given number using a while loop.
6. Write a program to check if a given number is a palindrome using a while loop.

Do-while loop

7. Write a program to print numbers from 1 to 10 using a do-while loop.

8. Write a Java program that performs basic arithmetic operations (addition, subtraction, multiplication, and division) using a do-while loop until the user chooses to exit.

For loop

9. Print Multiples of 5 from 5 to 50
10. Print Prime Numbers between 1 and 50
11. Print Sum of Even Numbers between 1 and 20
12. Print Sum of Odd Numbers between 1 and 20
13. Print Table of 7
14. Print Numbers Divisible by 3 and 5 from 1 to 100
15. Count Number of Digits in a Number
16. Find Sum of Digits in a Number
17. Print Multiples of 7 between 1 and 100
18. Calculate the sum of all even numbers from 1 to N.

Continue

19. Write a program to print the odd numbers from 1 to 20 using a for loop. Use the continue statement to skip even numbers.

20. Write a program to print numbers from 1 to 30, but skip numbers that are multiples of 5. Use the continue statement within a while loop.

Break

21. Write a program to find and print the first even number between 1 and 10 using a for loop. Use the break statement to exit the loop as soon as you find the first even number.

22. Write a program to print numbers from 1 to 30. Stop printing and exit the loop when you find a number greater than 15. Use the break statement within a for loop.