JavaScript Lab: For and While Loops

Objective:

By the end of this lab, you will understand how to use `for` and `while` loops in JavaScript to perform repetitive tasks.

Part 1: For Loops

A `for` loop repeats a block of code a specified number of times. It is ideal when you know how many times you want to run the loop.

Syntax of a For Loop:

```
for (initialization; condition; update) {
    // Code to execute in each iteration
}
```

- **Initialization:** Usually a variable is initialized to a starting value.
- **Condition:** The loop runs as long as this condition is true.
- **Update:** This changes the variable in the initialization section after each iteration.

Example 1:

Print numbers from 1 to 5:

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

In this example, the loop starts at `i = 1`, and runs as long as `i <= 5`. After each iteration, `i` is incremented by 1.

Example 2:

Print even numbers between 1 and 10:

```
for (let i = 1; i <= 10; i++) {
    if (i % 2 === 0) {
        console.log(i);
    }
}
```

Here, the loop runs from 1 to 10 and checks if `i` is even by using the condition `i % 2 === 0`.

Exercise 1: Print Numbers

Write a program that prints numbers from 1 to 10 using a `for` loop.

```
// Write your for loop here
```

Exercise 2: Print Multiplication Table

Write a JavaScript program that prints the multiplication table of 5 using a `for` loop.

```
// Write your for loop here
```

Part 2: While Loops

A `while` loop repeats a block of code as long as a specified condition is true. It is useful when the number of iterations is unknown or based on dynamic input.

Syntax of a While Loop:

```
while (condition) {
    // Code to execute as long as the condition is true
}
```

Example 3:

Print numbers from 1 to 5 using a `while` loop:

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

In this example, the loop checks the condition `i <= 5`. As long as the condition is true, it prints the value of `i` and increments it by 1 after each iteration.

Example 4:

Keep asking for user input until a correct answer is provided:

```
let answer;
while (answer !== "yes") {
    answer = prompt("Do you want to continue? (yes/no)");
}
```

Here, the loop continues asking the user for input until they type "yes".

Exercise 3: Sum of Numbers

Write a JavaScript program that uses a `while` loop to find the sum of numbers from 1 to 100.

```
// Write your while loop here
```

Part 3: Do-While Loops (ADVANCE)

A `do-while` loop is similar to a `while` loop, but it ensures that the code runs at least once, even if the condition is initially false.

Syntax of a Do-While Loop:

```
do {
    // Code to execute
```

```
} while (condition);
```

Example 5:

Print numbers from 1 to 5 using a `do-while` loop:

javascript
Copy code

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

In this example, the code inside the loop executes at least once before checking the condition.

Exercise 4: User Input Validation

Write a program that keeps asking for a number between 1 and 10 using a `do-while` loop. It should only stop when the user enters a valid number.

javascript
Copy code

```
// Write your do-while loop here
```

Part 4: Breaking and Continuing in Loops (ADVANCE)

Sometimes, you may want to stop a loop early or skip certain iterations. This can be done using `break` and `continue` statements.

- **Break Statement:** Stops the loop entirely.
- **Continue Statement:** Skips the rest of the current iteration and moves to the next iteration.

Example 6:

Breaking a loop when a number is found:

```
for (let i = 1; i <= 10; i++) {
    if (i === 5) {
        break; // Stops the loop when i equals 5
    }
    console.log(i);
}
```

This loop prints numbers from 1 to 4, but stops when `i` is 5.

Example 7:

Skipping an iteration using `continue`:

```
for (let i = 1; i <= 10; i++) {
    if (i === 5) {
        continue; // Skips the iteration when i equals 5
    }
    console.log(i);
}
```

This loop prints numbers from 1 to 10 but skips printing 5.

Exercise 5: Breaking out of a Loop

Write a program that prints numbers from 1 to 10, but stops when it reaches 7.

```
// Write your for loop here with break
```

Exercise 6: Skip Even Numbers

Write a program that prints numbers from 1 to 10, but skips even numbers using `continue`.

```
// Write your for loop here with continue
```

Part 5: Nested Loops

You can place loops inside other loops, which is called nesting. This is useful for tasks like working with multi-dimensional arrays or generating patterns.

Example 8:

Print a 3x3 grid of numbers:

javascript
Copy code
```
for (let i = 1; i <= 3; i++) {
    for (let j = 1; j <= 3; j++) {
        console.log(`Row ${i}, Column ${j}`);
    }
}
```

This example uses two nested `for` loops to print out a 3x3 grid, where the outer loop controls the rows and the inner loop controls the columns.

Exercise 7: Print a Right-Angled Triangle

Write a program that uses nested loops to print the following right-angled triangle pattern:

markdown
Copy code
```
*
**
***
****
*****
```

javascript
Copy code
```
// Write your nested loop here
```

Summary:

- You learned how to use `for`, `while`, and `do-while` loops in JavaScript to repeat tasks.
- You practiced breaking and continuing loops as well as using nested loops to solve more complex problems.

Complete the exercises and ensure that your loops behave as expected with different input values.