**Lesson 5: Loops in JavaScript (for, while, do...while)**

By the end of this lesson, students will be able to:

* Understand the Purpose of Loops:
  + Explain the need for loops in programming and describe the differences between for, while, and do...while loops.
* Implement for Loops:
  + Write and execute JavaScript programs using for loops to perform repetitive tasks where the number of iterations is known.
* Utilise while and do...while Loops:
  + Develop JavaScript programs that use while and do...while loops to repeat code blocks based on dynamic conditions.
* Apply Nested Loops:
  + Create JavaScript programs using nested loops to handle multi-dimensional data and complex iteration scenarios.

**Loops**

Loops are fundamental programming constructs that allow you to execute a block of code repeatedly based on a specified condition. Instead of writing the same code multiple times, loops enable you to run the same operation with different values efficiently.

JavaScript provides three primary types of loops:

* for loop: Repeats a block of code a known number of times.
* while loop: Repeats a code block if a specified condition is true.
* do...while loop: Similar to the while loop, it guarantees that the code block will be executed at least once before checking the condition.
* for Loop

The for loop is commonly used when you know the exact number of iterations you need. It consists of three parts:

* Initialisation: Setting up a counter variable.
* Condition: The loop runs if this condition is true.
* Increment/Decrement: Updates the counter variable after each iteration.

Syntax:

for (initialization; condition; increment/decrement) {

// Code to be executed

}

Example:

// Print numbers 1 to 5

for (let i = 1; i <= 5; i++) {

console.log(i);

}

In this example, the loop starts with i = 1 and continues to run as long as i <= 5. After each iteration, i is incremented by 1. The output will be:

1

2

3

4

5

* while Loop

The while loop is used when you want to repeat a block of code an unknown number of times as long as a specific condition remains true.

Syntax:

while (condition) {

// Code to be executed

}

Example:

// Print numbers 1 to 5

let i = 1;

while (i <= 5) {

console.log(i);

i++;

}

In this example, the loop starts with i = 1 and continues to run as long as i <= 5. The output will be:

1

2

3

4

5

* do...while Loop

The do...while loop is similar to the while loop but has one key difference: it guarantees that the code block will run at least once, even if the condition is initially false. The condition is checked after the code block has been executed.

Syntax:

do {

// Code to be executed

} while (condition);

Example:

// Print numbers 1 to 5

let i = 1;

do {

console.log(i);

i++;

} while (i <= 5);

In this example, the loop starts with i = 1, executes the block, and then checks if i <= 5. The output will be:

1

2

3

4

5

* Infinite Loops and Breaking Out of Loops

An infinite loop occurs when the loop's condition never becomes false, causing the loop to run indefinitely. This usually happens if the counter variable needs to be appropriately updated or the condition needs to be correctly specified.

Example of an Infinite Loop:

// This loop will run forever

let i = 1;

while (i > 0) {

console.log(i);

// Missing increment of i

}

To prevent infinite loops, ensure the loop's condition will eventually become false. You can use the break statement to exit a loop prematurely if necessary.

Example of Using break:

for (let i = 1; i <= 10; i++) {

if (i === 5) {

break; // Exit the loop when i equals 5

}

console.log(i);

}

The output will be:

1

2

3

4